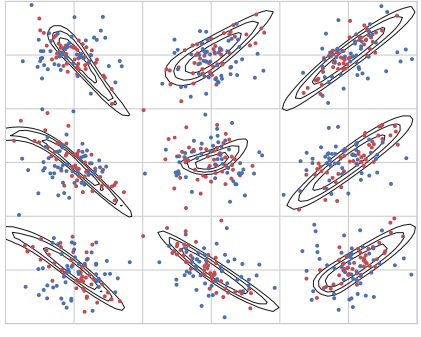
Better deep learning (sometimes) by learning kernel mean embeddings

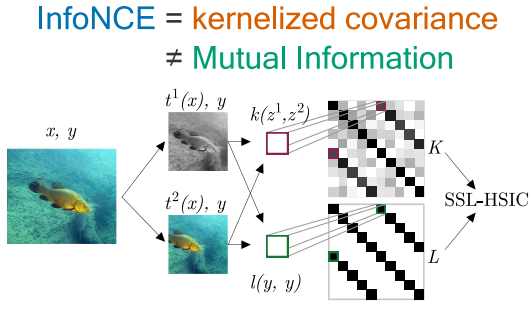
Danica J. Sutherland (she/her)

University of British Columbia (UBC) / Alberta Machine Intelligence Institute (Amii)

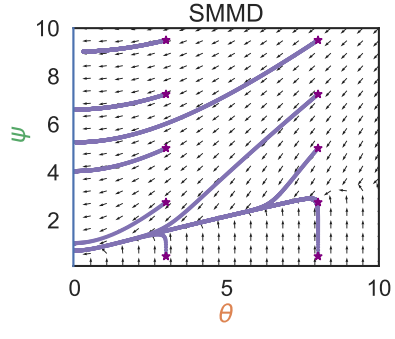
slides (but not the talk) about four related projects:



Feng Liu
Wenkai Xu
Hsiao-Yu (Fish) Tung
(+ 6 more...)



Yazhe Li
Roman Pogodin



Michael Arbel
Mikołaj Bińkowski

Li (Kevin) Wenliang
Heiko Strathmann

+ all with Arthur Gretton

LIKE22 - 12 Jan 2022

Deep learning and deep kernels

- Deep learning: models usually of form $f(x) = w^{\mathsf{T}} \phi_{\psi}(x)$
  - With a *learned* $\phi_{\psi}(x) : \mathcal{X} \to \mathbb{R}^D$

Deep learning and deep kernels

- Deep learning: models usually of form $f(x) = w^\mathsf{T} \phi_\psi(x)$
  - With a *learned* $\phi_\psi(x) : \mathcal{X} \to \mathbb{R}^D$
- If we fix ψ, have $f \in \mathcal{H}_\psi$ with $k_\psi(x, y) = \phi_\psi(x)^\mathsf{T} \phi_\psi(y)$

Deep learning and deep kernels

- Deep learning: models usually of form $f(x) = w^\mathsf{T} \phi_\psi(x)$
  - With a *learned* $\phi_\psi(x) : \mathcal{X} \to \mathbb{R}^D$
- If we fix ψ, have $f \in \mathcal{H}_\psi$ with $k_\psi(x, y) = \phi_\psi(x)^\mathsf{T} \phi_\psi(y)$
  - Same idea as NNGP approximation

Deep learning and deep kernels

- Deep learning: models usually of form $f(x) = w^\mathsf{T} \phi_\psi(x)$
  - With a *learned* $\phi_\psi(x) : \mathcal{X} \to \mathbb{R}^D$
- If we fix ψ, have $f \in \mathcal{H}_\psi$ with $k_\psi(x, y) = \phi_\psi(x)^\mathsf{T} \phi_\psi(y)$
  - Same idea as NNGP approximation
- Could train a classifier by:
  - Let $\tilde{L}(\psi) = L(f^*_\psi)$, loss of the best $f^*_\psi \in H_\psi$
  - Learn ψ by following $\nabla_\psi \tilde{L}(\psi) = \nabla_\psi L(f^*_\psi)$

Deep learning and deep kernels

- Deep learning: models usually of form $f(x) = w^\mathsf{T} \phi_\psi(x)$
  - With a *learned* $\phi_\psi(x) : \mathcal{X} \to \mathbb{R}^D$
- If we fix ψ, have $f \in \mathcal{H}_\psi$ with $k_\psi(x, y) = \phi_\psi(x)^\mathsf{T} \phi_\psi(y)$
  - Same idea as NNGP approximation
- Could train a classifier by:
  - Let $\tilde{L}(\psi) = L(f^*_\psi)$, loss of the best $f^*_\psi \in H_\psi$
  - Learn ψ by following $\nabla_\psi \tilde{L}(\psi) = \nabla_\psi L(f^*_\psi)$
- Generalize to a **deep kernel**:

$$k_\psi(x, y) = \kappa\left(\phi_\psi(x), \phi_\psi(y)\right)$$

Normal deep learning $\subset$ deep kernels

- Take $\phi_\psi(x) \in \mathbb{R}$ as output of *last* layer
- $k_\psi(x, y) = \phi_\psi(x)\phi_\psi(y)$
- Final function in $\mathcal{H}_\psi$ will be $a\phi_\psi(x)$

Normal deep learning $\subset$ deep kernels

- Take $\phi_\psi(x) \in \mathbb{R}$ as output of *last* layer
- $k_\psi(x, y) = \phi_\psi(x)\phi_\psi(y) + 1$
- Final function in $\mathcal{H}_\psi$ will be $a\phi_\psi(x) + b$

Normal deep learning $\subset$ deep kernels

- Take $\phi_\psi(x) \in \mathbb{R}$ as output of *last* layer
- $k_\psi(x, y) = \phi_\psi(x)\phi_\psi(y) + 1$
- Final function in $\mathcal{H}_\psi$ will be $a\phi_\psi(x) + b$
- With logistic loss: this is Platt scaling

Normal deep learning $\subset$ deep kernels

- Take $\phi_\psi(x) \in \mathbb{R}$ as output of *last* layer
- $k_\psi(x, y) = \phi_\psi(x)\phi_\psi(y) + 1$
- Final function in $\mathcal{H}_\psi$ will be $a\phi_\psi(x) + b$
- With logistic loss: this is Platt scaling

On Calibration of Modern Neural Networks

Chuan Guo [*1] Geoff Pleiss [*1] Yu Sun [*1] Kilian Q. Weinberger [1]

So what?

- This definitely does *not* say that deep learning is (even approximately) a kernel method

So what?

- This definitely does *not* say that deep learning is (even approximately) a kernel method
- ...despite what some people might want you think

Computer Science > Machine Learning

[Submitted on 30 Nov 2020]

Every Model Learned by Gradient Descent Is Approximately a Kernel Machine

Pedro Domingos

So what?

- This definitely does *not* say that deep learning is (even approximately) a kernel method
- ...despite what some people might want you think

Computer Science > Machine Learning

[Submitted on 30 Nov 2020]

Every Model Learned by Gradient Descent Is Approximately a Kernel Machine

Pedro Domingos

- We know theoretically deep learning can learn some things faster than any kernel method [see Malach+ ICML-21 + refs]

So what?

- This definitely does *not* say that deep learning is (even approximately) a kernel method
- ...despite what some people might want you think

- We know theoretically deep learning can learn some things faster than any kernel method [see Malach+ ICML-21 + refs]
- But deep kernel learning ≠ traditional kernel models
  - exactly like how usual deep learning ≠ linear models

What do deep kernels give us?

- In “normal” classification: slightly richer function space 🤷

What do deep kernels give us?

- In “normal” classification: slightly richer function space 🤷
- Meta-learning: common ϕ_ψ, constantly varying f^*_ψ

META-LEARNING WITH DIFFERENTIABLE CLOSED-FORM SOLVERS

Luca Bertinetto
FiveAI & University of Oxford
luca@robots.ox.ac.uk

João Henriques
University of Oxford
joao@robots.ox.ac.uk

Philip H.S. Torr
FiveAI & University of Oxford
philip.torr@eng.ox.ac.uk

Andrea Vedaldi
University of Oxford
vedaldi@robots.ox.ac.uk

Meta-Learning with Differentiable Convex Optimization

Kwonjoon Lee[2] Subhransu Maji[1,3] Avinash Ravichandran[1] Stefano Soatto[1,4]
[1]Amazon Web Services [2]UC San Diego [3]UMass Amherst [4]UCLA
kwl042@ucsd.edu {smmaji,ravinash,soattos}@amazon.com

What do deep kernels give us?

- In “normal” classification: slightly richer function space 🤷
- Meta-learning: common ϕ_ψ, constantly varying f^*_ψ

What do deep kernels give us?

- In “normal” classification: slightly richer function space 🤷
- Meta-learning: common ϕ_ψ, constantly varying f^*_ψ
- Two-sample testing
  - Simple form of f^*_ψ for cheap permutation testing

What do deep kernels give us?

- In “normal” classification: slightly richer function space 🤷
- Meta-learning: common ϕ_ψ, constantly varying f^*_ψ
- Two-sample testing
  - Simple form of f^*_ψ for cheap permutation testing
- Self-supervised learning
  - Better understanding of what's really going on, at least

What do deep kernels give us?

- In “normal” classification: slightly richer function space 🤷
- Meta-learning: common ϕ_ψ, constantly varying f^*_ψ
- Two-sample testing
  - Simple form of f^*_ψ for cheap permutation testing
- Self-supervised learning
  - Better understanding of what's really going on, at least
- Generative modeling with MMD GANs
  - Better gradient for generator to follow (?)

What do deep kernels give us?

- In “normal” classification: slightly richer function space 🤷
- Meta-learning: common ϕ_ψ, constantly varying f^*_ψ
- Two-sample testing
  - Simple form of f^*_ψ for cheap permutation testing
- Self-supervised learning
  - Better understanding of what's really going on, at least
- Generative modeling with MMD GANs
  - Better gradient for generator to follow (?)
- Score matching in exponential families (density estimation)
  - Optimize regularization weights, better gradient (?)

Maximum Mean Discrepancy (MMD)

$$\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) = \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[f(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)]$$

Maximum Mean Discrepancy (MMD)

$$\begin{aligned}\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) &= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[f(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)] \\ &= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\langle f, \varphi(X) \rangle_{\mathcal{H}}] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\langle f, \varphi(Y) \rangle_{\mathcal{H}}]\end{aligned}$$

Maximum Mean Discrepancy (MMD)

$$
\begin{aligned}
\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) &= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[f(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)] \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\langle f, \varphi(X) \rangle_{\mathcal{H}}] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\langle f, \varphi(Y) \rangle_{\mathcal{H}}] \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \left\langle f, \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\varphi(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\varphi(Y)] \right\rangle_{\mathcal{H}}
\end{aligned}
$$

Maximum Mean Discrepancy (MMD)

$$
\begin{aligned}
\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) &= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[f(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)] \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\langle f, \varphi(X) \rangle_{\mathcal{H}}] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\langle f, \varphi(Y) \rangle_{\mathcal{H}}] \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \left\langle f, \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\varphi(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\varphi(Y)] \right\rangle_{\mathcal{H}} \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \left\langle f, \mu_{\mathbb{P}}^k - \mu_{\mathbb{Q}}^k \right\rangle_{\mathcal{H}}
\end{aligned}
$$

Maximum Mean Discrepancy (MMD)

$$
\begin{aligned}
\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) &= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[f(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)] \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\langle f, \varphi(X) \rangle_{\mathcal{H}}] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\langle f, \varphi(Y) \rangle_{\mathcal{H}}] \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \left\langle f, \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\varphi(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\varphi(Y)] \right\rangle_{\mathcal{H}} \\
&= \sup_{\|f\|_{\mathcal{H}} \leq 1} \left\langle f, \mu^k_{\mathbb{P}} - \mu^k_{\mathbb{Q}} \right\rangle_{\mathcal{H}} = \left\| \mu^k_{\mathbb{P}} - \mu^k_{\mathbb{Q}} \right\|_{\mathcal{H}}
\end{aligned}
$$

MMD as feature matching

$$\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) = \left\| \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\varphi(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\varphi(Y)] \right\|_{\mathcal{H}}$$

- $\varphi : X \to \mathcal{H}$ is the *feature map* for $k(x, y) = \langle \varphi(x), \varphi(y) \rangle$

MMD as feature matching

$$\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) = \left\| \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\varphi(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\varphi(Y)] \right\|_{\mathcal{H}}$$

- $\varphi : X \to \mathcal{H}$ is the *feature map* for $k(x, y) = \langle \varphi(x), \varphi(y) \rangle$
- If $k(x, y) = x^\mathsf{T} y$, $\varphi(x) = x$, then
  the MMD is distance between means

MMD as feature matching

$$\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) = \left\| \mathop{\mathbb{E}}_{X \sim \mathbb{P}}[\varphi(X)] - \mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[\varphi(Y)] \right\|_{\mathcal{H}}$$

- $\varphi : X \to \mathcal{H}$ is the *feature map* for $k(x, y) = \langle \varphi(x), \varphi(y) \rangle$
- If $k(x, y) = x^\mathsf{T} y$, $\varphi(x) = x$, then
  the MMD is distance between means
- Many kernels: **infinite-dimensional $\mathcal{H}$**

Estimating MMD

$$\mathrm{MMD}_k^2(\mathbb{P}, \mathbb{Q}) = \mathop{\mathbb{E}}_{X, X' \sim \mathbb{P}}[k(X, X')] + \mathop{\mathbb{E}}_{Y, Y' \sim \mathbb{Q}}[k(Y, Y')] - 2 \mathop{\mathbb{E}}_{\substack{X \sim \mathbb{P} \\ Y \sim \mathbb{Q}}}[k(X, Y)]$$

Estimating MMD

$$\mathrm{MMD}_k^2(\mathbb{P}, \mathbb{Q}) = \mathop{\mathbb{E}}_{X, X' \sim \mathbb{P}}[k(X, X')] + \mathop{\mathbb{E}}_{Y, Y' \sim \mathbb{Q}}[k(Y, Y')] - 2 \mathop{\mathbb{E}}_{\substack{X \sim \mathbb{P} \\ Y \sim \mathbb{Q}}}[k(X, Y)]$$

$$\widehat{\mathrm{MMD}}_k^2(X, Y) = \mathrm{mean}(K_{XX}) + \mathrm{mean}(K_{YY}) - 2\,\mathrm{mean}(K_{XY})$$

Estimating MMD

$$\mathrm{MMD}_k^2(\mathbb{P}, \mathbb{Q}) = \mathop{\mathbb{E}}_{X,X' \sim \mathbb{P}}[k(X, X')] + \mathop{\mathbb{E}}_{Y,Y' \sim \mathbb{Q}}[k(Y, Y')] - 2 \mathop{\mathbb{E}}_{\substack{X \sim \mathbb{P} \\ Y \sim \mathbb{Q}}}[k(X, Y)]$$

$$\widehat{\mathrm{MMD}}_k^2(X, Y) = \mathrm{mean}(K_{XX}) + \mathrm{mean}(K_{YY}) - 2\,\mathrm{mean}(K_{XY})$$

K_{XX}

Estimating MMD

$$\mathrm{MMD}_k^2(\mathbb{P}, \mathbb{Q}) = \mathop{\mathbb{E}}_{X,X' \sim \mathbb{P}}[k(X, X')] + \mathop{\mathbb{E}}_{Y,Y' \sim \mathbb{Q}}[k(Y, Y')] - 2 \mathop{\mathbb{E}}_{\substack{X \sim \mathbb{P} \\ Y \sim \mathbb{Q}}}[k(X, Y)]$$

$$\widehat{\mathrm{MMD}}_k^2(X, Y) = \mathrm{mean}(K_{XX}) + \mathrm{mean}(K_{YY}) - 2\,\mathrm{mean}(K_{XY})$$

K_{XX}

K_{YY}

Estimating MMD

$$\mathrm{MMD}_k^2(\mathbb{P}, \mathbb{Q}) = \mathop{\mathbb{E}}_{X,X' \sim \mathbb{P}}[k(X, X')] + \mathop{\mathbb{E}}_{Y,Y' \sim \mathbb{Q}}[k(Y, Y')] - 2 \mathop{\mathbb{E}}_{\substack{X \sim \mathbb{P} \\ Y \sim \mathbb{Q}}}[k(X, Y)]$$

$$\widehat{\mathrm{MMD}}_k^2(X, Y) = \mathrm{mean}(K_{XX}) + \mathrm{mean}(K_{YY}) - 2\,\mathrm{mean}(K_{XY})$$

K_{XX}

1.0	0.2	0.6
0.2	1.0	0.5
0.6	0.5	1.0

K_{YY}

1.0	0.8	0.7
0.8	1.0	0.6
0.7	0.6	1.0

K_{XY}

0.3	0.1	0.2
0.2	0.3	0.3
0.2	0.1	0.4

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Question: is $\mathbb{P} = \mathbb{Q}$?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?
- Do Brits have the same friend network types as Americans?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?
- Do Brits have the same friend network types as Americans?
- When does my laser agree with the one on Mars?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?
- Do Brits have the same friend network types as Americans?
- When does my laser agree with the one on Mars?
- Are storms in the 2000s different from storms in the 1800s?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?
- Do Brits have the same friend network types as Americans?
- When does my laser agree with the one on Mars?
- Are storms in the 2000s different from storms in the 1800s?
- Does presence of this protein affect DNA binding? [MMDiff2]

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?
- Do Brits have the same friend network types as Americans?
- When does my laser agree with the one on Mars?
- Are storms in the 2000s different from storms in the 1800s?
- Does presence of this protein affect DNA binding? [MMDiff2]
- Do these `dob` and `birthday` columns mean the same thing?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?
- Do Brits have the same friend network types as Americans?
- When does my laser agree with the one on Mars?
- Are storms in the 2000s different from storms in the 1800s?
- Does presence of this protein affect DNA binding? [MMDiff2]
- Do these `dob` and `birthday` columns mean the same thing?
- Does my generative model $\mathbb{Q}_\theta$ match $\mathbb{P}_{\mathbf{data}}$?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Do smokers/non-smokers get different cancers?
- Do Brits have the same friend network types as Americans?
- When does my laser agree with the one on Mars?
- Are storms in the 2000s different from storms in the 1800s?
- Does presence of this protein affect DNA binding? [MMDiff2]
- Do these `dob` and `birthday` columns mean the same thing?
- Does my generative model $\mathbb{Q}_\theta$ match $\mathbb{P}_{\mathbf{data}}$?
- Independence testing: is $P(X, Y) = P(X)P(Y)$?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Question: is $\mathbb{P} = \mathbb{Q}$?

I: Two-sample testing

- Given samples from two unknown distributions

$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Question: is $\mathbb{P} = \mathbb{Q}$?
- Hypothesis testing approach:

$$H_0 : \mathbb{P} = \mathbb{Q} \qquad H_1 : \mathbb{P} \neq \mathbb{Q}$$

I: Two-sample testing

- Given samples from two unknown distributions

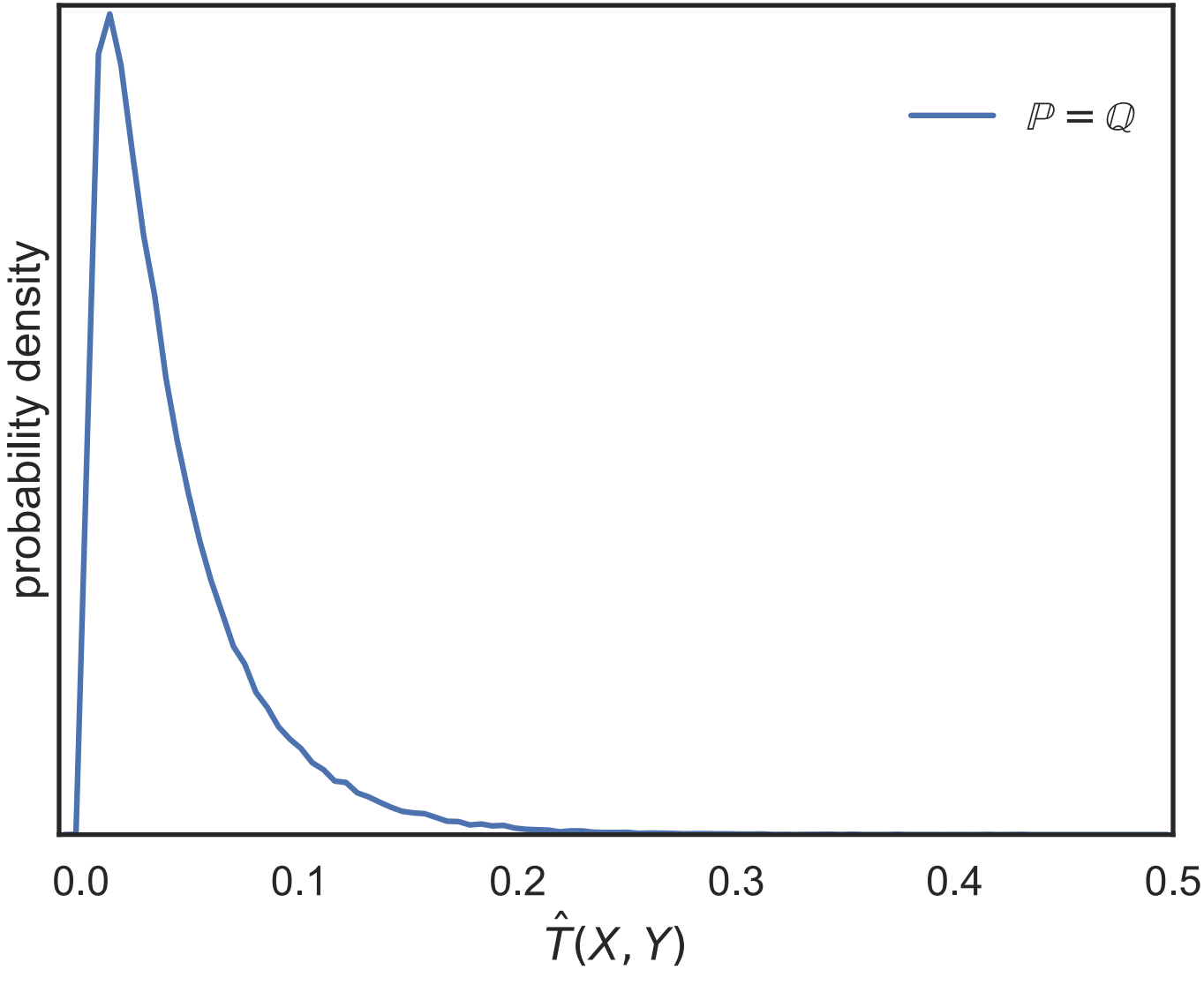
$$X \sim \mathbb{P} \qquad Y \sim \mathbb{Q}$$

- Question: is $\mathbb{P} = \mathbb{Q}$?
- Hypothesis testing approach:

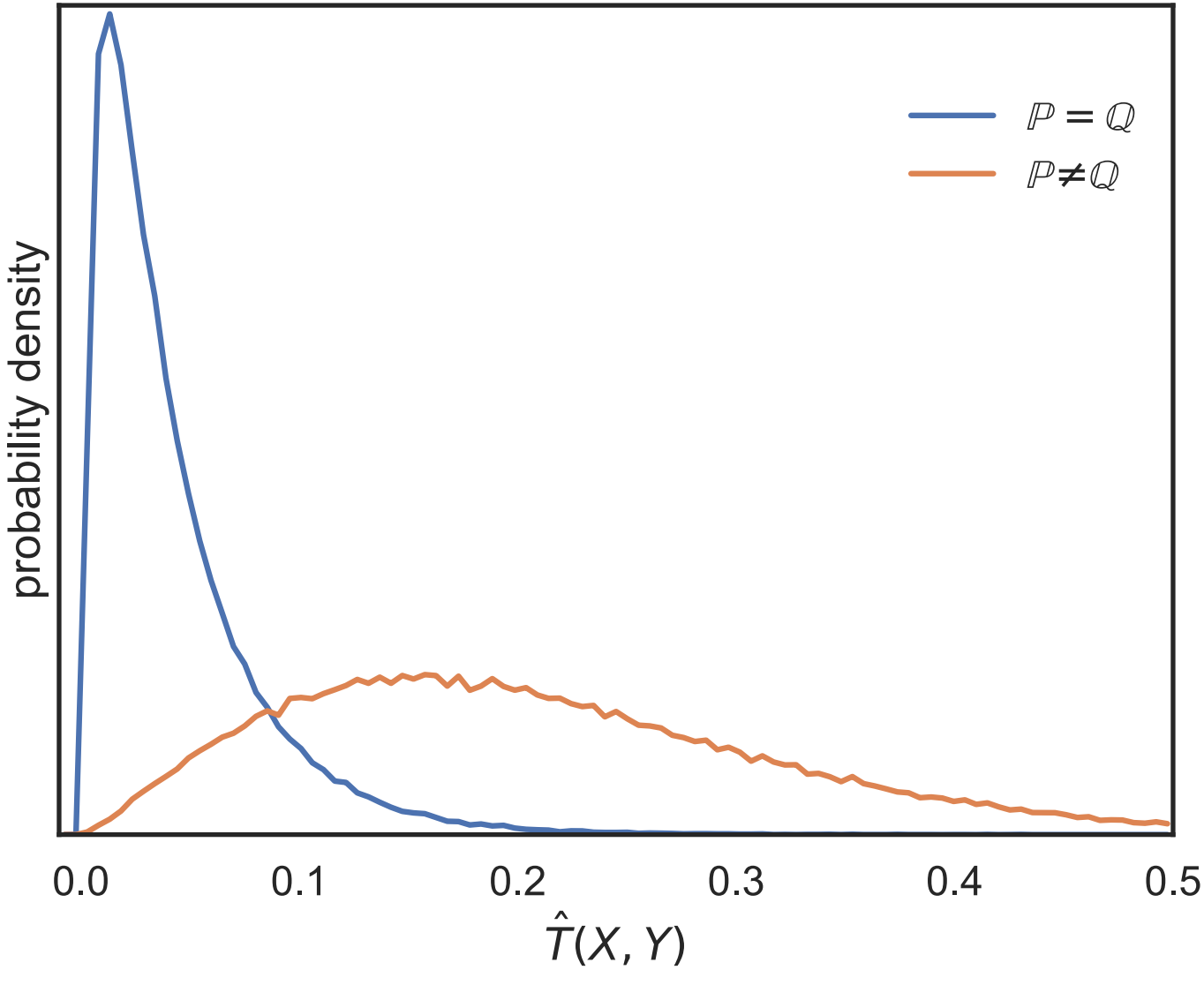
$$H_0 : \mathbb{P} = \mathbb{Q} \qquad H_1 : \mathbb{P} \neq \mathbb{Q}$$

- Reject H_0 if test statistic $\hat{T}(X, Y) > c_\alpha$

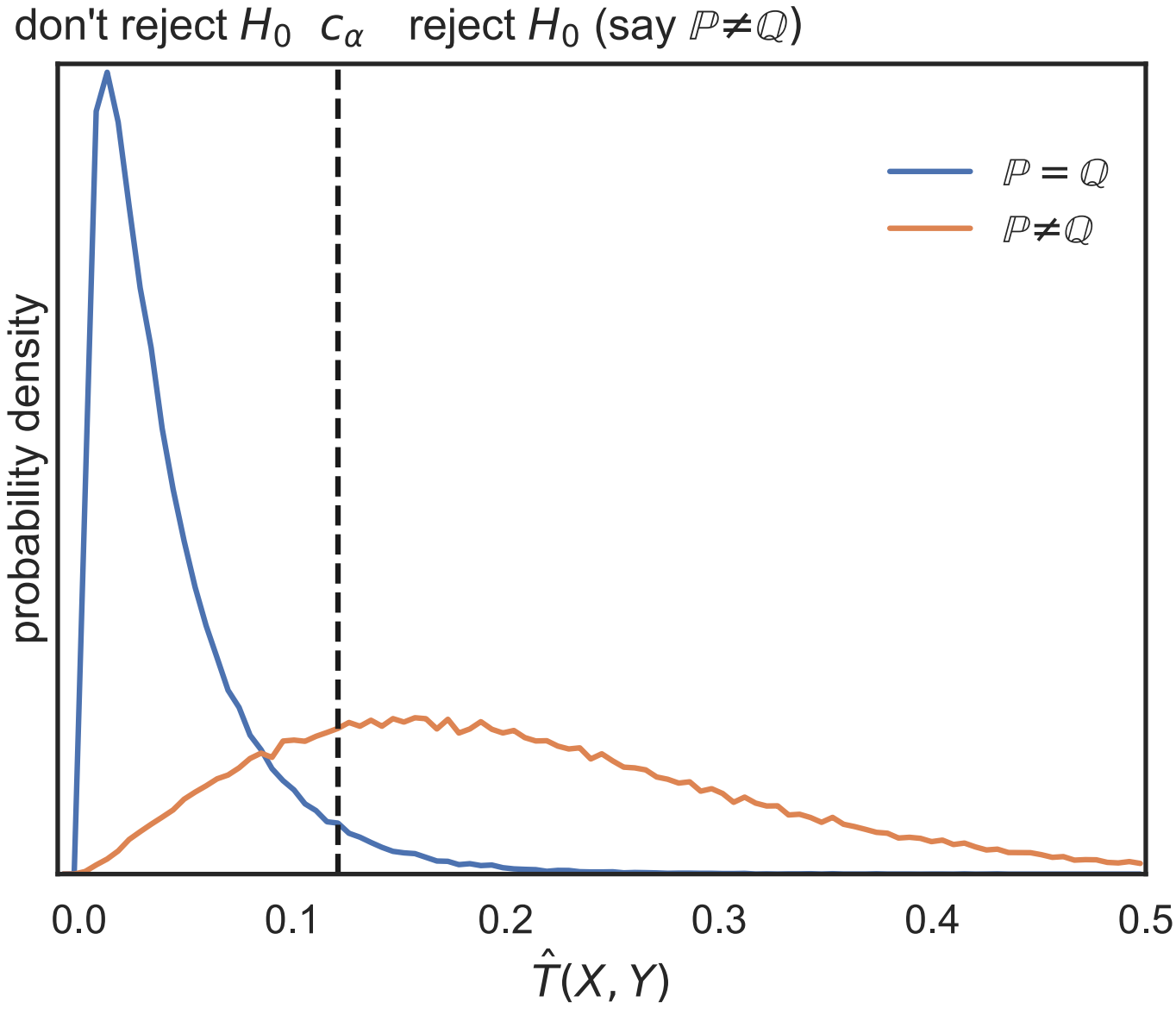
What's a hypothesis test again?



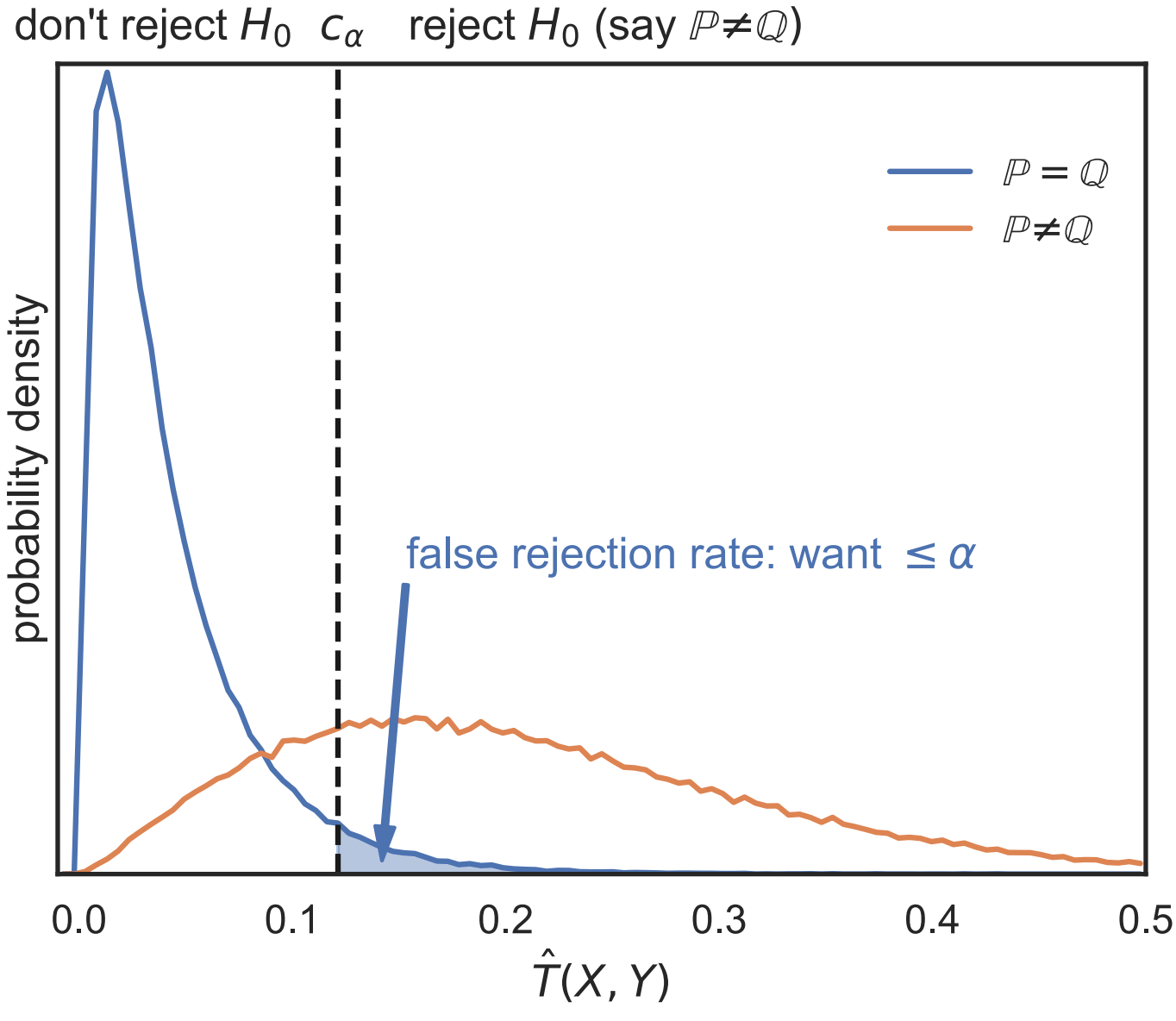
What's a hypothesis test again?



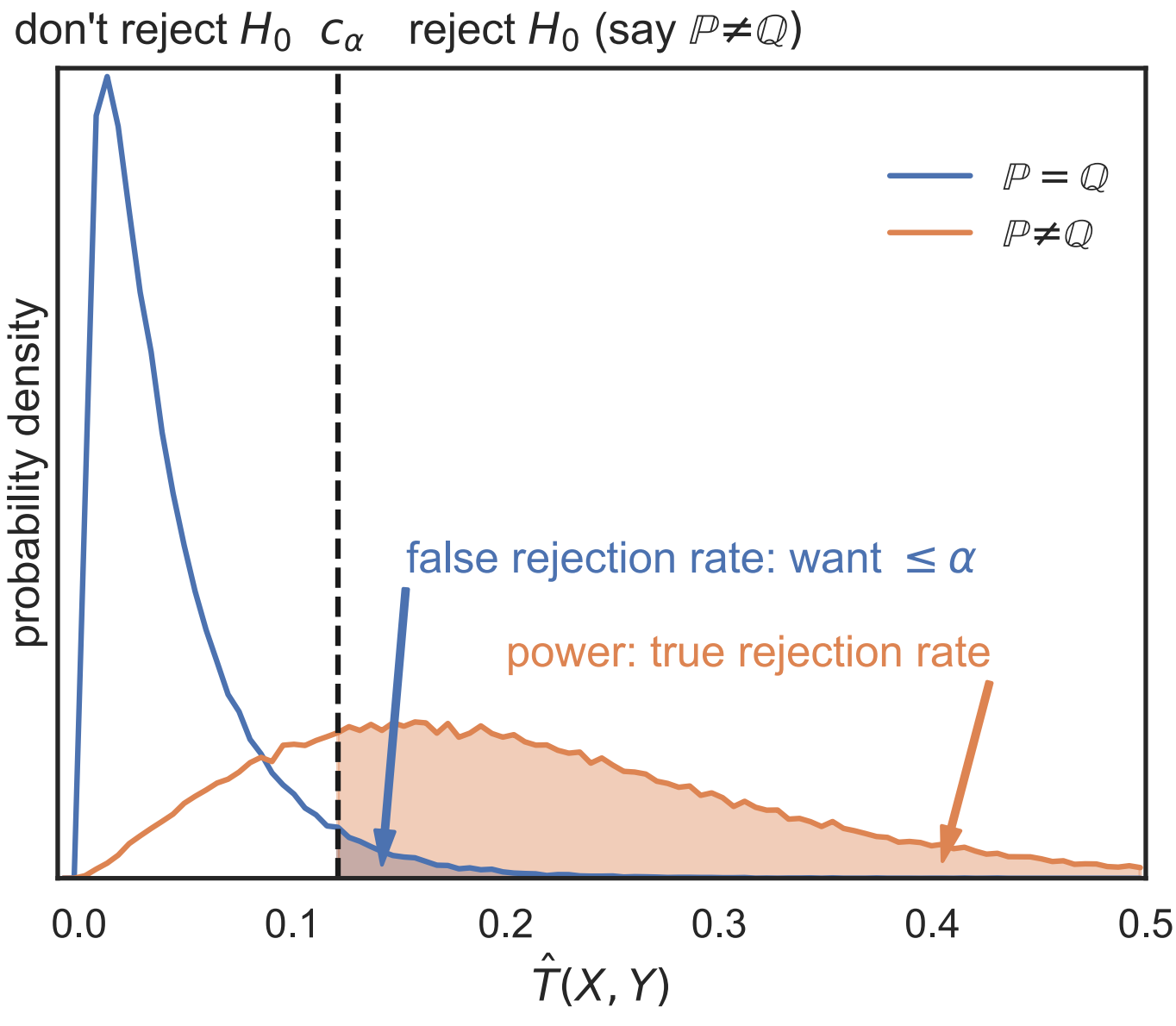
What's a hypothesis test again?



What's a hypothesis test again?



What's a hypothesis test again?



Permutation testing to find c_α

Need $\mathbf{Pr}_{H_0}\left(T(X, Y) > c_\alpha\right) \leq \alpha$

$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \qquad Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5$

c_α: $1 - \alpha$th quantile of $\left\{ \qquad \right\}$

Permutation testing to find c_α

Need $\Pr_{H_0}\left(T(X, Y) > c_\alpha\right) \leq \alpha$

c_α: $1-\alpha$th quantile of $\Big\{$ $\Big\}$

Permutation testing to find c_α

Need $\Pr_{H_0}\left(T(X, Y) > c_\alpha\right) \leq \alpha$

c_α: $1-\alpha$th quantile of $\left\{\hat{T}(\tilde{X}_1, \tilde{Y}_1),\right.$ $\left.\right\}$

Permutation testing to find c_α

Need $\Pr_{H_0}\left(T(X, Y) > c_\alpha\right) \leq \alpha$

c_α: $1-\alpha$th quantile of $\left\{\hat{T}(\tilde{X}_1, \tilde{Y}_1),\ \hat{T}(\tilde{X}_2, \tilde{Y}_2),\ \right\}$

Permutation testing to find c_α

Need $\Pr_{H_0}\left(T(X, Y) > c_\alpha\right) \leq \alpha$

$$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \qquad Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad Y_5$$

c_α: $1-\alpha$th quantile of $\left\{\hat{T}(\tilde{X}_1, \tilde{Y}_1),\ \hat{T}(\tilde{X}_2, \tilde{Y}_2),\ \cdots\right\}$

MMD-based tests

- If k is *characteristic*, $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$
- Efficient permutation testing for $\widehat{\mathrm{MMD}}(X, Y)$

MMD-based tests

- If k is *characteristic*, $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$
- Efficient permutation testing for $\widehat{\mathrm{MMD}}(X, Y)$
  - H_0: $n\widehat{\mathrm{MMD}}^2$ converges in distribution
  - H_1: $\sqrt{n}(\widehat{\mathrm{MMD}}^2 - \mathrm{MMD}^2)$ asymptotically normal

MMD-based tests

- If k is *characteristic*, $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$
- Efficient permutation testing for $\widehat{\mathrm{MMD}}(X, Y)$
  - H_0: $n\widehat{\mathrm{MMD}}^2$ converges in distribution
  - H_1: $\sqrt{n}(\widehat{\mathrm{MMD}}^2 - \mathrm{MMD}^2)$ asymptotically normal
- Any characteristic kernel gives consistent test

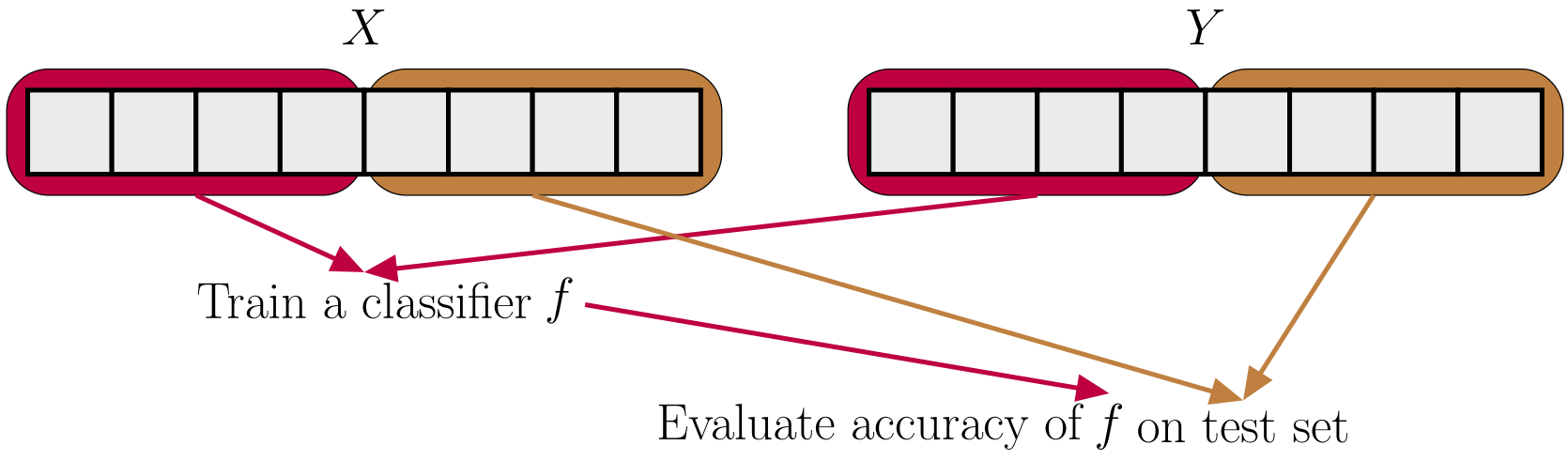
MMD-based tests

- If k is *characteristic*, $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$
- Efficient permutation testing for $\widehat{\mathrm{MMD}}(X, Y)$
  - H_0: $n\widehat{\mathrm{MMD}}^2$ converges in distribution
  - H_1: $\sqrt{n}(\widehat{\mathrm{MMD}}^2 - \mathrm{MMD}^2)$ asymptotically normal
- Any characteristic kernel gives consistent test…eventually

MMD-based tests

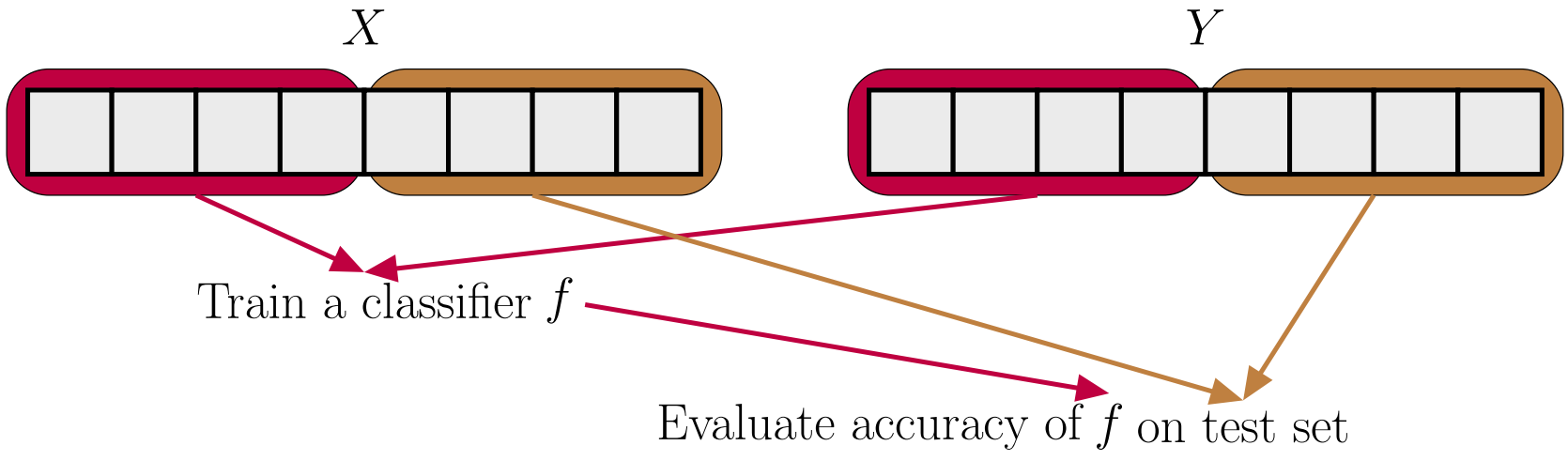
- If k is *characteristic*, $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$
- Efficient permutation testing for $\widehat{\mathrm{MMD}}(X, Y)$
  - H_0: $n\widehat{\mathrm{MMD}}^2$ converges in distribution
  - H_1: $\sqrt{n}(\widehat{\mathrm{MMD}}^2 - \mathrm{MMD}^2)$ asymptotically normal
- Any characteristic kernel gives consistent test…eventually
- Need enormous n if kernel is bad for problem

Classifier two-sample tests



- $\hat{T}(X, Y)$ is the accuracy of f on the test set
- Under H_0, classification impossible: $\hat{T} \sim \mathbf{Binomial}(n, \frac{1}{2})$

Classifier two-sample tests



- $\hat{T}(X, Y)$ is the accuracy of f on the test set
- Under H_0, classification impossible: $\hat{T} \sim \mathbf{Binomial}(n, \frac{1}{2})$
- With $k(x, y) = \frac{1}{4} f(x) f(y)$ where $f(x) \in \{-1, 1\}$, get $\widehat{\mathrm{MMD}}(X, Y) = \left| \hat{T}(X, Y) - \frac{1}{2} \right|$

Optimizing test power

- Asymptotics of $\widehat{\mathrm{MMD}}^2$ give us immediately that

$$\Pr_{H_1}\left(n\widehat{\mathrm{MMD}}^2 > c_\alpha\right) \approx \Phi\left(\frac{\sqrt{n}\,\mathrm{MMD}^2}{\sigma_{H_1}} - \frac{c_\alpha}{\sqrt{n}\sigma_{H_1}}\right)$$

MMD, σ_{H_1}, c_α are constants: first term dominates

Optimizing test power

- Asymptotics of $\widehat{\mathrm{MMD}}^2$ give us immediately that

$$\Pr_{H_1}\left(n\widehat{\mathrm{MMD}}^2 > c_\alpha\right) \approx \Phi\left(\frac{\sqrt{n}\,\mathrm{MMD}^2}{\sigma_{H_1}} - \frac{c_\alpha}{\sqrt{n}\sigma_{H_1}}\right)$$

  MMD, σ_{H_1}, c_α are constants: first term dominates

- Pick k to maximize an estimate of $\mathrm{MMD}^2 \,/\sigma_{H_1}$

Optimizing test power

- Asymptotics of $\widehat{\mathrm{MMD}}^2$ give us immediately that

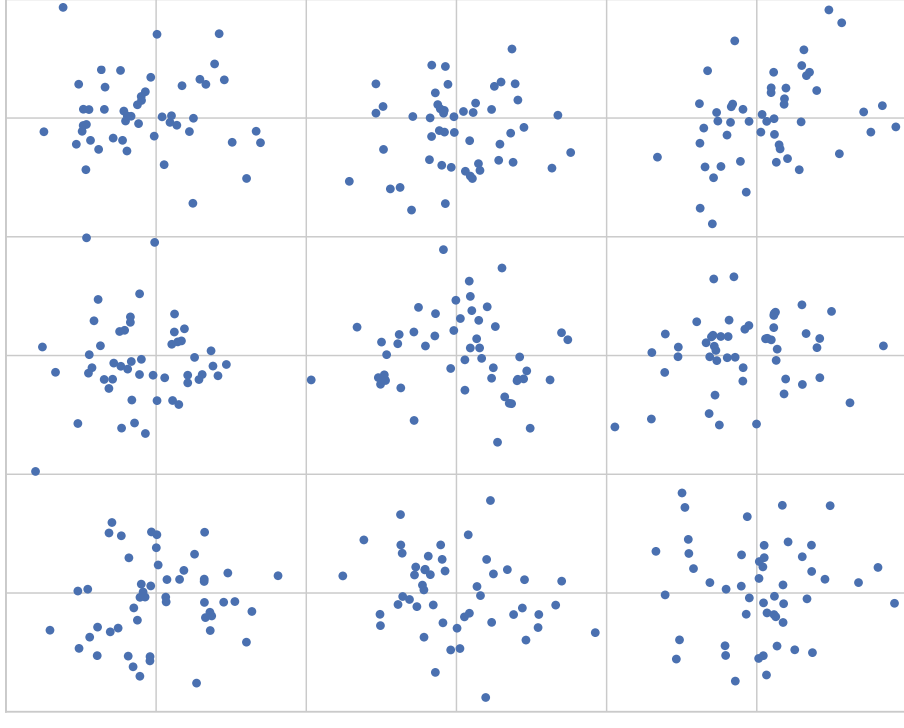
$$\Pr_{H_1}\left(n\widehat{\mathrm{MMD}}^2 > c_\alpha\right) \approx \Phi\left(\frac{\sqrt{n}\,\mathrm{MMD}^2}{\sigma_{H_1}} - \frac{c_\alpha}{\sqrt{n}\sigma_{H_1}}\right)$$

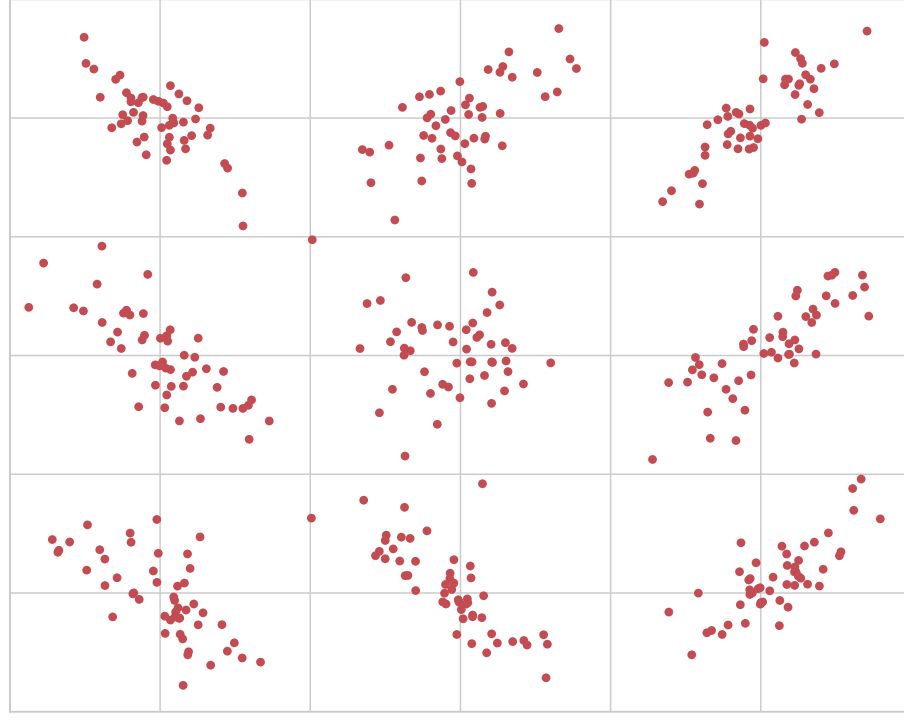
  MMD, σ_{H_1}, c_α are constants: first term dominates

- Pick k to maximize an estimate of $\mathrm{MMD}^2 \,/\sigma_{H_1}$

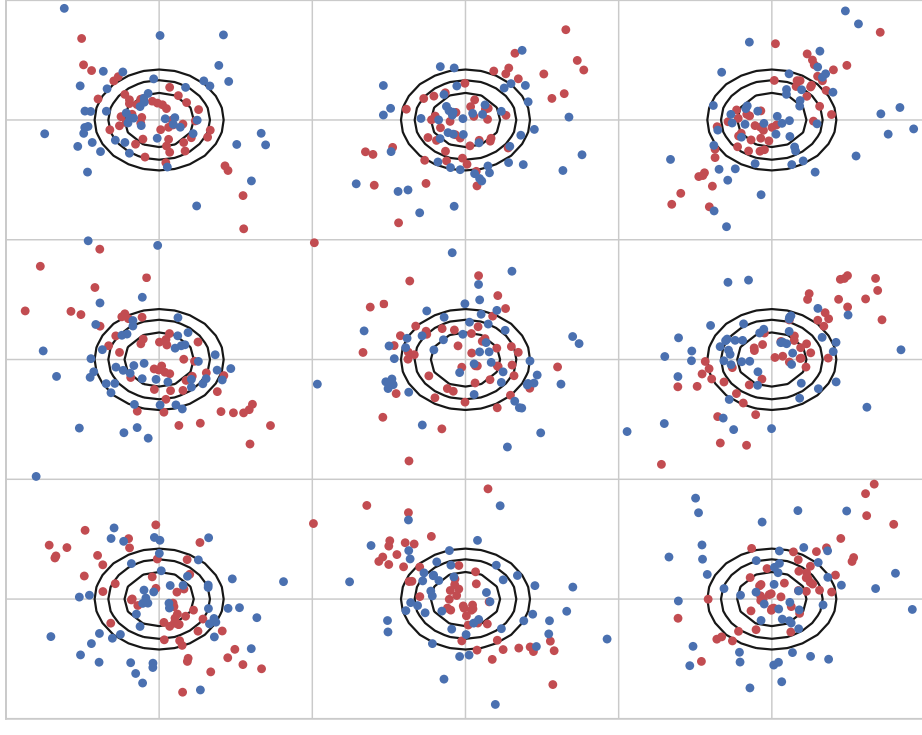
- Can show uniform $\mathcal{O}_P\left(n^{-\frac{1}{3}}\right)$ convergence of estimator

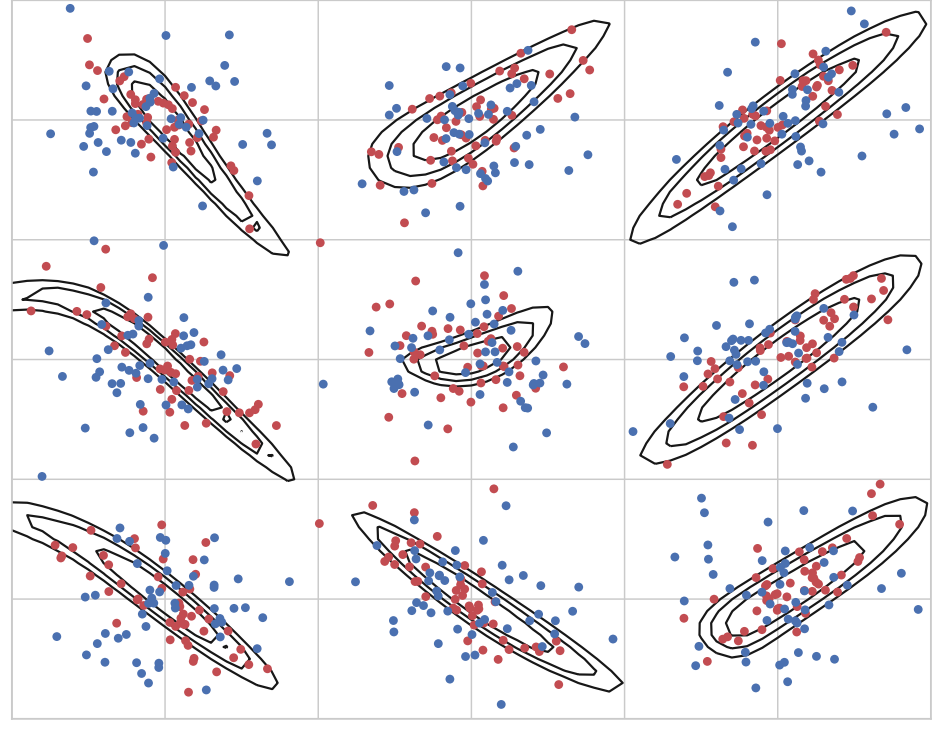
Blobs dataset



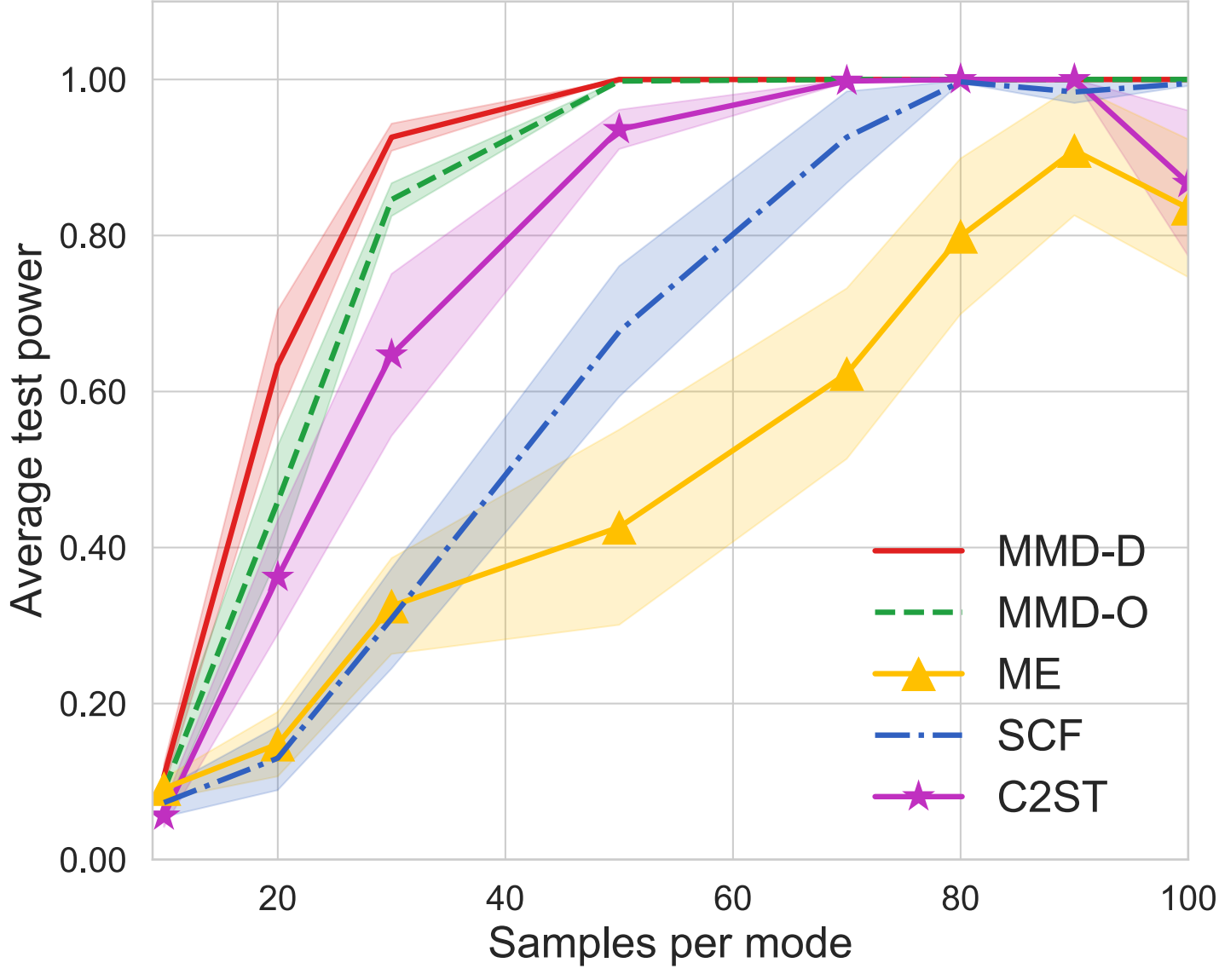


Blobs kernels

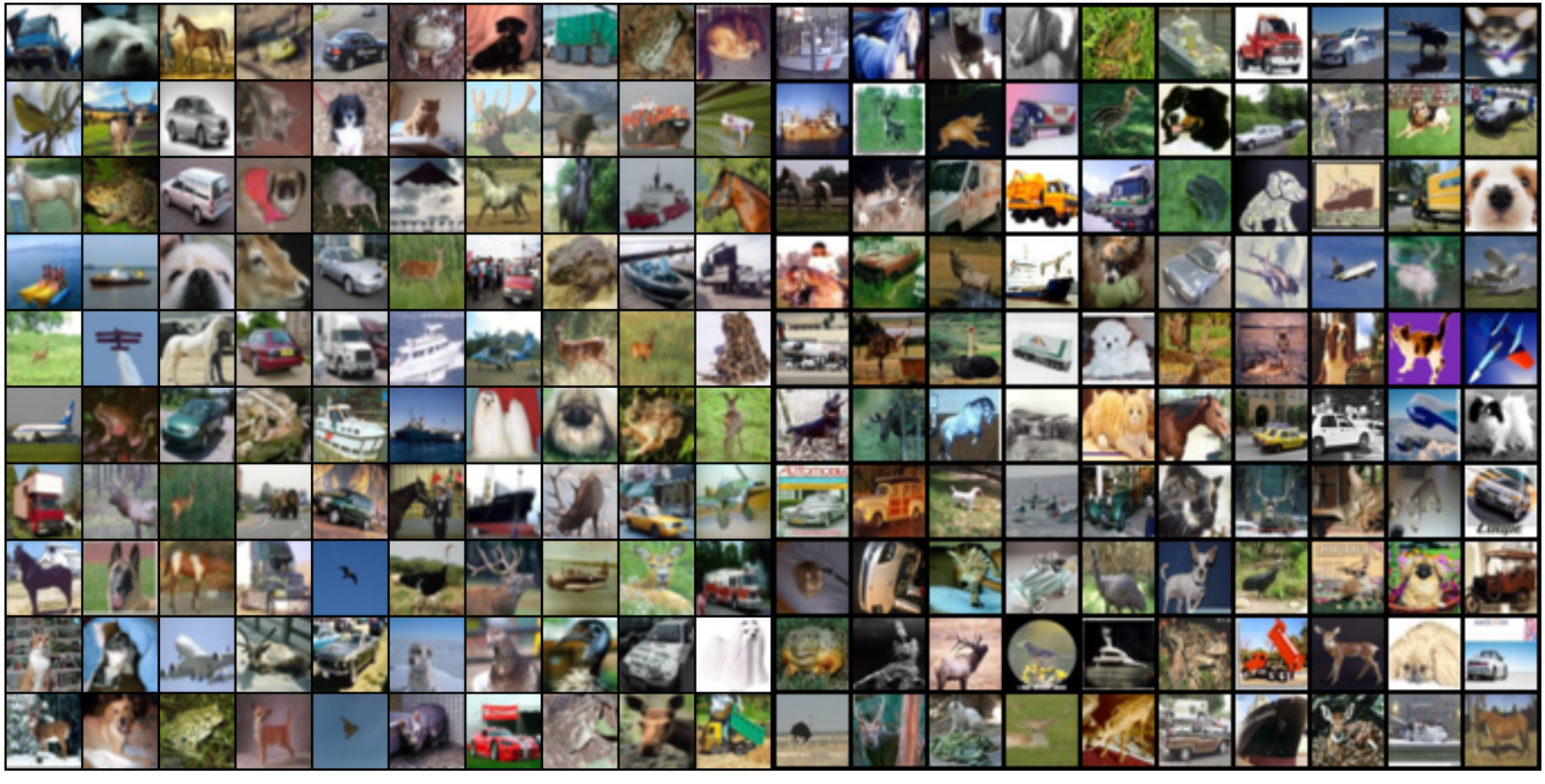




Blobs results



CIFAR-10 vs CIFAR-10.1



Train on 1 000, test on 1 031, repeat 10 times. Rejection rates:

ME	SCF	C2ST	MMD-O	MMD-D
0.588	0.171	0.452	0.316	**0.744**

Ablation vs classifier-based tests

	Cross-entropy			Max power		
Dataset	**Sign**	**Lin**	**Ours**	**Sign**	**Lin**	**Ours**
Blobs	0.84	0.94	0.90	–	0.95	0.99
High-d Gauss. mix.	0.47	0.59	0.29	–	0.64	0.66
Higgs	0.26	0.40	0.35	–	0.30	0.40
MNIST vs GAN	0.65	0.71	0.80	–	0.94	1.00

But...

- What if you don't have much data for your testing problem?

But...

- What if you don't have much data for your testing problem?
- Need enough data to pick a good kernel

But...

- What if you don't have much data for your testing problem?
- Need enough data to pick a good kernel
- Also need enough test data to actually detect the difference

But...

- What if you don't have much data for your testing problem?
- Need enough data to pick a good kernel
- Also need enough test data to actually detect the difference
- Best split depends on best kernel's quality / how hard to find

But...

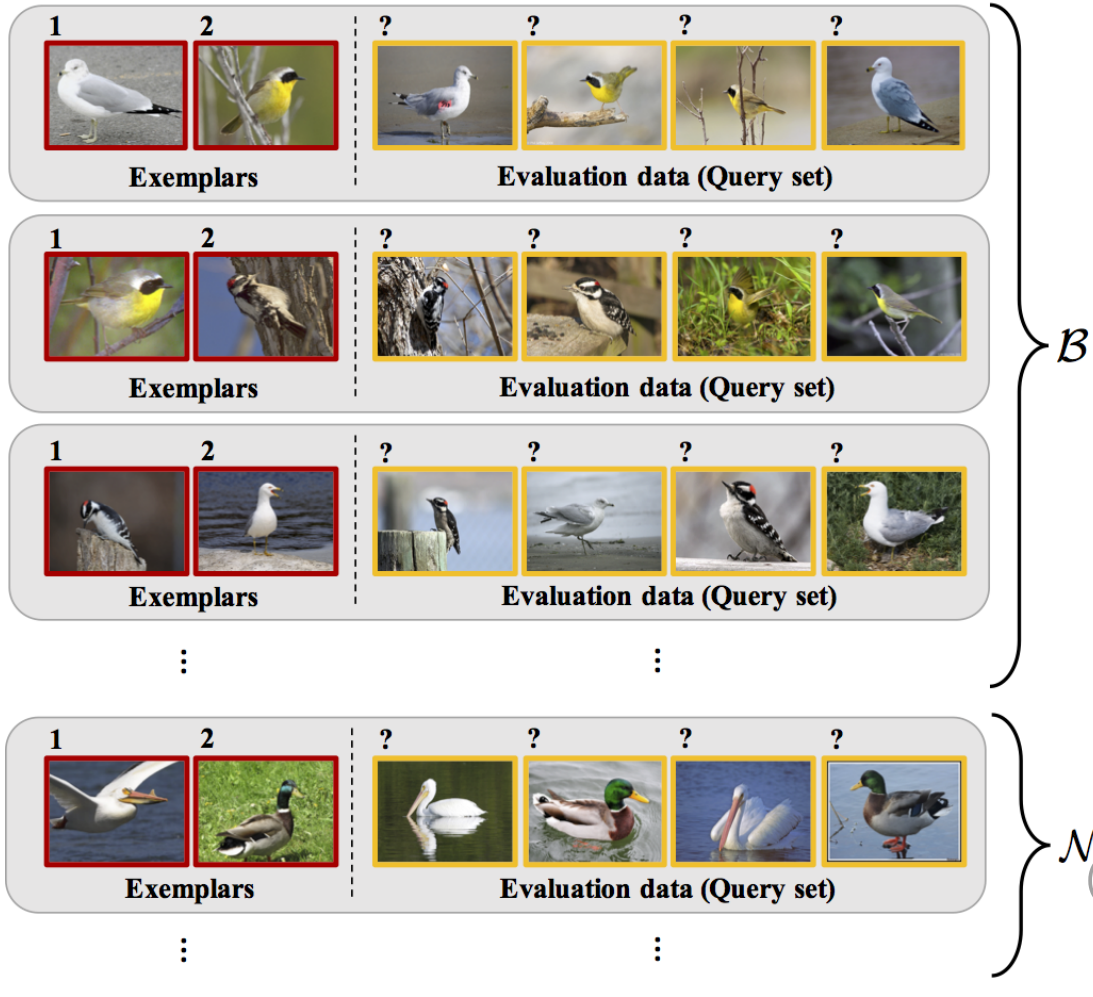
- What if you don't have much data for your testing problem?
- Need enough data to pick a good kernel
- Also need enough test data to actually detect the difference
- Best split depends on best kernel's quality / how hard to find
  - Don't know that ahead of time; can't try more than one

Meta-testing

- One idea: what if we have *related* problems?

Meta-testing

- One idea: what if we have *related* problems?
- Similar setup to meta-learning:



(from Wei+ 2018)

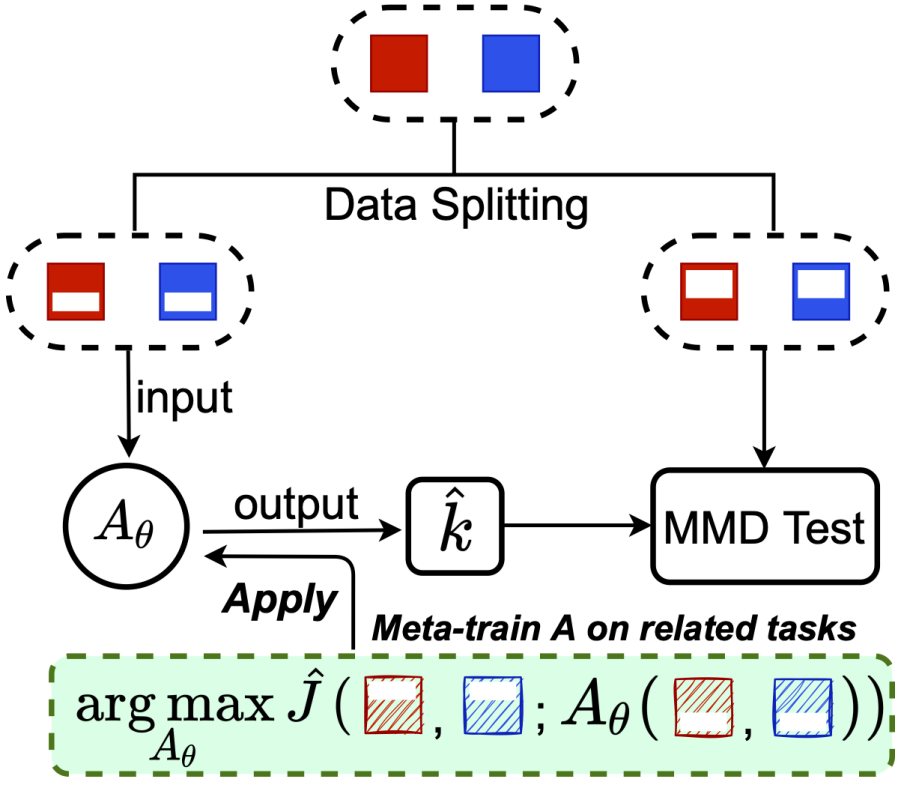
Meta-testing for CIFAR-10 vs CIFAR-10.1

- CIFAR-10 has 60,000 images, but CIFAR-10.1 only has 2,031
- Where do we get related data from?

Meta-testing for CIFAR-10 vs CIFAR-10.1

- CIFAR-10 has 60,000 images, but CIFAR-10.1 only has 2,031
- Where do we get related data from?
- One option: set up tasks to distinguish classes of CIFAR-10 (airplane vs automobile, airplane vs bird, ...)

One approach (MAML-like)

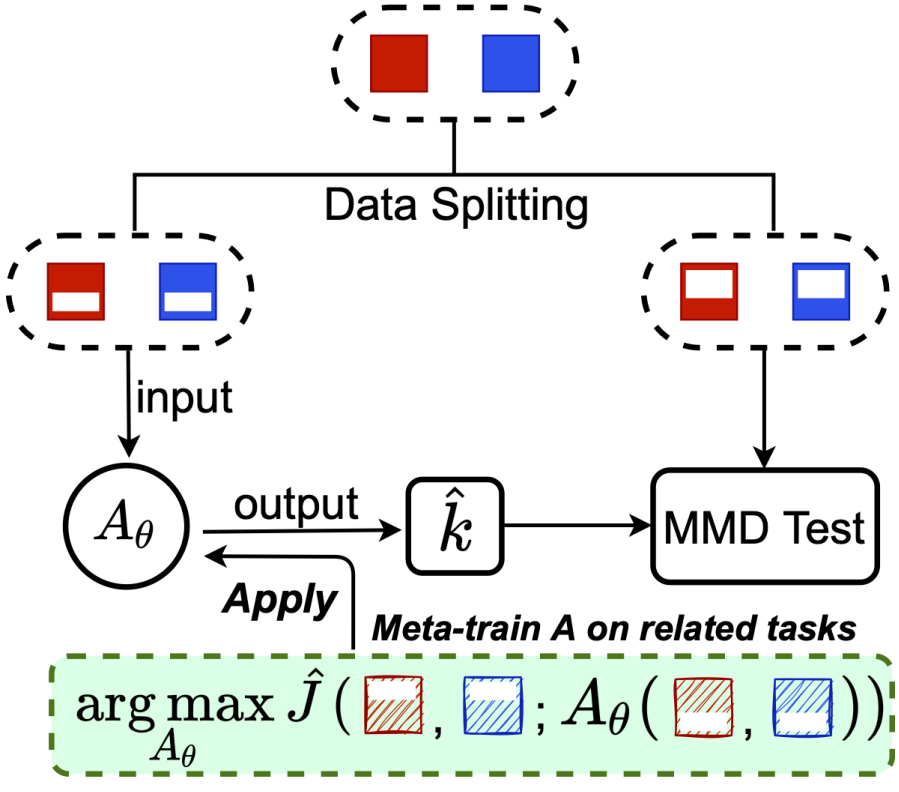


A_θ is, e.g., 5 steps of gradient descent

we learn the initialization, maybe step size, etc

Samples from $\mathbb{P}$ Samples from $\mathbb{Q}$ Training Samples Testing Samples Meta-Samples

One approach (MAML-like)



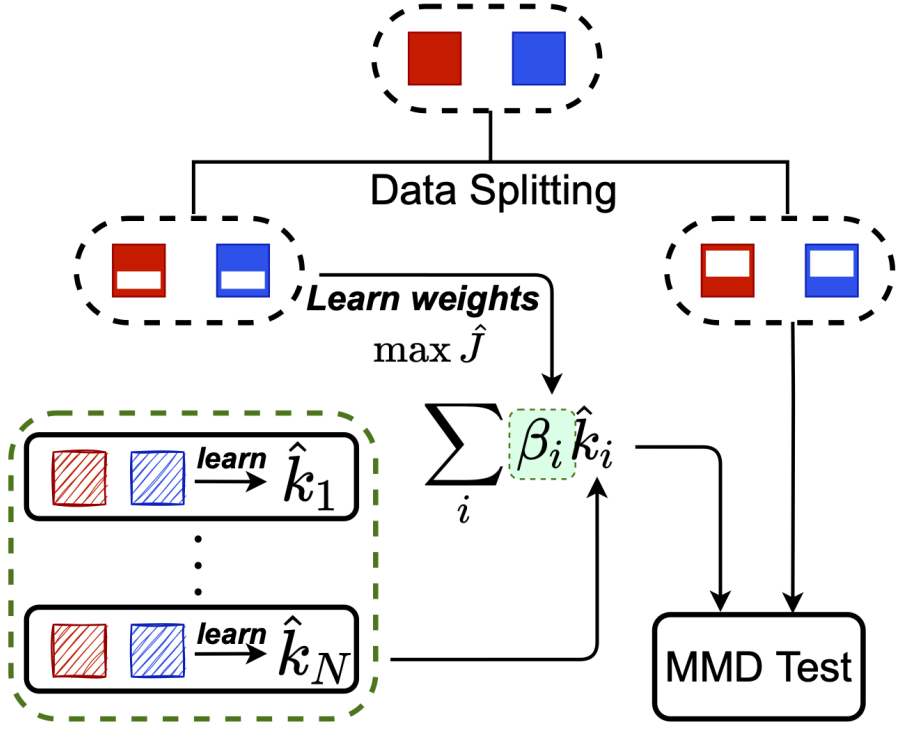
A_θ is, e.g., 5 steps of gradient descent

we learn the initialization, maybe step size, etc

Samples from $\mathbb{P}$ Samples from $\mathbb{Q}$ Training Samples Testing Samples Meta-Samples

This works, but not as well as we'd hoped...
Initialization might work okay on everything, not really adapt

Another approach: Meta-MKL



Inspired by classic multiple kernel learning

Only need to learn linear combination β_i on test task: much easier

Samples from $\mathbb{P}$ Samples from $\mathbb{Q}$ Training Samples Testing Samples Meta-Samples

Theoretical analysis for Meta-MKL

- Same big-O dependence on test task size 😐
- But multiplier is *much* better:
  based on number of meta-training tasks, not on network size

Theoretical analysis for Meta-MKL

- Same big-O dependence on test task size 😐
- But multiplier is *much* better:
  based on number of meta-training tasks, not on network size
- (Analysis assumes meta-tasks are “related” enough)

Results on CIFAR-10.1

Methods	$m_{tr} = 100$			$m_{tr} = 200$		
	$m_{te} = 200$	$m_{te} = 500$	$m_{te} = 900$	$m_{te} = 200$	$m_{te} = 500$	$m_{te} = 900$
ME	$0.084_{\pm 0.009}$	$0.096_{\pm 0.016}$	$0.160_{\pm 0.035}$	$0.104_{\pm 0.013}$	$0.202_{\pm 0.020}$	$0.326_{\pm 0.039}$
SCF	$0.047_{\pm 0.013}$	$0.037_{\pm 0.011}$	$0.047_{\pm 0.015}$	$0.026_{\pm 0.009}$	$0.018_{\pm 0.006}$	$0.026_{\pm 0.012}$
C2ST-S	$0.059_{\pm 0.009}$	$0.062_{\pm 0.007}$	$0.059_{\pm 0.007}$	$0.052_{\pm 0.011}$	$0.054_{\pm 0.011}$	$0.057_{\pm 0.008}$
C2ST-L	$0.064_{\pm 0.009}$	$0.064_{\pm 0.006}$	$0.063_{\pm 0.007}$	$0.075_{\pm 0.014}$	$0.066_{\pm 0.011}$	$0.067_{\pm 0.008}$
MMD-O	$0.091_{\pm 0.011}$	$0.141_{\pm 0.009}$	$0.279_{\pm 0.018}$	$0.084_{\pm 0.007}$	$0.160_{\pm 0.011}$	$0.319_{\pm 0.020}$
MMD-D	$0.104_{\pm 0.007}$	$0.222_{\pm 0.020}$	$0.418_{\pm 0.046}$	$0.117_{\pm 0.013}$	$0.226_{\pm 0.021}$	$0.444_{\pm 0.037}$
AGT-KL	$0.170_{\pm 0.032}$	$0.457_{\pm 0.052}$	$0.765_{\pm 0.045}$	$0.152_{\pm 0.023}$	$0.463_{\pm 0.060}$	$0.778_{\pm 0.050}$
Meta-KL	$0.245_{\pm 0.010}$	$0.671_{\pm 0.026}$	$0.959_{\pm 0.013}$	$0.226_{\pm 0.015}$	$0.668_{\pm 0.032}$	$0.972_{\pm 0.006}$
Meta-MKL	$\mathbf{0.277}_{\pm \mathbf{0.016}}$	$\mathbf{0.728}_{\pm \mathbf{0.020}}$	$\mathbf{0.973}_{\pm \mathbf{0.008}}$	$\mathbf{0.255}_{\pm \mathbf{0.020}}$	$\mathbf{0.724}_{\pm \mathbf{0.026}}$	$\mathbf{0.993}_{\pm \mathbf{0.003}}$

Challenges for testing

- When $\mathbb{P} \neq \mathbb{Q}$, can we tell *how* they're different?

Challenges for testing

- When $\mathbb{P} \neq \mathbb{Q}$, can we tell *how* they're different?
  - Methods so far: some mostly for low-d
  - Some look at points with large critic function

Challenges for testing

- When $\mathbb{P} \neq \mathbb{Q}$, can we tell *how* they're different?
  - Methods so far: some mostly for low-d
  - Some look at points with large critic function

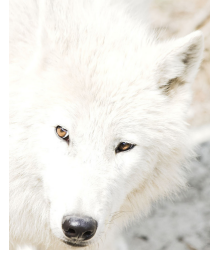




Challenges for testing

- When $\mathbb{P} \neq \mathbb{Q}$, can we tell *how* they're different?
  - Methods so far: some mostly for low-d
  - Some look at points with large critic function




- Finding kernels / features that *can't* do certain things
  - distinguish by emotion, but can't distinguish by skin color

Challenges for testing

- When $\mathbb{P} \neq \mathbb{Q}$, can we tell *how* they're different?
  - Methods so far: some mostly for low-d
  - Some look at points with large critic function

- Finding kernels / features that *can't* do certain things
  - distinguish by emotion, but can't distinguish by skin color
- Avoid the need for data splitting (selective inference)

Challenges for testing

- When $\mathbb{P} \neq \mathbb{Q}$, can we tell *how* they're different?
  - Methods so far: some mostly for low-d
  - Some look at points with large critic function

- Finding kernels / features that *can't* do certain things
  - distinguish by emotion, but can't distinguish by skin color
- Avoid the need for data splitting (selective inference)
  - Kübler+ NeurIPS-20 gave one method
  - only for multiple kernel learning
  - only with data-inefficient (streaming) estimator

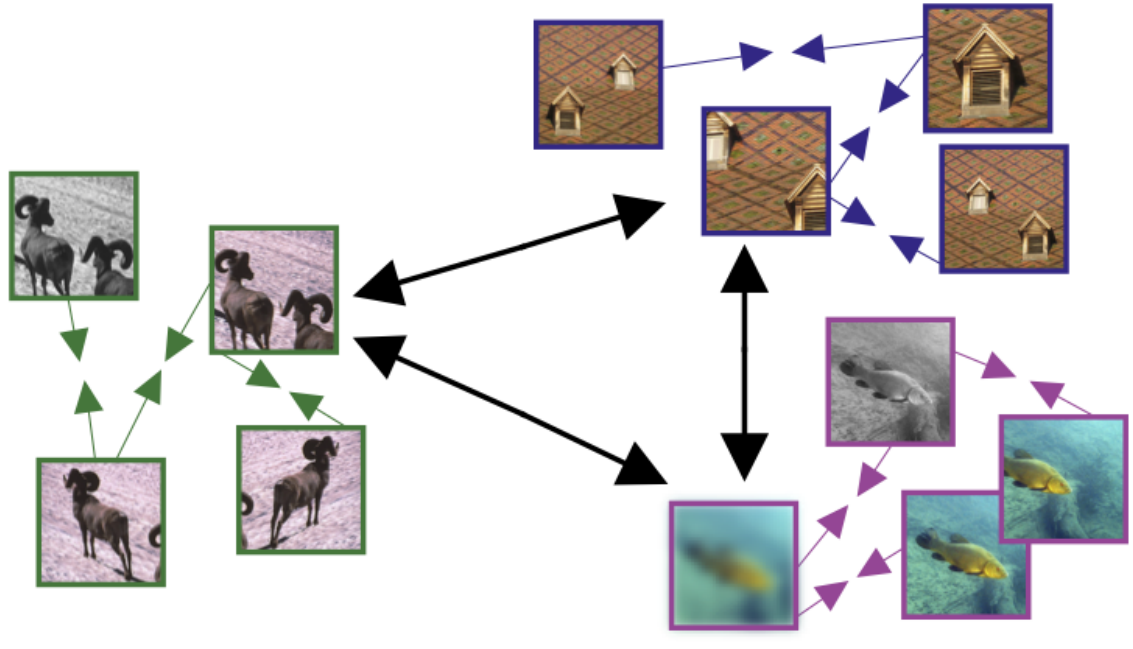
II: Self-supervised learning

Given a bunch of unlabeled samples X,
want to find "good" features $Z = f(X)$
(e.g. so that a linear classifier on Z works with few samples)

II: Self-supervised learning

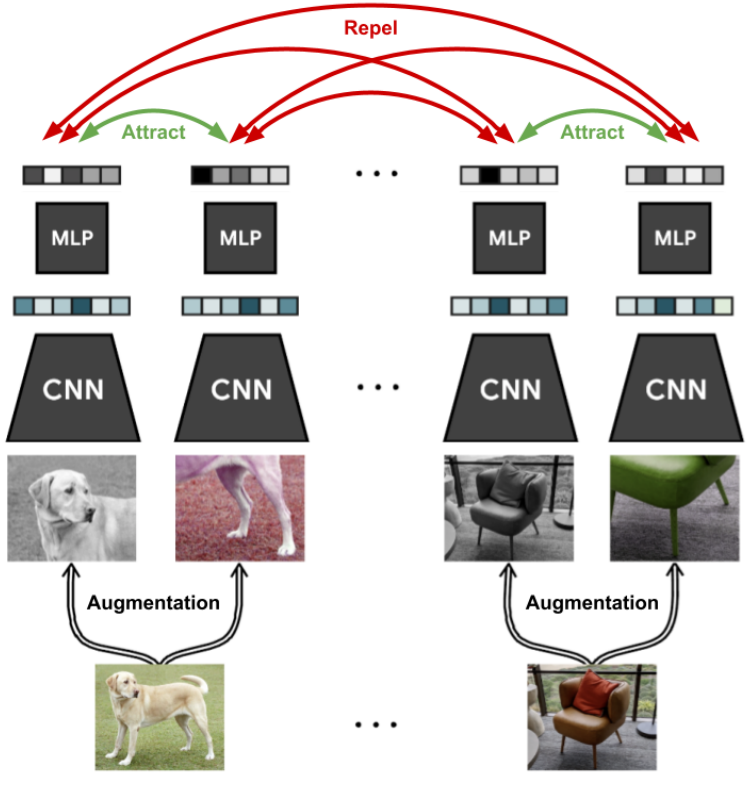
Given a bunch of unlabeled samples X,
want to find "good" features $Z = f(X)$
(e.g. so that a linear classifier on Z works with few samples)

One common approach: contrastive learning



InfoNCE [van den Oord+ 2018, Poole+ ICML-19]

$$\mathop{\mathbb{E}}_{Z_1}\left[\mathop{\mathbb{E}}_{Z_2\sim \text{pos}}[k(Z_1, Z_2)]] + \log \mathop{\mathbb{E}}_{Z_2}[\exp(k(Z_1, Z_2))\right] \leq \text{MI}(Z_1, Z_2)$$

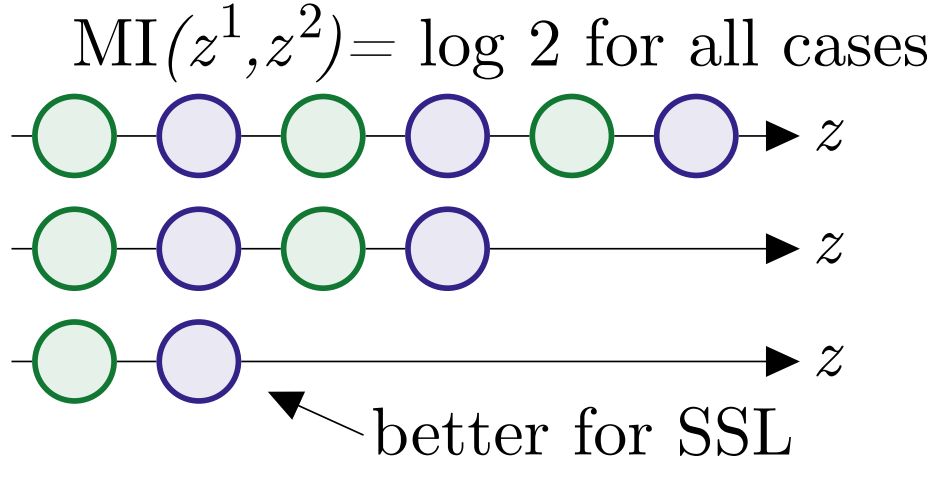


Variants:
CPC, SimCLR, MoCo, SwAV, ...

Mutual information isn't why SSL works!

InfoNCE approximates MI between "positive" views

But MI is invariant to transformations important to SSL!



Hilbert-Schmidt Independence Criterion (HSIC)

$$\begin{aligned}\mathrm{HSIC}(X, Y) &= \|\mathbb{E}[\phi(X) \otimes \phi(Y)] - \mathbb{E}[\phi(X)] \otimes \mathbb{E}[\phi(Y)]\|^2_{HS} \\ &= \mathrm{MMD}^2(\mathbb{P}_{XY}, \mathbb{P}_X \otimes \mathbb{P}_Y)\end{aligned}$$

Hilbert-Schmidt Independence Criterion (HSIC)

$$
\begin{aligned}
\mathrm{HSIC}(X, Y) &= \left\| \mathbb{E}[\phi(X) \otimes \phi(Y)] - \mathbb{E}[\phi(X)] \otimes \mathbb{E}[\phi(Y)] \right\|_{HS}^2 \\
&= \mathrm{MMD}^2\left(\mathbb{P}_{XY}, \mathbb{P}_X \otimes \mathbb{P}_Y\right) \\
&\leq C_k \, \mathrm{MI}(X, Y) \qquad (C_k \text{ depends only on } \|k\|_\infty)
\end{aligned}
$$

Hilbert-Schmidt Independence Criterion (HSIC)

$$\begin{aligned}\mathrm{HSIC}(X, Y) &= \left\|\mathbb{E}[\phi(X) \otimes \phi(Y)] - \mathbb{E}[\phi(X)] \otimes \mathbb{E}[\phi(Y)]\right\|_{HS}^2 \\ &= \mathrm{MMD}^2\left(\mathbb{P}_{XY}, \mathbb{P}_X \otimes \mathbb{P}_Y\right) \\ &\leq C_k \, \mathrm{MI}(X, Y) \qquad (C_k \text{ depends only on } \|k\|_\infty)\end{aligned}$$

With a linear kernel: $\mathrm{HSIC} = \left\|\mathbb{E}[XY^\mathsf{T}] - \mathbb{E}[X]\,\mathbb{E}[Y]^\mathsf{T}\right\|_F^2$

Hilbert-Schmidt Independence Criterion (HSIC)

$$\begin{aligned}\mathrm{HSIC}(X, Y) &= \left\|\mathbb{E}[\phi(X) \otimes \phi(Y)] - \mathbb{E}[\phi(X)] \otimes \mathbb{E}[\phi(Y)]\right\|_{HS}^2 \\ &= \mathrm{MMD}^2\left(\mathbb{P}_{XY}, \mathbb{P}_X \otimes \mathbb{P}_Y\right) \\ &\leq C_k \, \mathrm{MI}(X, Y) \qquad (C_k \text{ depends only on } \|k\|_\infty)\end{aligned}$$

With a linear kernel: $\mathrm{HSIC} = \left\|\mathbb{E}[XY^\mathsf{T}] - \mathbb{E}[X]\,\mathbb{E}[Y]^\mathsf{T}\right\|_F^2$

Estimator based on kernel matrices:

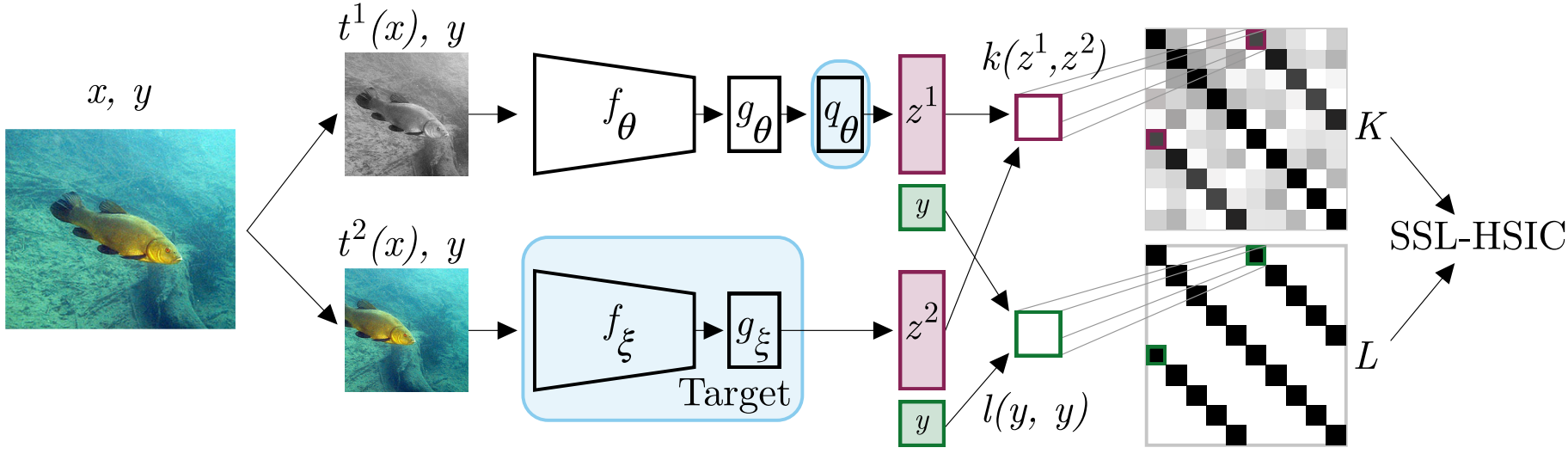
$$\widehat{\mathrm{HSIC}} = \frac{1}{(n-1)^2}\mathrm{tr}(KHLH)$$

- H is the centering matrix
- K is the kernel matrix on X
- L is the kernel matrix on Y

SSL-HSIC

$$\mathcal{L}_{\text{SSL-HSIC}} = -\,\text{HSIC}(Z, Y) + \gamma\sqrt{\text{HSIC}(Z, Z)}$$

(Y is just an indicator of which source image it came from)



Target representation Z' is output of $f_\theta(X)$

HSIC uses learned kernel $k(Z'_1, Z'_2) = \text{IMQ}(g(Z'_1), g(Z'_2))$

Your InfoNCE model is secretly a kernel method...

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathop{\mathbb{E}}_{(Z_1, Z_2)\sim \text{pos}}[k(Z_1, Z_2)] + \mathop{\mathbb{E}}_{Z_1} \log \mathop{\mathbb{E}}_{Z_2}[\exp k(Z_1, Z_2)]$$

Your InfoNCE model is secretly a kernel method...

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathop{\mathbb{E}}_{(Z_1,Z_2)\sim\text{pos}}[k(Z_1,Z_2)] + \mathop{\mathbb{E}}_{Z_1}\log\mathop{\mathbb{E}}_{Z_2}[\exp k(Z_1,Z_2)]$$

$$\approx \underbrace{-\mathop{\mathbb{E}}_{z_1,z_2\sim\text{pos}}[k(Z_1,Z_2)] + \mathop{\mathbb{E}}_{Z_1}\mathop{\mathbb{E}}_{Z_2}[k(Z_1,Z_2)]}_{\propto -\,\text{HSIC}(Z,Y)} + \underbrace{\frac{1}{2}\mathop{\mathbb{E}}_{Z_1}\left[\mathop{\text{Var}}_{Z_2}[k(Z_1,Z_2)]\right]}_{\text{variance penalty}}$$

Your InfoNCE model is secretly a kernel method...

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathop{\mathbb{E}}_{(Z_1,Z_2)\sim\text{pos}}[k(Z_1,Z_2)] + \mathop{\mathbb{E}}_{Z_1}\log\mathop{\mathbb{E}}_{Z_2}[\exp k(Z_1,Z_2)]$$

$$\approx \underbrace{-\mathop{\mathbb{E}}_{z_1,z_2\sim\text{pos}}[k(Z_1,Z_2)] + \mathop{\mathbb{E}}_{Z_1}\mathop{\mathbb{E}}_{Z_2}[k(Z_1,Z_2)]}_{\propto -\,\text{HSIC}(Z,Y)} + \underbrace{\frac{1}{2}\mathop{\mathbb{E}}_{Z_1}\left[\mathop{\text{Var}}_{Z_2}[k(Z_1,Z_2)]\right]}_{\text{variance penalty}}$$

Very similar loss! Just different regularizer

Your InfoNCE model is secretly a kernel method...

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathop{\mathbb{E}}_{(Z_1,Z_2)\sim\text{pos}}[k(Z_1,Z_2)] + \mathop{\mathbb{E}}_{Z_1}\log\mathop{\mathbb{E}}_{Z_2}[\exp k(Z_1,Z_2)]$$

$$\approx \underbrace{-\mathop{\mathbb{E}}_{z_1,z_2\sim\text{pos}}[k(Z_1,Z_2)] + \mathop{\mathbb{E}}_{Z_1}\mathop{\mathbb{E}}_{Z_2}[k(Z_1,Z_2)]}_{\propto -\,\text{HSIC}(Z,Y)} + \underbrace{\frac{1}{2}\mathop{\mathbb{E}}_{Z_1}\left[\mathop{\text{Var}}_{Z_2}[k(Z_1,Z_2)]\right]}_{\text{variance penalty}}$$

Very similar loss! Just different regularizer

When variance is small,

$$-\,\text{HSIC}(Z,Y) + \gamma\,\text{HSIC}(Z,Z) \le \mathcal{L}_{\text{InfoNCE}} + o(\text{variance})$$

Clustering interpretation

SSL-HSIC estimates agreement of Z with cluster structure of Y

With linear kernels:

$$-\operatorname{HSIC}(Z, Y) \propto \sum_{i=1}^{n} \sum_{p=1}^{m} \left\| Z_i^{(p)} - \bar{Z}_i \right\|^2 - nm$$

where $\bar{Z}_i$ is mean of the m augmentations

Clustering interpretation

SSL-HSIC estimates agreement of Z with cluster structure of Y

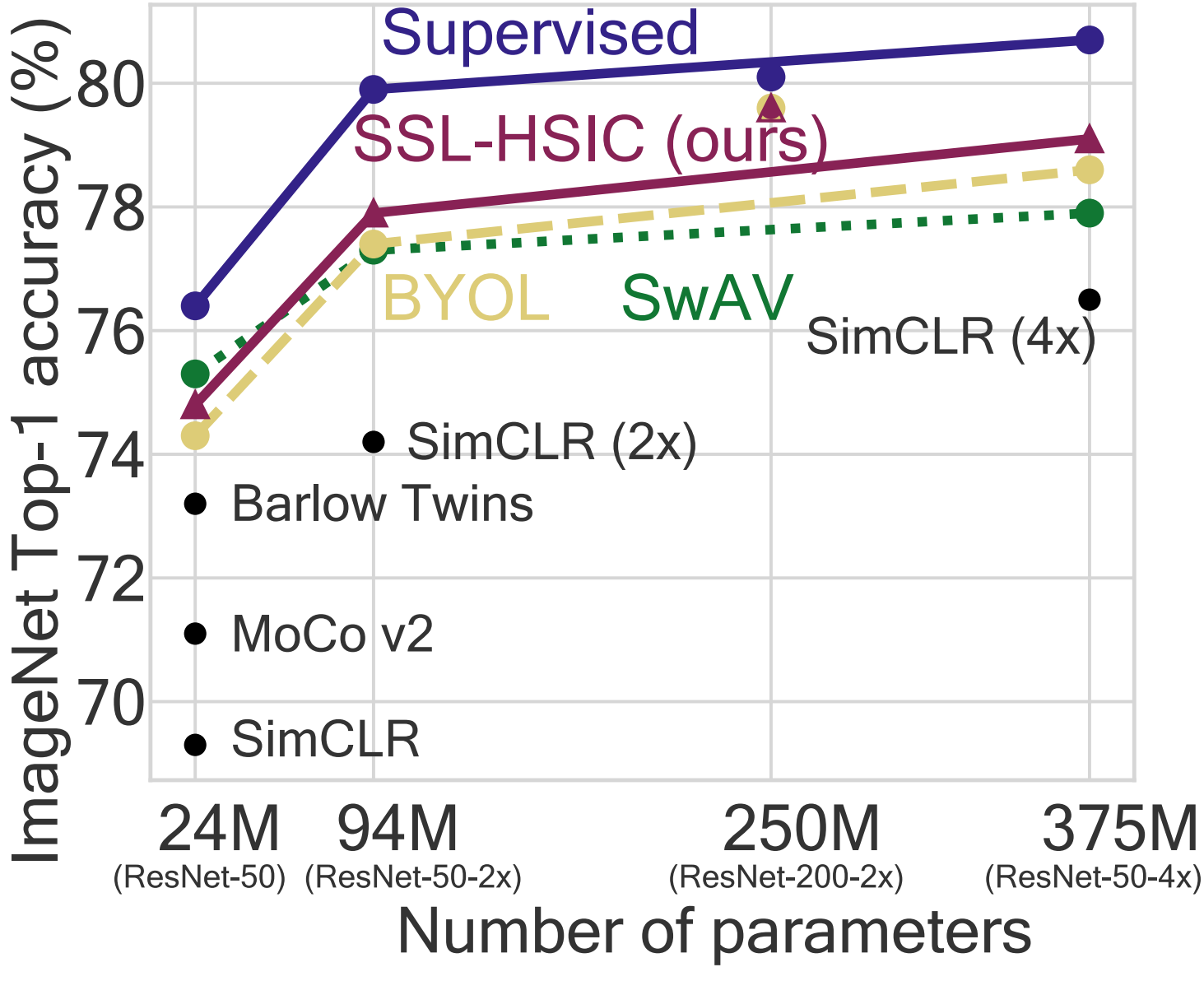
With linear kernels:

$$-\mathrm{HSIC}(Z, Y) \propto \sum_{i=1}^{n}\sum_{p=1}^{m}\left\|Z_i^{(p)} - \bar{Z}_i\right\|^2 - nm$$

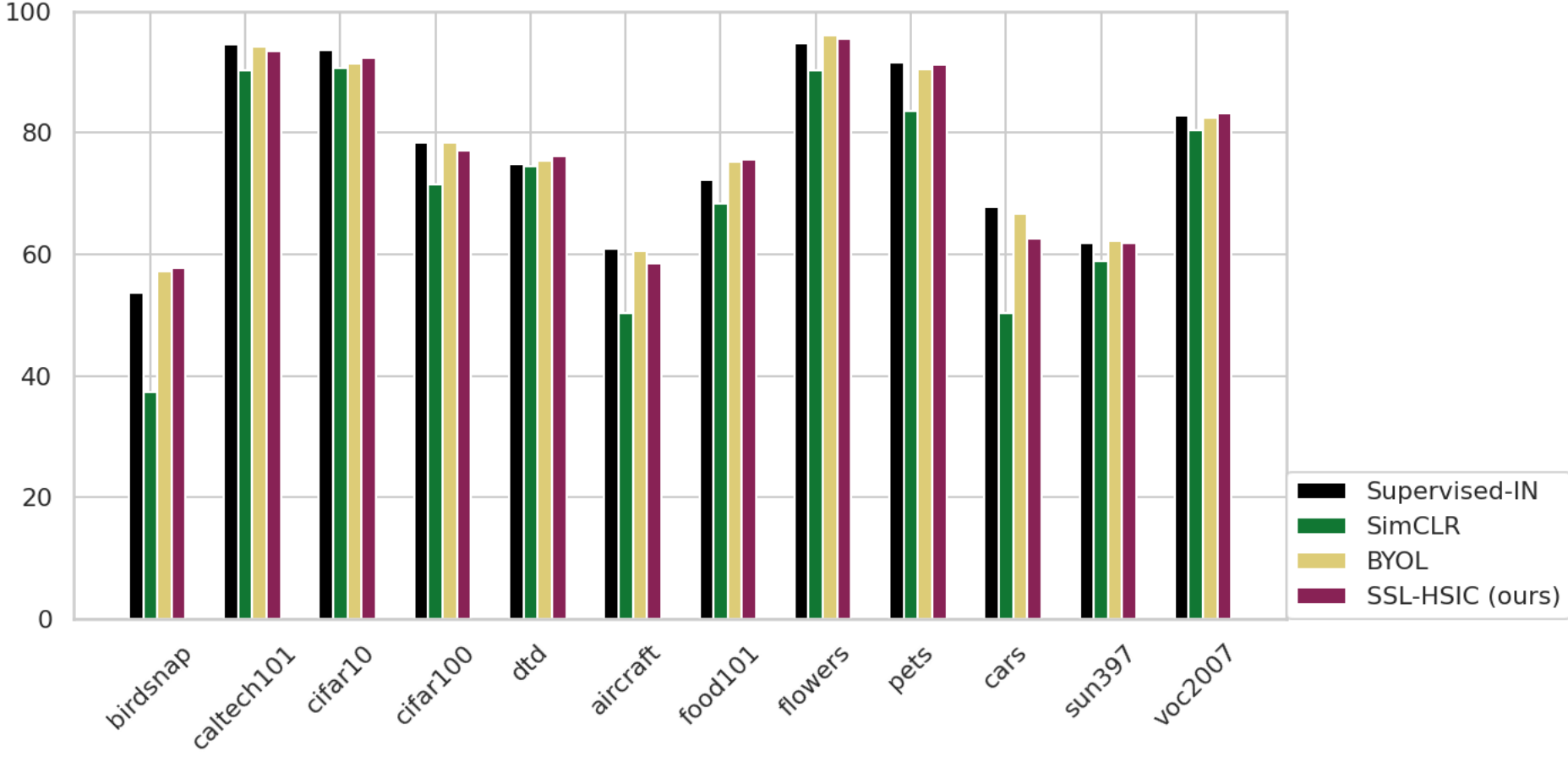
where $\bar{Z}_i$ is mean of the m augmentations

Resembles BYOL loss with no target network (but still works!)

ImageNet results: linear evaluation



Transfer from ImageNet to classification tasks



III: Training implicit generative models

Given samples from a distribution $\mathbb{P}$ over $\mathcal{X}$,
we want a model that can produce new samples from $\mathbb{Q}_\theta \approx \mathbb{P}$

$X \sim \mathbb{P}$

$Y \sim \mathbb{Q}_\theta$

III: Training implicit generative models

we want $\mathbb{Q}_\theta \approx \mathbb{P}$

thispersondoesnotexist.com

III: Training implicit generative models

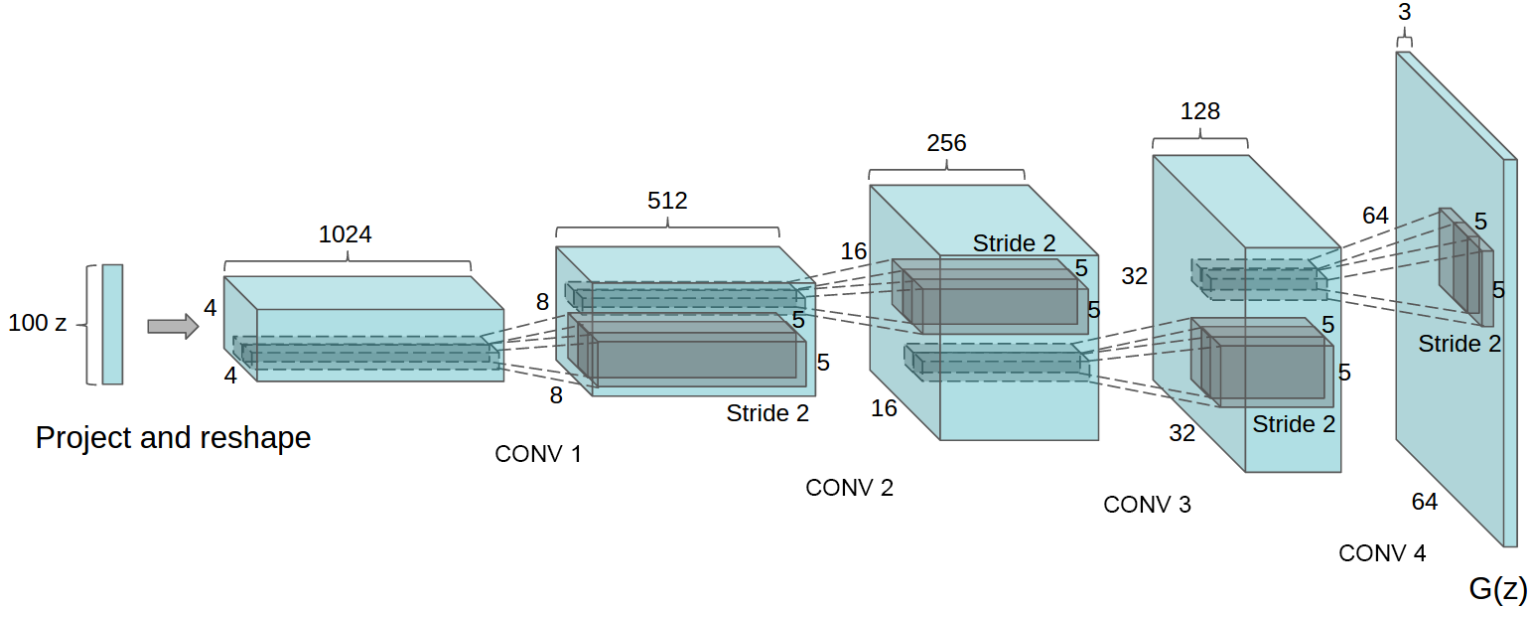
Given samples from a distribution $\mathbb{P}$ over $\mathcal{X}$,
we want a model that can produce new samples from $\mathbb{Q}_\theta \approx \mathbb{P}$

"Everybody Dance Now" [Chan+ ICCV-19]

Generator networks

Fixed distribution of latents: $Z \sim \mathbf{Uniform}\left([-1,1]^{100}\right)$

Maps through a network: $G_\theta(Z) \sim \mathbb{Q}_\theta$

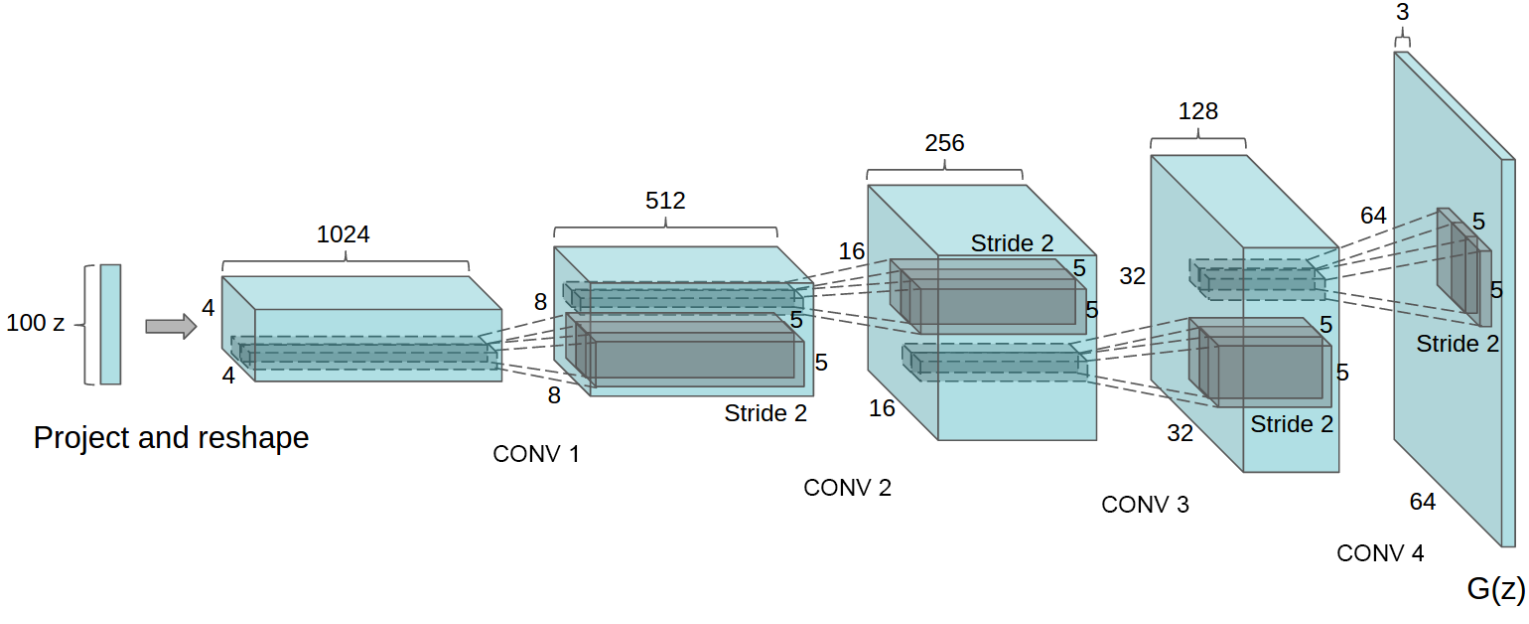


DCGAN generator [Radford+ ICLR-16]

Generator networks

Fixed distribution of latents: $Z \sim \mathbf{Uniform}\left([-1,1]^{100}\right)$

Maps through a network: $G_\theta(Z) \sim \mathbb{Q}_\theta$



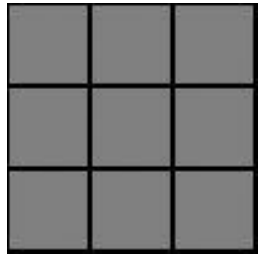
DCGAN generator [Radford+ ICLR-16]

How to choose θ?

GANs and their flaws

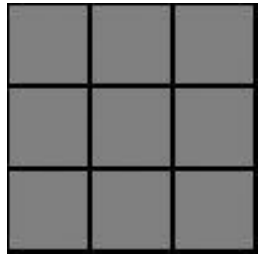
- GANs [Goodfellow+ NeurIPS-14] minimize discriminator accuracy (like classifier test) between $\mathbb{P}$ and $\mathbb{Q}_\theta$
- Problem: if there's a perfect classifier, discontinuous loss, no gradient to improve it [Arjovsky/Bottou ICLR-17]
- Disjoint at init:

$\mathbb{P}$: 

$\mathbb{Q}_{\theta_0}$: 

GANs and their flaws

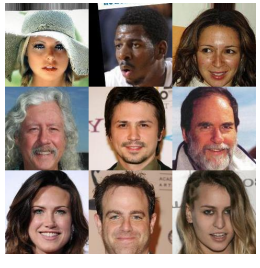
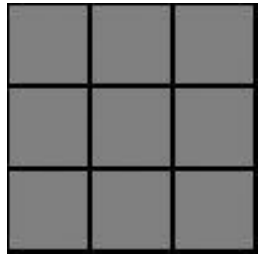
- GANs [Goodfellow+ NeurIPS-14] minimize discriminator accuracy (like classifier test) between $\mathbb{P}$ and $\mathbb{Q}_\theta$
- Problem: if there's a perfect classifier, discontinuous loss, no gradient to improve it [Arjovsky/Bottou ICLR-17]
- Disjoint at init:

$\mathbb{P}$:  $\mathbb{Q}_{\theta_0}$: 

- For usual $G_\theta : \mathbb{R}^{100} \to \mathbb{R}^{64\times64\times3}$, $\mathbb{Q}_\theta$ is supported on a countable union of manifolds with dim ≤ 100

GANs and their flaws

- GANs [Goodfellow+ NeurIPS-14] minimize discriminator accuracy (like classifier test) between $\mathbb{P}$ and $\mathbb{Q}_\theta$
- Problem: if there's a perfect classifier, discontinuous loss, no gradient to improve it [Arjovsky/Bottou ICLR-17]
- Disjoint at init:

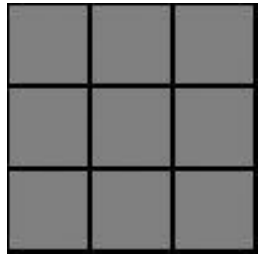
$\mathbb{P}$:  $\mathbb{Q}_{\theta_0}$: 

- For usual $G_\theta : \mathbb{R}^{100} \to \mathbb{R}^{64\times64\times3}$, $\mathbb{Q}_\theta$ is supported on a countable union of manifolds with dim ≤ 100
- "Natural image manifold" usually considered low-dim

GANs and their flaws

- GANs [Goodfellow+ NeurIPS-14] minimize discriminator accuracy (like classifier test) between $\mathbb{P}$ and $\mathbb{Q}_\theta$
- Problem: if there's a perfect classifier, discontinuous loss, no gradient to improve it [Arjovsky/Bottou ICLR-17]
- Disjoint at init:

$\mathbb{P}$: 

$\mathbb{Q}_{\theta_0}$: 

- For usual $G_\theta : \mathbb{R}^{100} \to \mathbb{R}^{64\times64\times3}$, $\mathbb{Q}_\theta$ is supported on a countable union of manifolds with dim ≤ 100
- “Natural image manifold” usually considered low-dim
- Won't align at init, so won't ever align

WGANs and MMD GANs

- Integral probability metrics with “smooth” $\mathcal{F}$ are continuous
- WGAN: $\mathcal{F}$ a set of neural networks satisfying $\|f\|_L \leq 1$
- WGAN-GP: instead penalize $\mathbb{E}\|\nabla_x f(x)\|$ near the data

WGANs and MMD GANs

- Integral probability metrics with "smooth" $\mathcal{F}$ are continuous
- WGAN: $\mathcal{F}$ a set of neural networks satisfying $\|f\|_L \leq 1$
- WGAN-GP: instead penalize $\mathbb{E}\|\nabla_x f(x)\|$ near the data
- Both losses are MMD with $k_\psi(x, y) = \phi_\psi(x)\phi_\psi(y)$

WGANs and MMD GANs

- Integral probability metrics with "smooth" $\mathcal{F}$ are continuous
- WGAN: $\mathcal{F}$ a set of neural networks satisfying $\|f\|_L \leq 1$
- WGAN-GP: instead penalize $\mathbb{E}\|\nabla_x f(x)\|$ near the data
- Both losses are MMD with $k_\psi(x, y) = \phi_\psi(x)\phi_\psi(y)$
  - $$\min_\theta \left[\mathcal{D}^{\Psi}_{\mathrm{MMD}}(\mathbb{P}, \mathbb{Q}_\theta) = \sup_{\psi \in \Psi} \mathrm{MMD}_\psi(\mathbb{P}, \mathbb{Q}_\theta) \right]$$

WGANs and MMD GANs

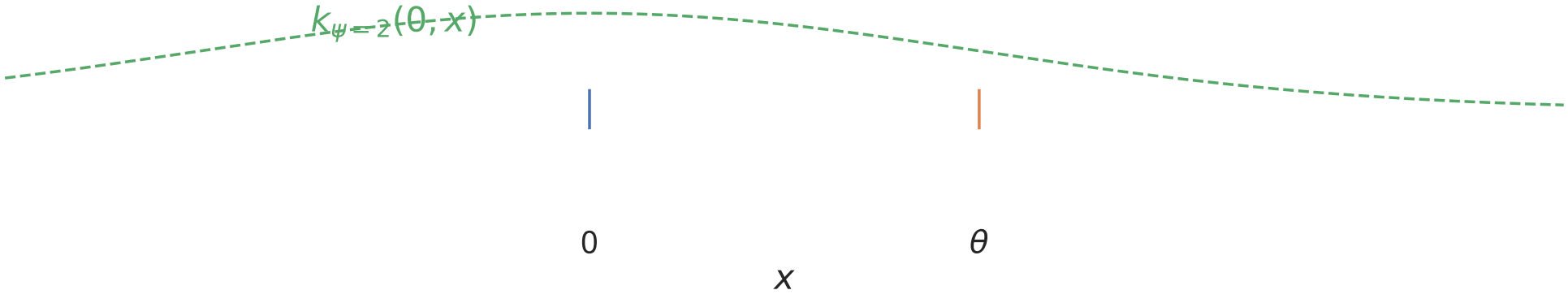
- Integral probability metrics with "smooth" $\mathcal{F}$ are continuous
- WGAN: $\mathcal{F}$ a set of neural networks satisfying $\|f\|_L \leq 1$
- WGAN-GP: instead penalize $\mathbb{E}\|\nabla_x f(x)\|$ near the data
- Both losses are MMD with $k_\psi(x, y) = \phi_\psi(x)\phi_\psi(y)$
  - $$\min_\theta \left[\mathcal{D}^{\Psi}_{\text{MMD}}(\mathbb{P}, \mathbb{Q}_\theta) = \sup_{\psi \in \Psi} \text{MMD}_\psi(\mathbb{P}, \mathbb{Q}_\theta) \right]$$
- Some kind of constraint on ϕ_ψ is important!

Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

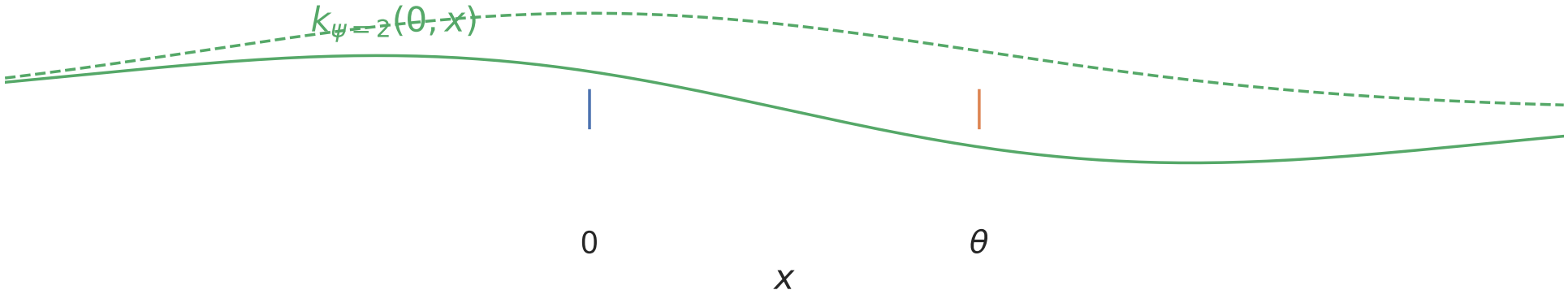
Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:



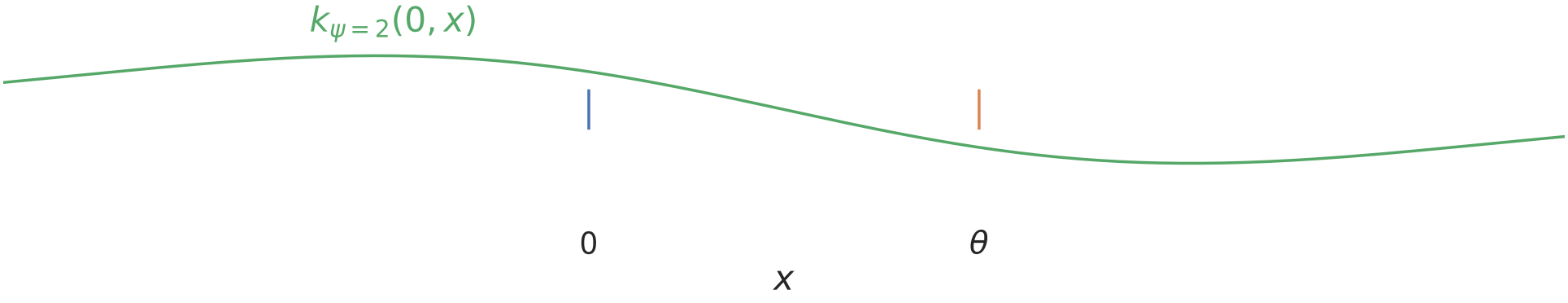
Non-smoothness of plain MMD GANs

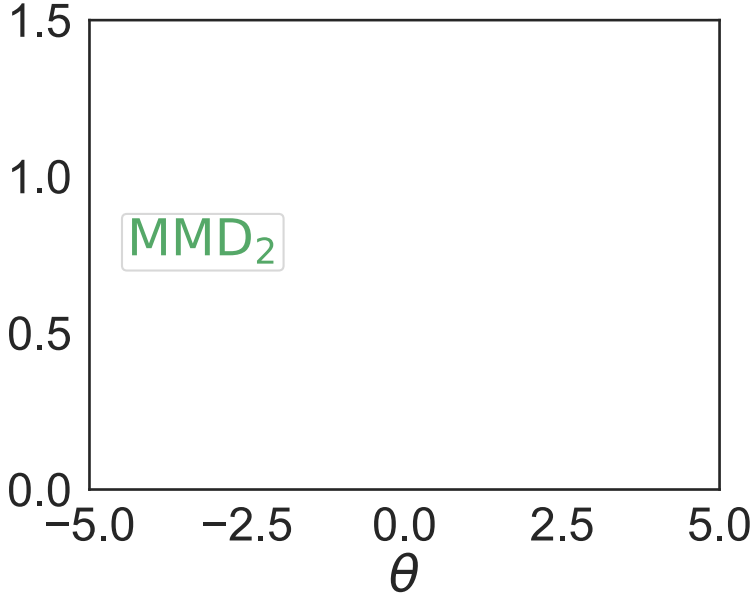
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:



Non-smoothness of plain MMD GANs

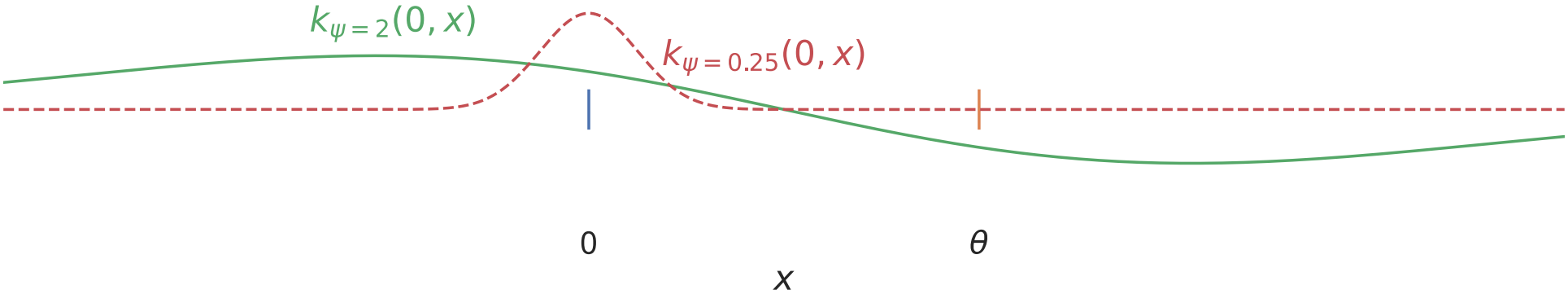
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

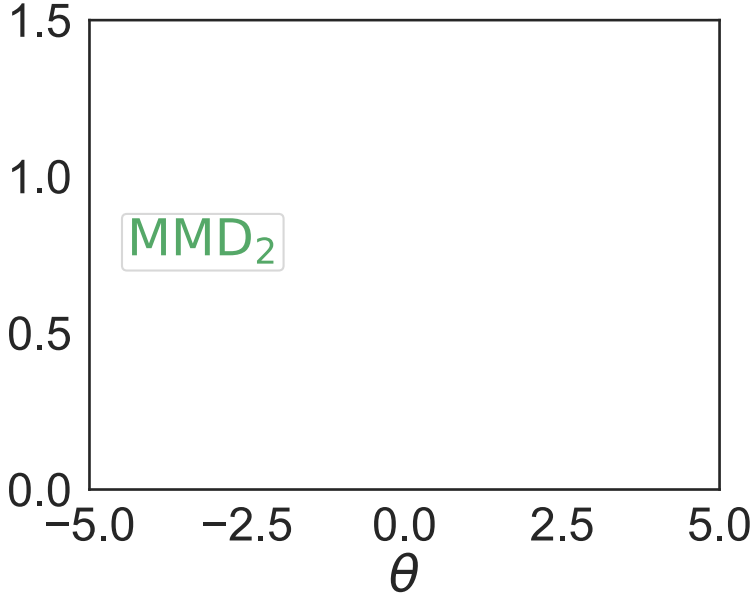




Non-smoothness of plain MMD GANs

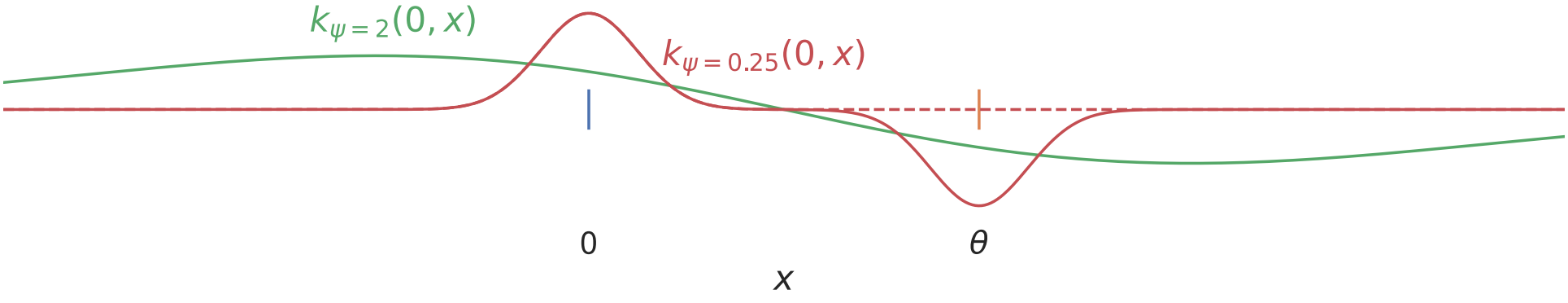
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

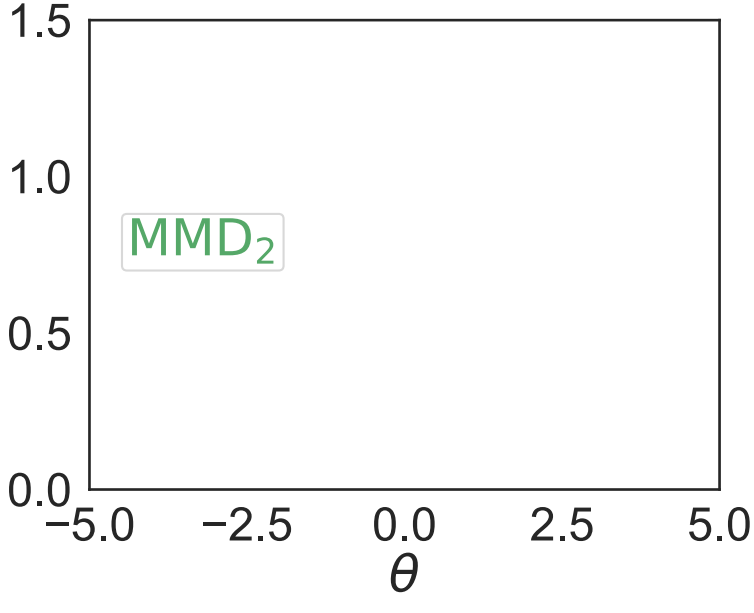




Non-smoothness of plain MMD GANs

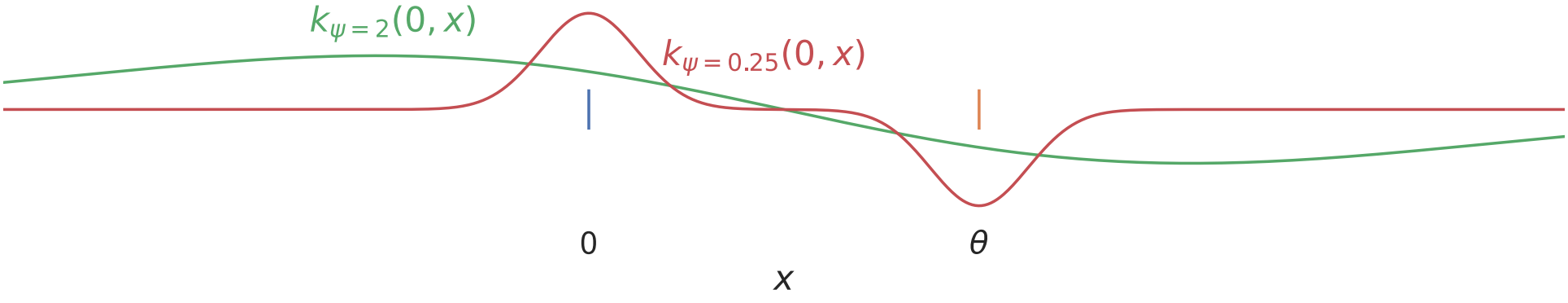
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

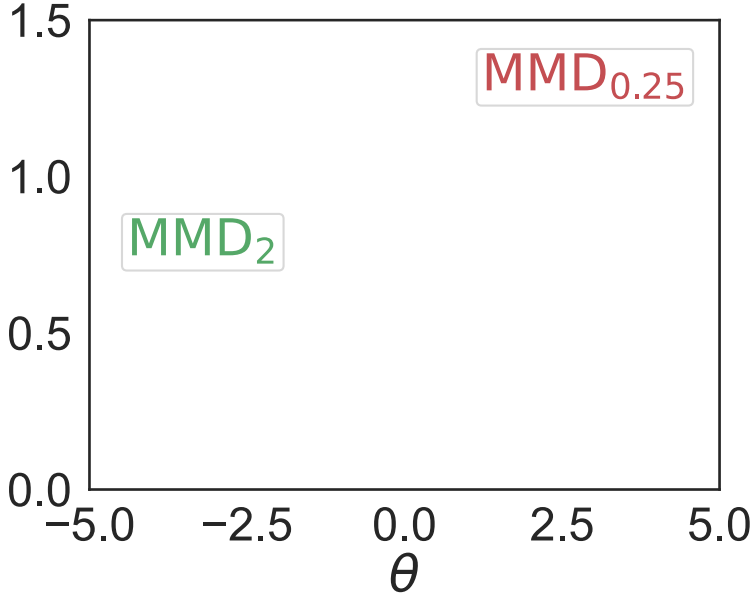




Non-smoothness of plain MMD GANs

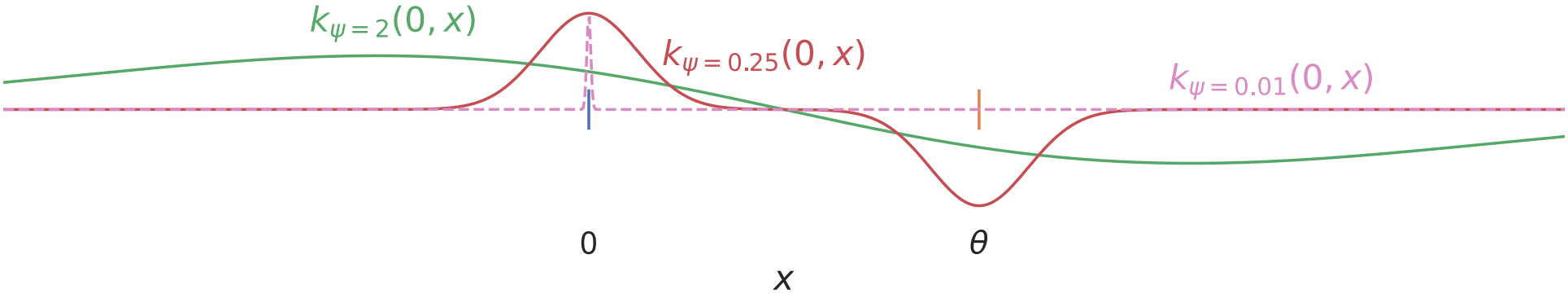
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

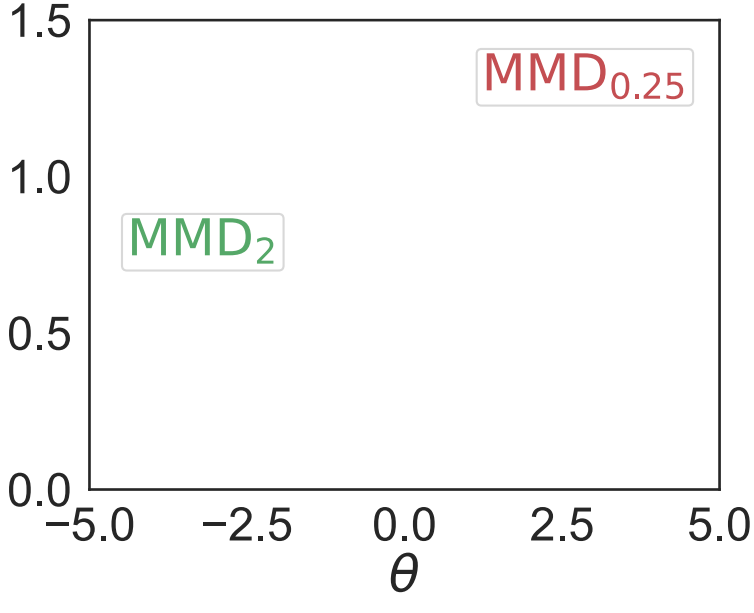




Non-smoothness of plain MMD GANs

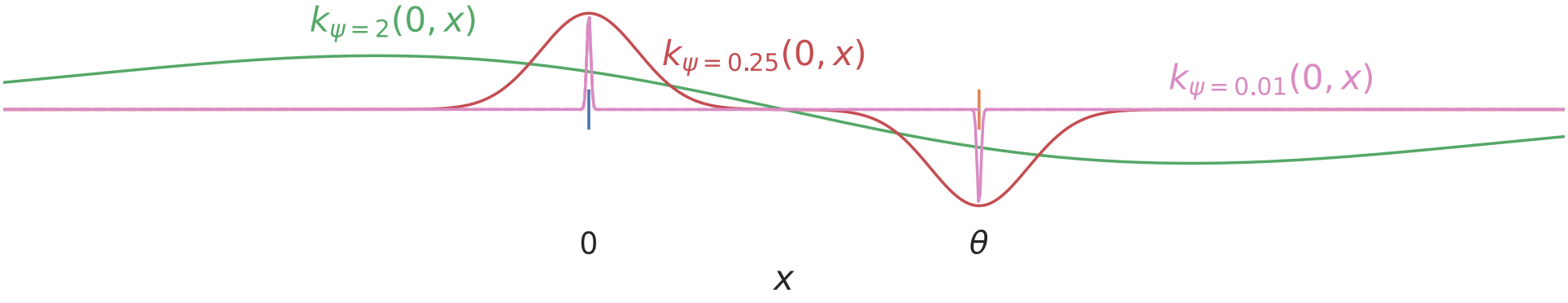
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

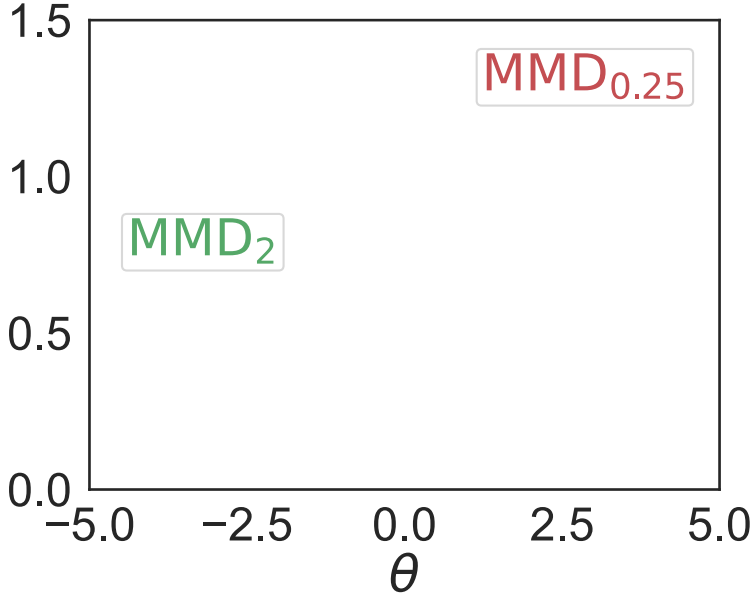




Non-smoothness of plain MMD GANs

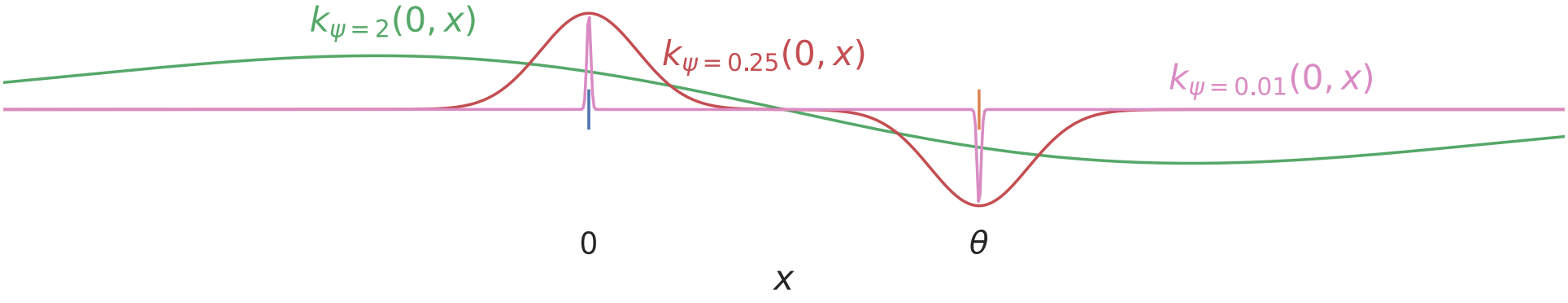
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

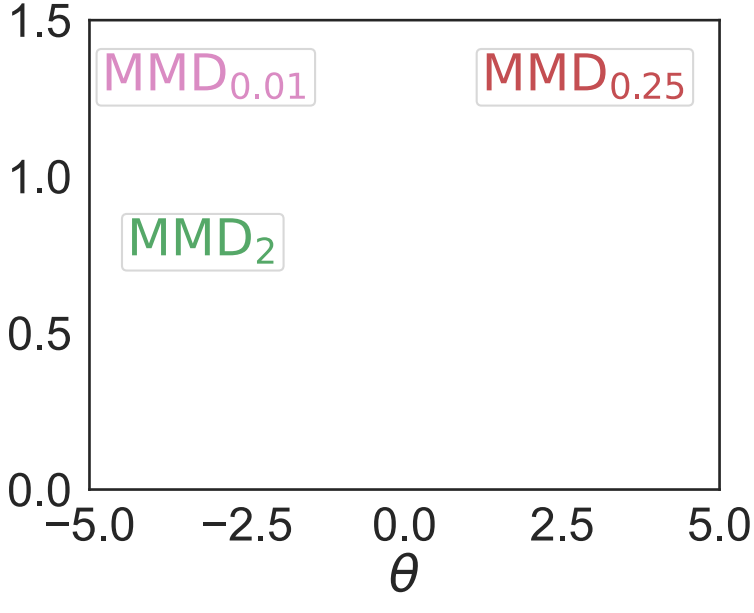




Non-smoothness of plain MMD GANs

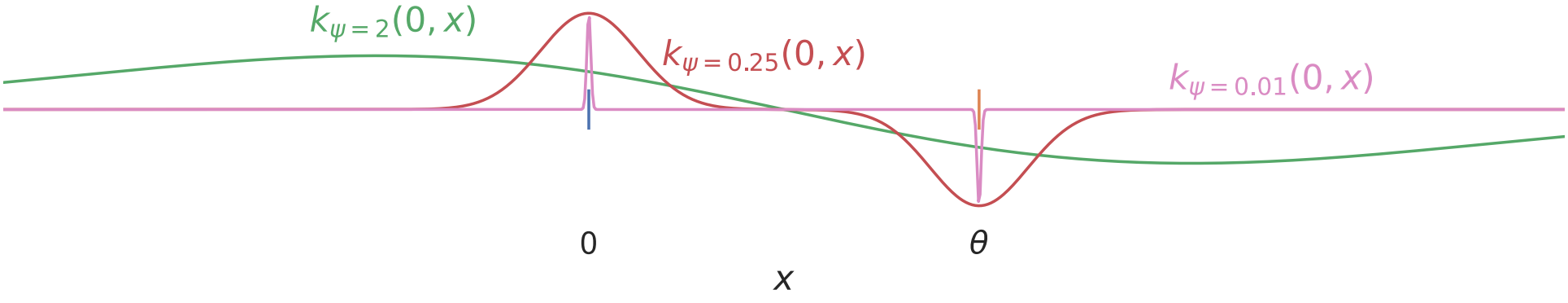
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

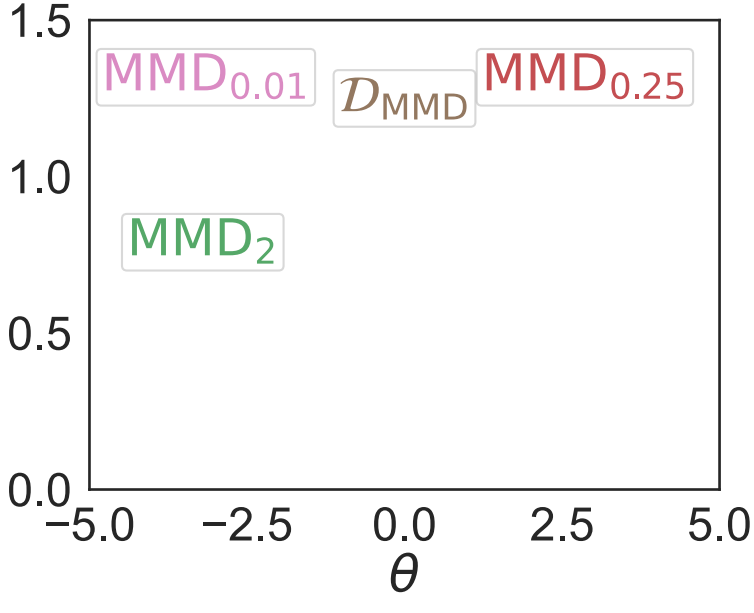




Non-smoothness of plain MMD GANs

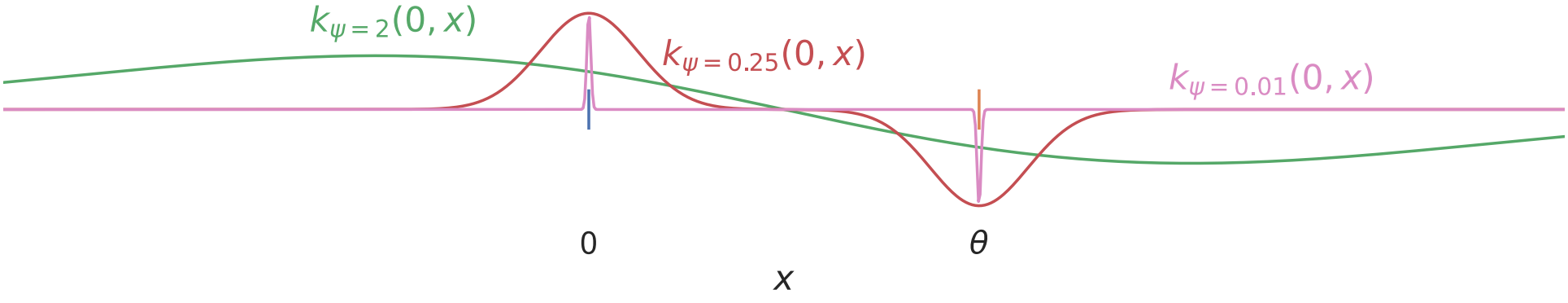
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

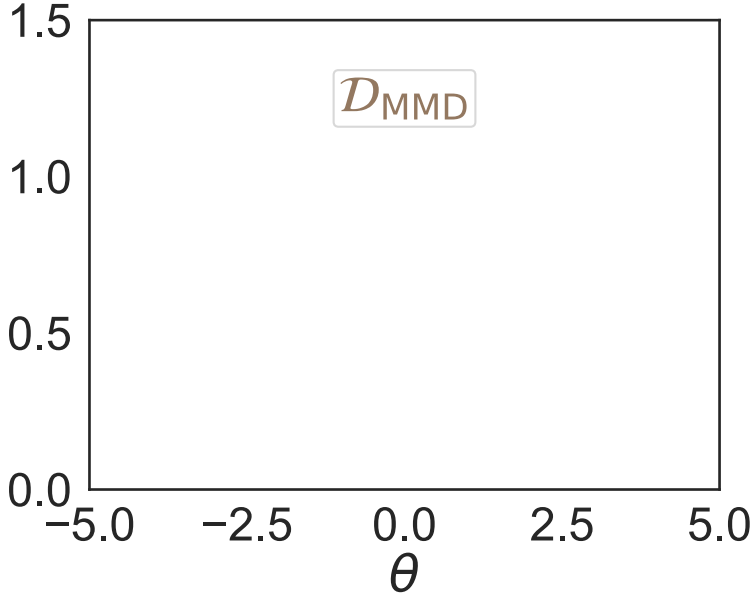




Non-smoothness of plain MMD GANs

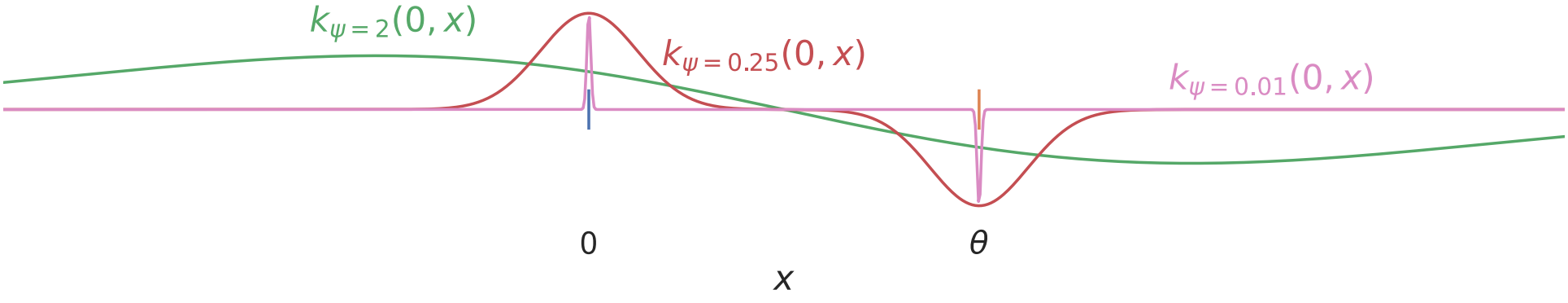
Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

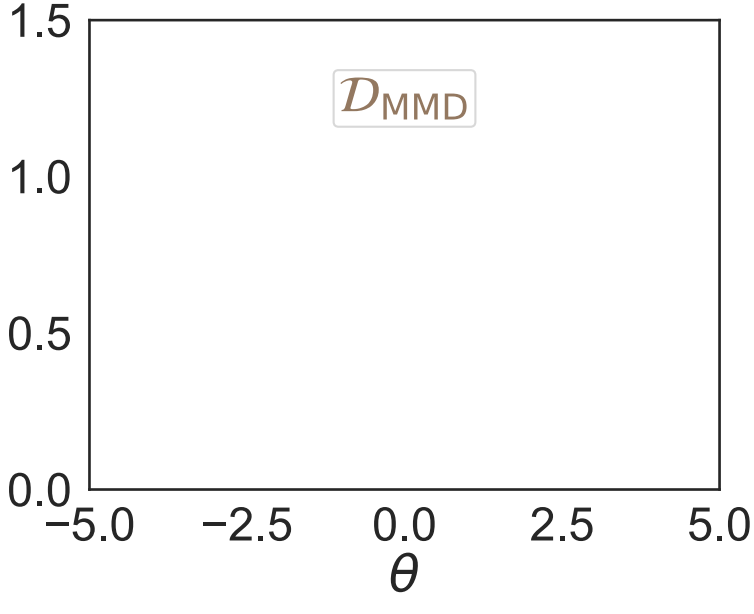


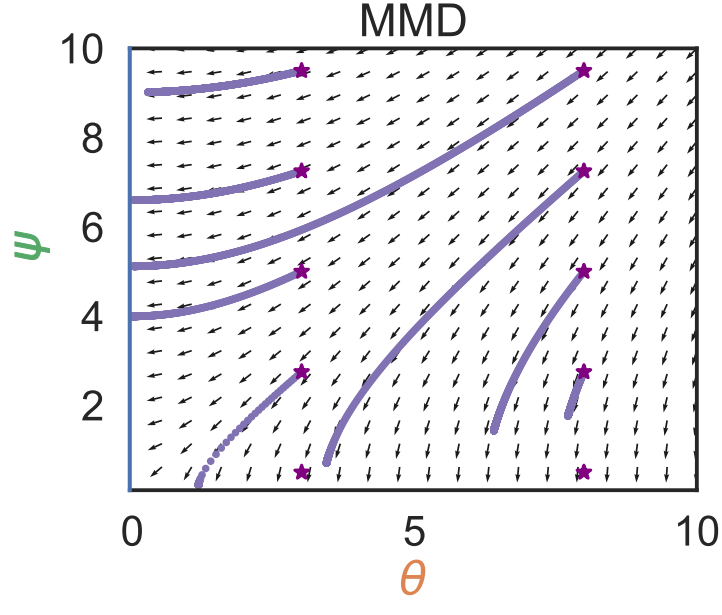


Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

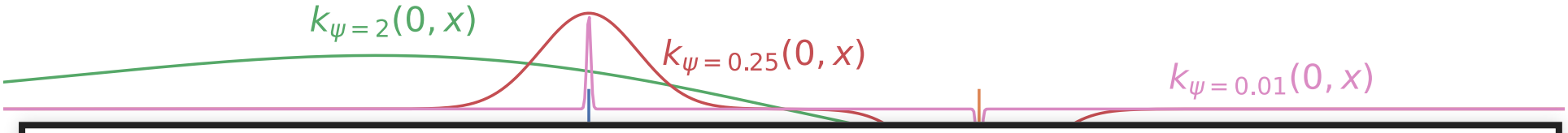






Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

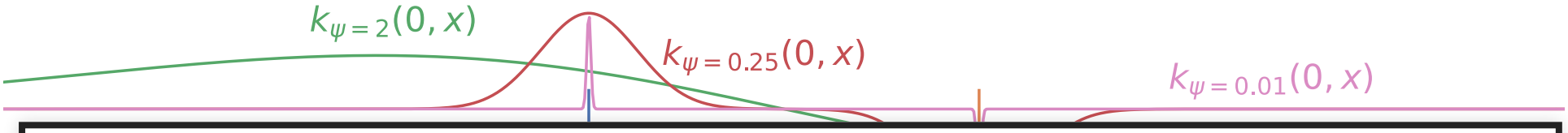


- Just need to stay away from tiny bandwidths ψ

θ θ

Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

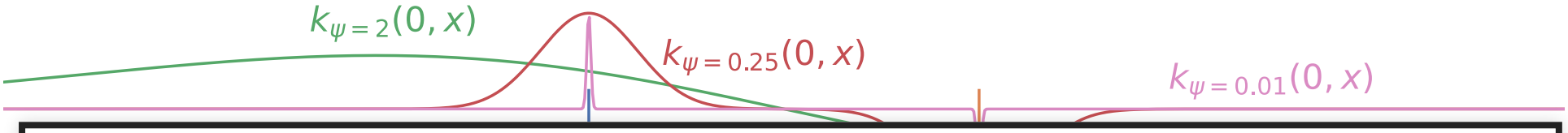


- Just need to stay away from tiny bandwidths ψ
- ...deep kernel analogue is hard.

θ θ

Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

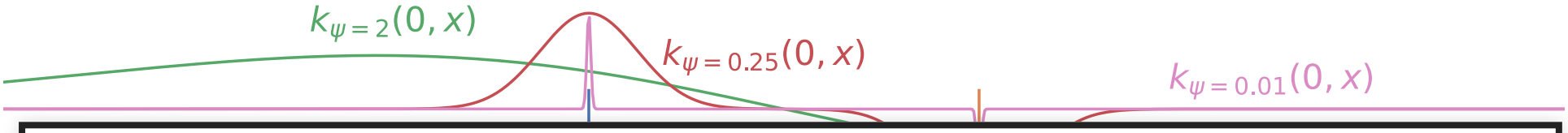


- Just need to stay away from tiny bandwidths ψ
- ...deep kernel analogue is hard.
- Instead, keep witness function from being too steep

θ θ

Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:

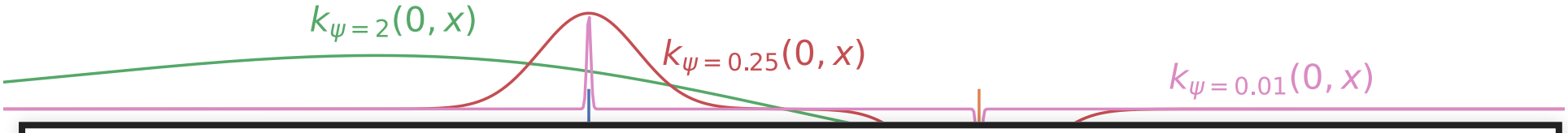


- Just need to stay away from tiny bandwidths ψ
- ...deep kernel analogue is hard.
- Instead, keep witness function from being too steep
- $\sup_x \|\nabla f(x)\|$ would give Wasserstein
  - Nice distance, but hard to estimate

θ θ

Non-smoothness of plain MMD GANs

Illustrative problem in $\mathbb{R}$, DiracGAN [Mescheder+ ICML-18]:



- Just need to stay away from tiny bandwidths ψ
- ...deep kernel analogue is hard.
- Instead, keep witness function from being too steep
- $\sup_x \|\nabla f(x)\|$ would give Wasserstein
  - Nice distance, but hard to estimate
- Control $\|\nabla f(\tilde{X})\|$ *on average, near the data*
  - [Gulrajani+ NeurIPS-17 / Roth+ NeurIPS-17 / Mescheder+ ICML-18]

θ θ

MMD-GAN with gradient control

- If Ψ gives uniformly Lipschitz critics, $\mathcal{D}_{\mathrm{MMD}}^{\Psi}$ is smooth
- Original MMD-GAN paper [Li+ NeurIPS-17]: box constraint

MMD-GAN with gradient control

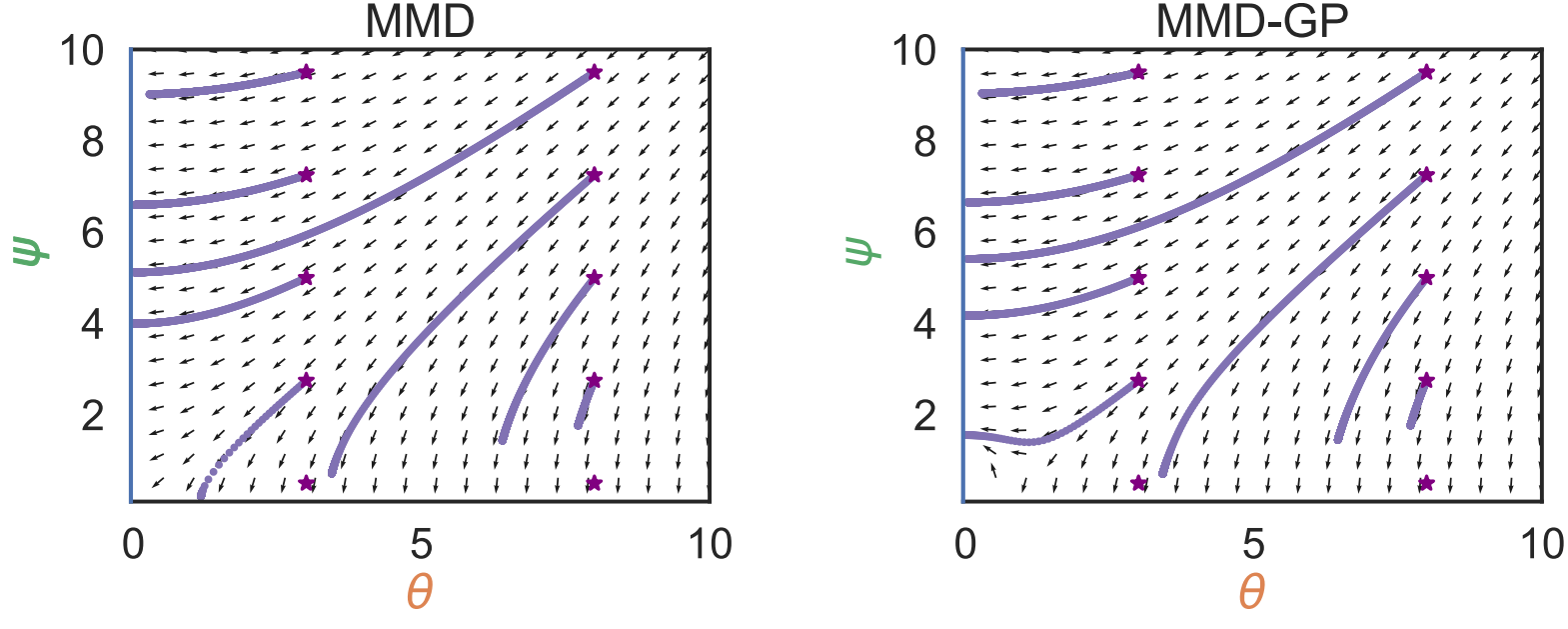
- If Ψ gives uniformly Lipschitz critics, $\mathcal{D}^{\Psi}_{\mathrm{MMD}}$ is smooth
- Original MMD-GAN paper [Li+ NeurIPS-17]: box constraint
- We [Bińkowski+ ICLR-18] used gradient penalty on critic instead

MMD-GAN with gradient control

- If Ψ gives uniformly Lipschitz critics, $\mathcal{D}_{\mathrm{MMD}}^{\Psi}$ is smooth
- Original MMD-GAN paper [Li+ NeurIPS-17]: box constraint
- We [Bińkowski+ ICLR-18] used gradient penalty on critic instead
  - Better in practice, but doesn't fix the Dirac problem...

MMD-GAN with gradient control

- If Ψ gives uniformly Lipschitz critics, $\mathcal{D}^{\Psi}_{\mathrm{MMD}}$ is smooth
- Original MMD-GAN paper [Li+ NeurIPS-17]: box constraint
- We [Bińkowski+ ICLR-18] used gradient penalty on critic instead
  - Better in practice, but doesn't fix the Dirac problem...



New distance: Scaled MMD

Want to ensure $\mathbb{E}_{\tilde{X} \sim \mathbb{S}}[\|\nabla f(\tilde{X})\|^2] \leq 1$

New distance: Scaled MMD

Want to ensure $\mathbb{E}_{\tilde{X} \sim \mathbb{S}}[\|\nabla f(\tilde{X})\|^2] \leq 1$

Can solve with $\langle \partial_i \phi(x), f \rangle_{\mathcal{H}} = \partial_i f(x)$...but too expensive!

New distance: Scaled MMD

Want to ensure $\mathbb{E}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2] \leq 1$

Can solve with $\langle \partial_i \phi(x), f \rangle_{\mathcal{H}} = \partial_i f(x)$...but too expensive!

Guaranteed if $\|f\|_{\mathcal{H}} \leq \sigma_{\mathbb{S},k,\lambda}$

$$\sigma_{\mathbb{S},k,\lambda} := \left(\lambda + \mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}\left[k(\tilde{X},\tilde{X}) + [\nabla_1 \cdot \nabla_2 k](\tilde{X},\tilde{X})\right]\right)^{-\frac{1}{2}}$$

New distance: Scaled MMD

Want to ensure $\mathbb{E}_{\tilde{X} \sim \mathbb{S}}[\|\nabla f(\tilde{X})\|^2] \leq 1$

Can solve with $\langle \partial_i \phi(x), f \rangle_{\mathcal{H}} = \partial_i f(x)$...but too expensive!

Guaranteed if $\|f\|_{\mathcal{H}} \leq \sigma_{\mathbb{S},k,\lambda}$

$$\sigma_{\mathbb{S},k,\lambda} := \left(\lambda + \mathop{\mathbb{E}}_{\tilde{X} \sim \mathbb{S}} \left[k(\tilde{X}, \tilde{X}) + [\nabla_1 \cdot \nabla_2 k](\tilde{X}, \tilde{X}) \right] \right)^{-\frac{1}{2}}$$

Gives distance $\mathrm{SMMD}_{\mathbb{S},k,\lambda}(\mathbb{P}, \mathbb{Q}) = \sigma_{\mathbb{S},k,\lambda} \, \mathrm{MMD}_k(\mathbb{P}, \mathbb{Q})$

New distance: Scaled MMD

Want to ensure $\mathbb{E}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2] \leq 1$

Can solve with $\langle \partial_i \phi(x), f\rangle_{\mathcal{H}} = \partial_i f(x)$...but too expensive!

Guaranteed if $\|f\|_{\mathcal{H}} \leq \sigma_{\mathbb{S},k,\lambda}$

$$\sigma_{\mathbb{S},k,\lambda} := \left(\lambda + \mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}\left[k(\tilde{X},\tilde{X}) + [\nabla_1\cdot\nabla_2 k](\tilde{X},\tilde{X})\right]\right)^{-\frac{1}{2}}$$

Gives distance $\mathbf{SMMD}_{\mathbb{S},k,\lambda}(\mathbb{P},\mathbb{Q}) = \sigma_{\mathbb{S},k,\lambda}\,\mathbf{MMD}_k(\mathbb{P},\mathbb{Q})$

$$\mathcal{D}^{\Psi}_{\mathrm{MMD}} \ \mathbf{has}\ \mathcal{F} = \bigcup_{\psi\in\Psi}\left\{f : \|f\|_{\mathcal{H}_\psi} \leq 1\right\}$$

$$\mathcal{D}^{\mathbb{S},\Psi,\lambda}_{\mathrm{SMMD}} \ \mathbf{has}\ \mathcal{F} = \bigcup_{\psi\in\Psi}\left\{f : \|f\|_{\mathcal{H}_\psi} \leq \sigma_{\mathbb{S},k,\lambda}\right\}$$

Deriving the Scaled MMD

$$\mathop{\mathbb{E}}_{\tilde{X} \sim \mathbb{S}} \left[\|\nabla f(\tilde{X})\|^2 \right] \leq 1$$

Deriving the Scaled MMD

$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[f(\tilde{X})^2]+\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2]+\lambda\|f\|_{\mathcal{H}}^2\leq 1$$

Deriving the Scaled MMD

$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[f(\tilde{X})^2]+\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2]+\lambda\|f\|_{\mathcal{H}}^2\leq 1$$

$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[f(\tilde{X})^2]=\left\langle f,\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}\left[k(\tilde{X},\cdot)\otimes k(\tilde{X},\cdot)\right]f\right\rangle$$

Deriving the Scaled MMD

$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[f(\tilde{X})^2]+\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2]+\lambda\|f\|_{\mathcal{H}}^2\leq 1$$

$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[f(\tilde{X})^2]=\left\langle f,\ \mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}\left[k(\tilde{X},\cdot)\otimes k(\tilde{X},\cdot)\right]f\right\rangle$$

$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2]=\left\langle f,\ \mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}\left[\sum_{i=1}^{d}\partial_i k(\tilde{X},\cdot)\otimes\partial_i k(\tilde{X},\cdot)\right]f\right\rangle$$

Deriving the Scaled MMD

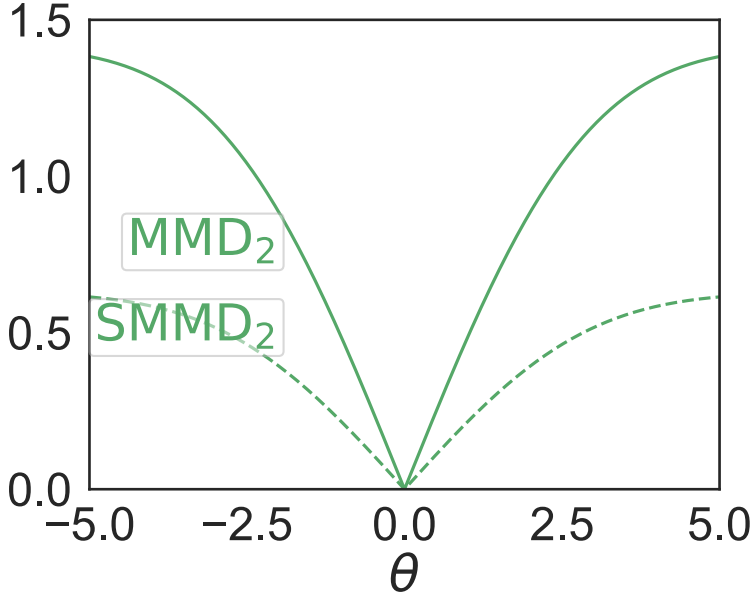
$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[f(\tilde{X})^2]+\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2]+\lambda\|f\|_{\mathcal{H}}^2\leq 1$$

$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[f(\tilde{X})^2]=\left\langle f,\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}\left[k(\tilde{X},\cdot)\otimes k(\tilde{X},\cdot)\right]f\right\rangle$$

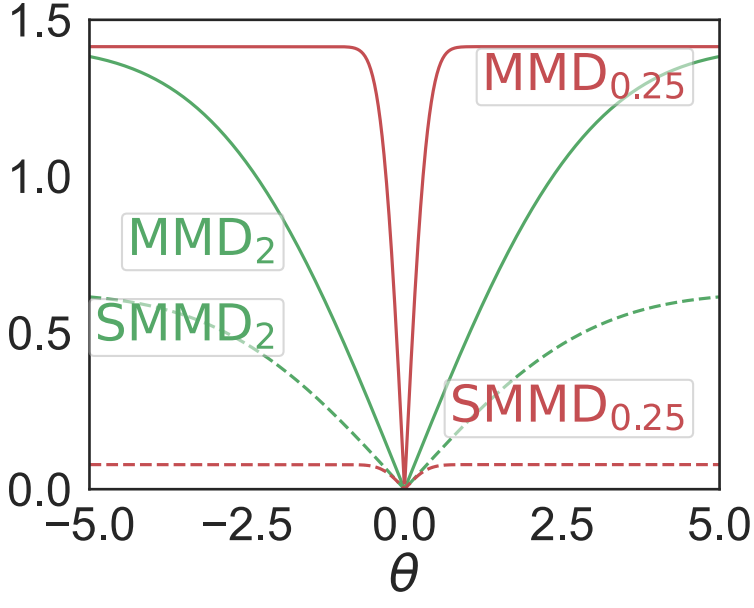
$$\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}[\|\nabla f(\tilde{X})\|^2]=\left\langle f,\mathop{\mathbb{E}}_{\tilde{X}\sim\mathbb{S}}\left[\sum_{i=1}^{d}\partial_i k(\tilde{X},\cdot)\otimes\partial_i k(\tilde{X},\cdot)\right]f\right\rangle$$

$$\langle f,D_\lambda f\rangle\leq\|D_\lambda\|\,\|f\|_{\mathcal{H}}^2\leq\sigma_{\mathbb{S},k,\lambda}^{-2}\|f\|_{\mathcal{H}}^2$$

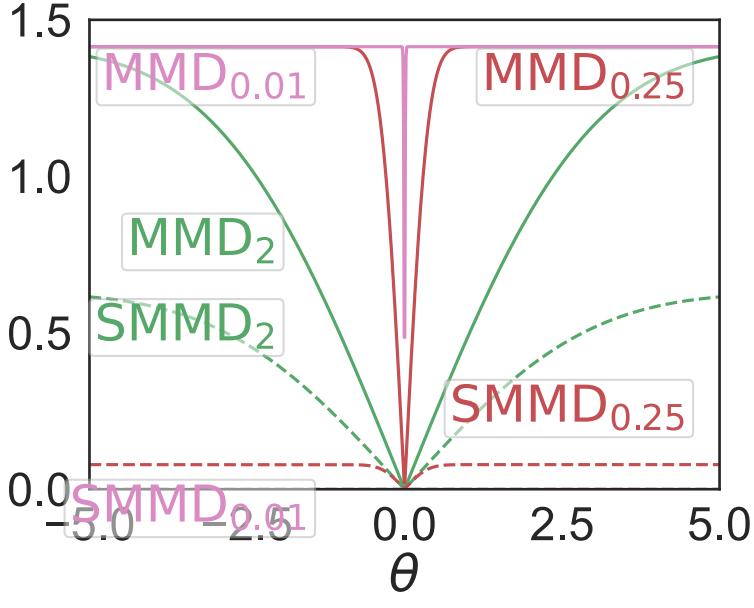
Smoothness of $\mathcal{D}_{\mathrm{SMMD}}$



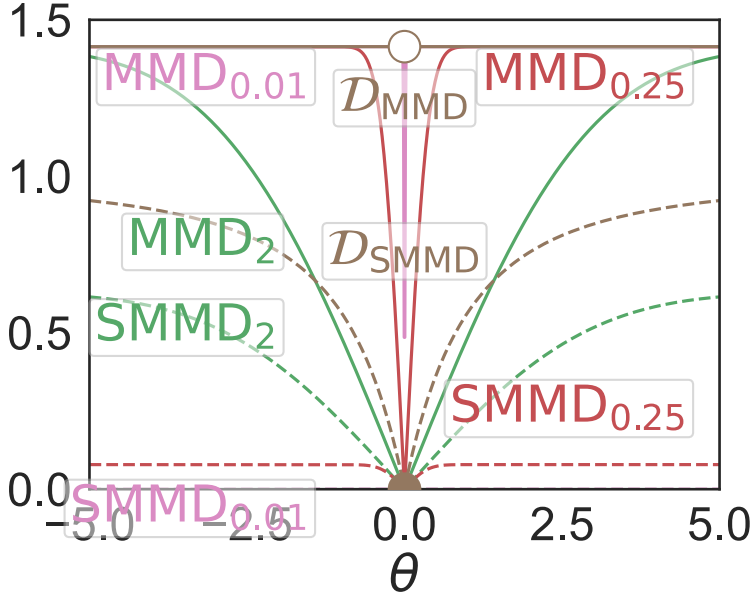
Smoothness of $\mathcal{D}_{\mathrm{SMMD}}$



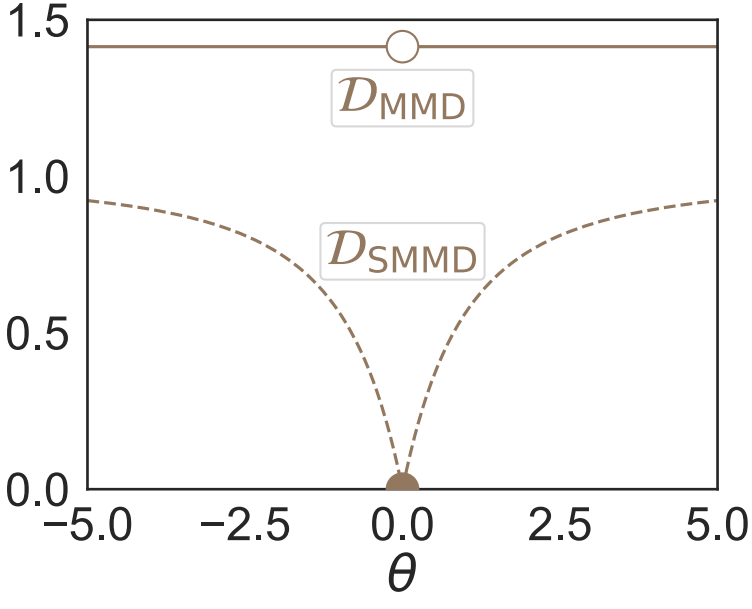
Smoothness of $\mathcal{D}_{\mathrm{SMMD}}$

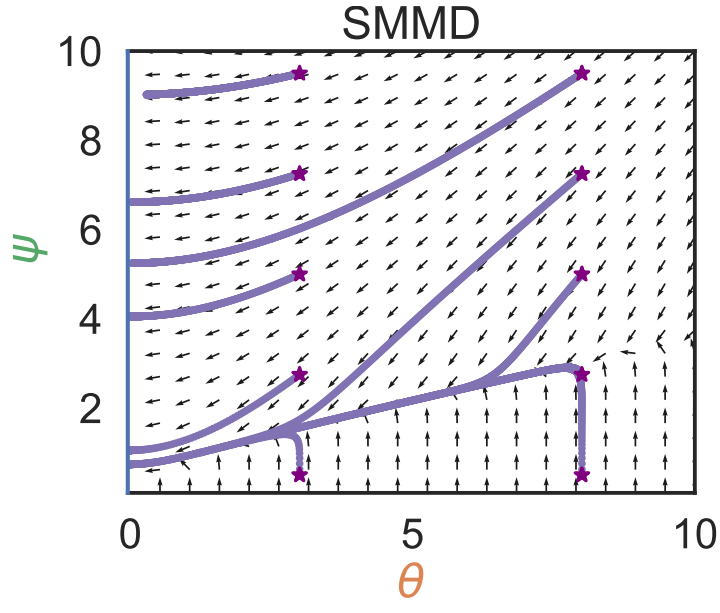


Smoothness of $\mathcal{D}_{\mathrm{SMMD}}$

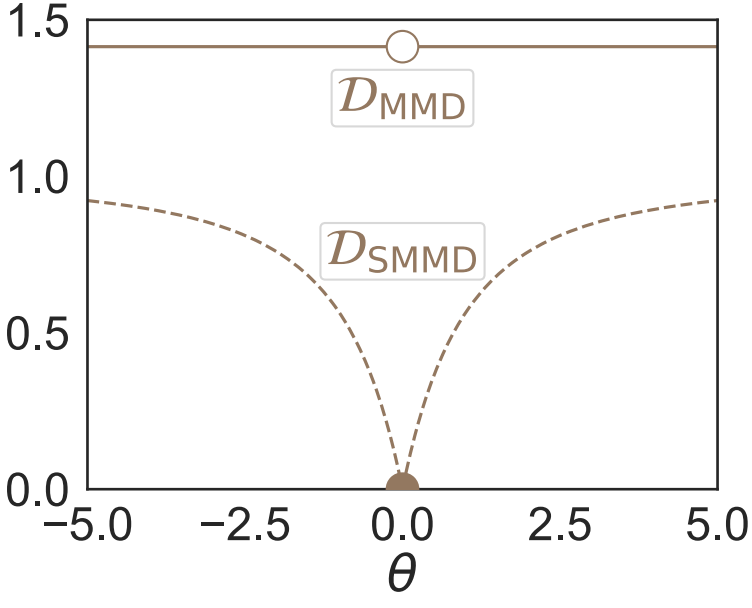


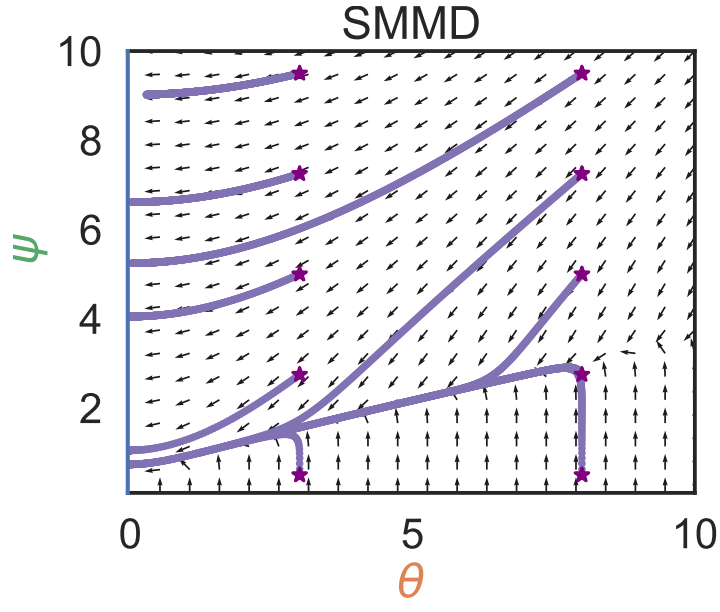
Smoothness of $\mathcal{D}_{\mathrm{SMMD}}$





Smoothness of $\mathcal{D}_{\text{SMMD}}$





Theorem: $\mathcal{D}^{\mathbb{S},\Psi,\lambda}_{\text{SMMD}}$ is continuous.

If $\mathbb{S}$ has a density; k_{top} is Gaussian/linear/...;

ϕ_ψ is fully-connected, Leaky-ReLU, non-increasing width;

all weights in Ψ have bounded condition number; then

$$\mathcal{W}(\mathbb{Q}_n, \mathbb{P}) \to 0 \text{ implies } \mathcal{D}^{\mathbb{S},\Psi,\lambda}_{\text{SMMD}}(\mathbb{Q}_n, \mathbb{P}) \to 0.$$

Results on 160×160 CelebA

SN-SMMD-GAN

KID: 0.006

WGAN-GP

KID: 0.022

Evaluating generative models

Evaluating generative models

- Human evaluation: good at precision, bad at recall
- Likelihood: hard for GANs, maybe not right thing anyway
- Two-sample tests: always reject!

Evaluating generative models

- Human evaluation: good at precision, bad at recall
- Likelihood: hard for GANs, maybe not right thing anyway
- Two-sample tests: always reject!
- Most common: Fréchet Inception Distance, FID
  - Run pretrained featurizer on model and target
  - Model each as Gaussian; compute W_2

Evaluating generative models

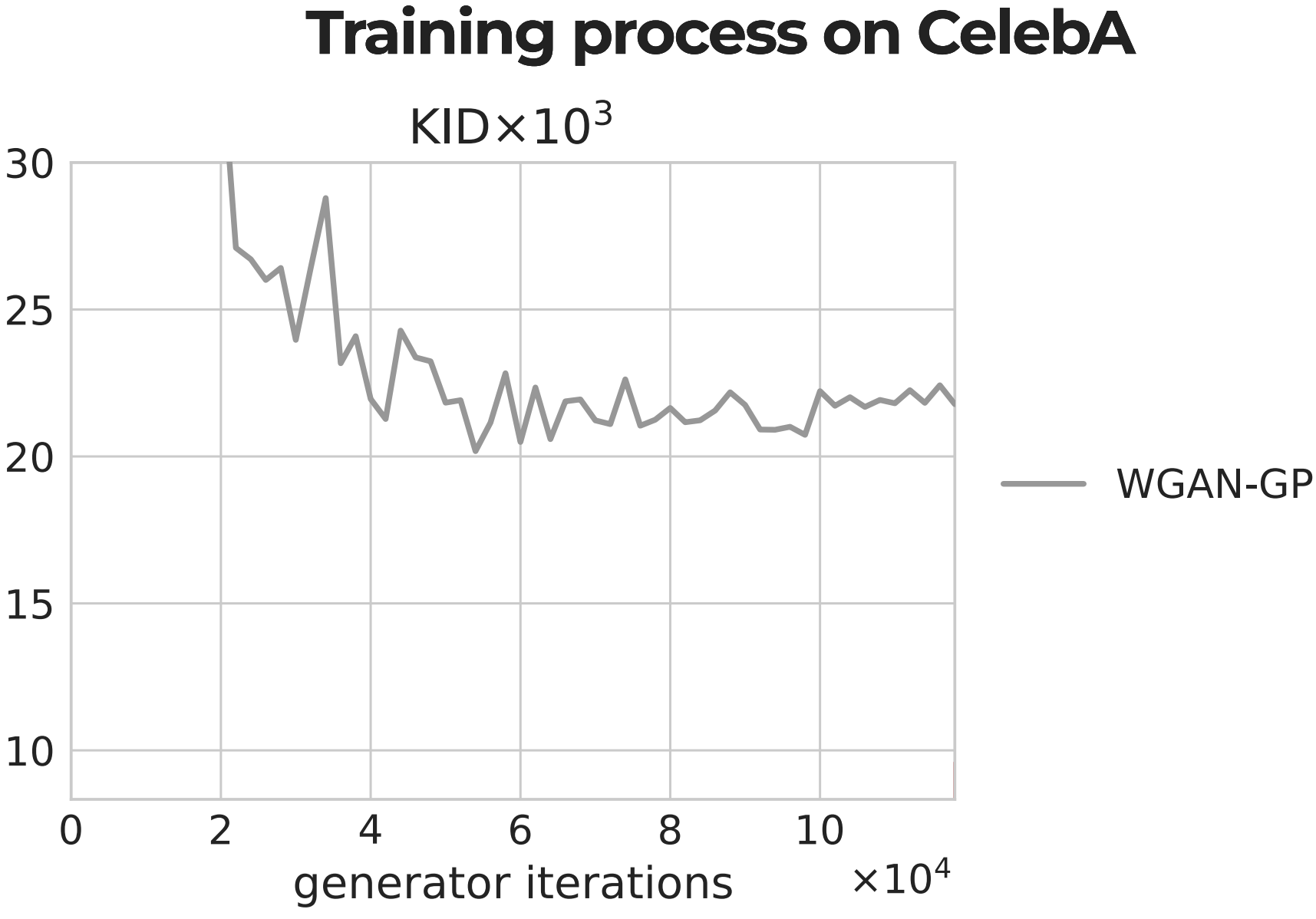
- Human evaluation: good at precision, bad at recall
- Likelihood: hard for GANs, maybe not right thing anyway
- Two-sample tests: always reject!
- Most common: Fréchet Inception Distance, FID
  - Run pretrained featurizer on model and target
  - Model each as Gaussian; compute W_2
  - Strong bias, small variance: very misleading

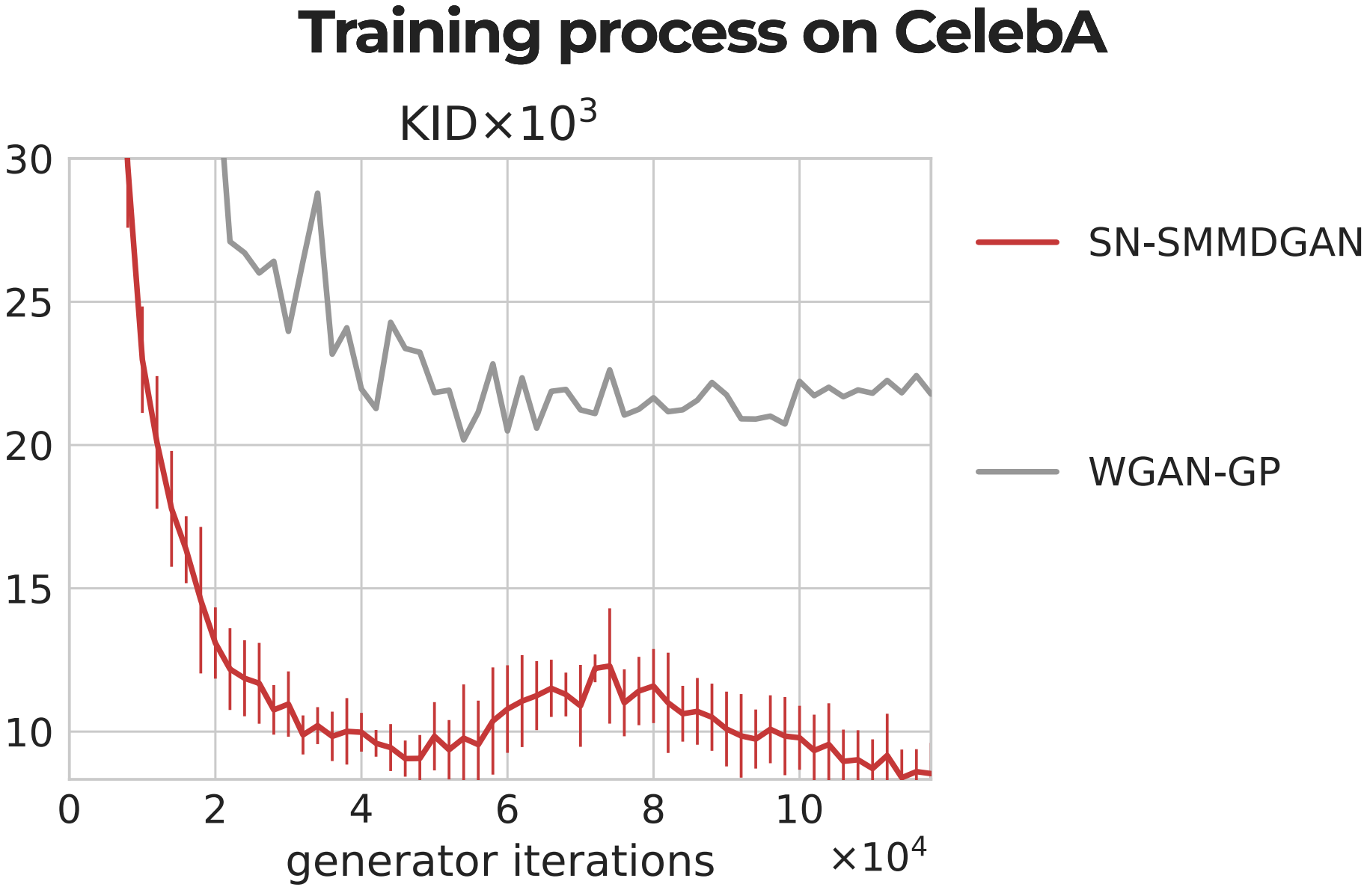
Evaluating generative models

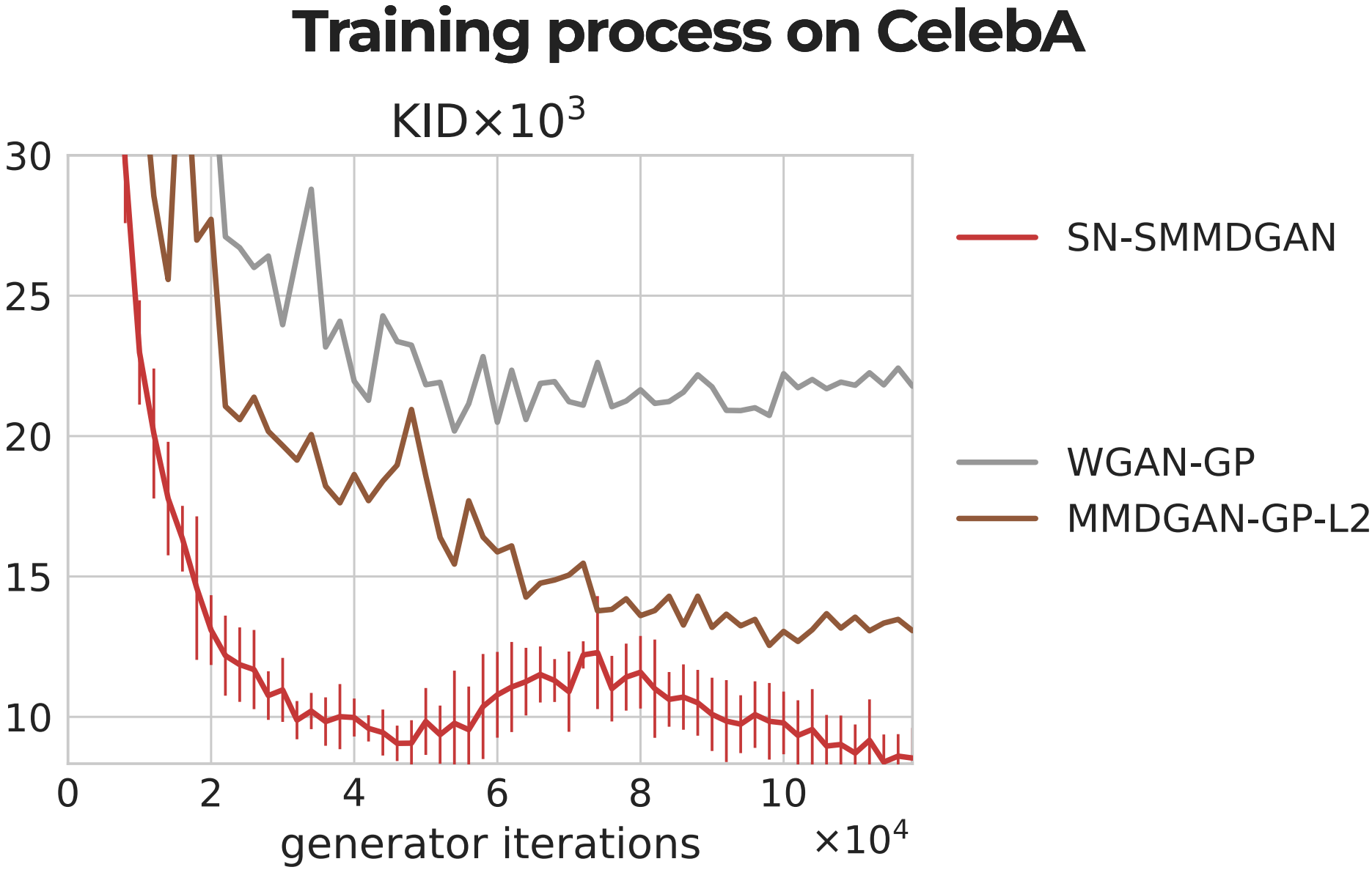
- Human evaluation: good at precision, bad at recall
- Likelihood: hard for GANs, maybe not right thing anyway
- Two-sample tests: always reject!
- Most common: Fréchet Inception Distance, FID
  - Run pretrained featurizer on model and target
  - Model each as Gaussian; compute W_2
  - Strong bias, small variance: very misleading
  - Simple examples where $\mathbf{FID}(\mathbb{Q}_1) > \mathbf{FID}(\mathbb{Q}_2)$ but $\widehat{\mathbf{FID}}(\hat{\mathbb{Q}}_1) < \widehat{\mathbf{FID}}(\hat{\mathbb{Q}}_2)$ for reasonable sample size

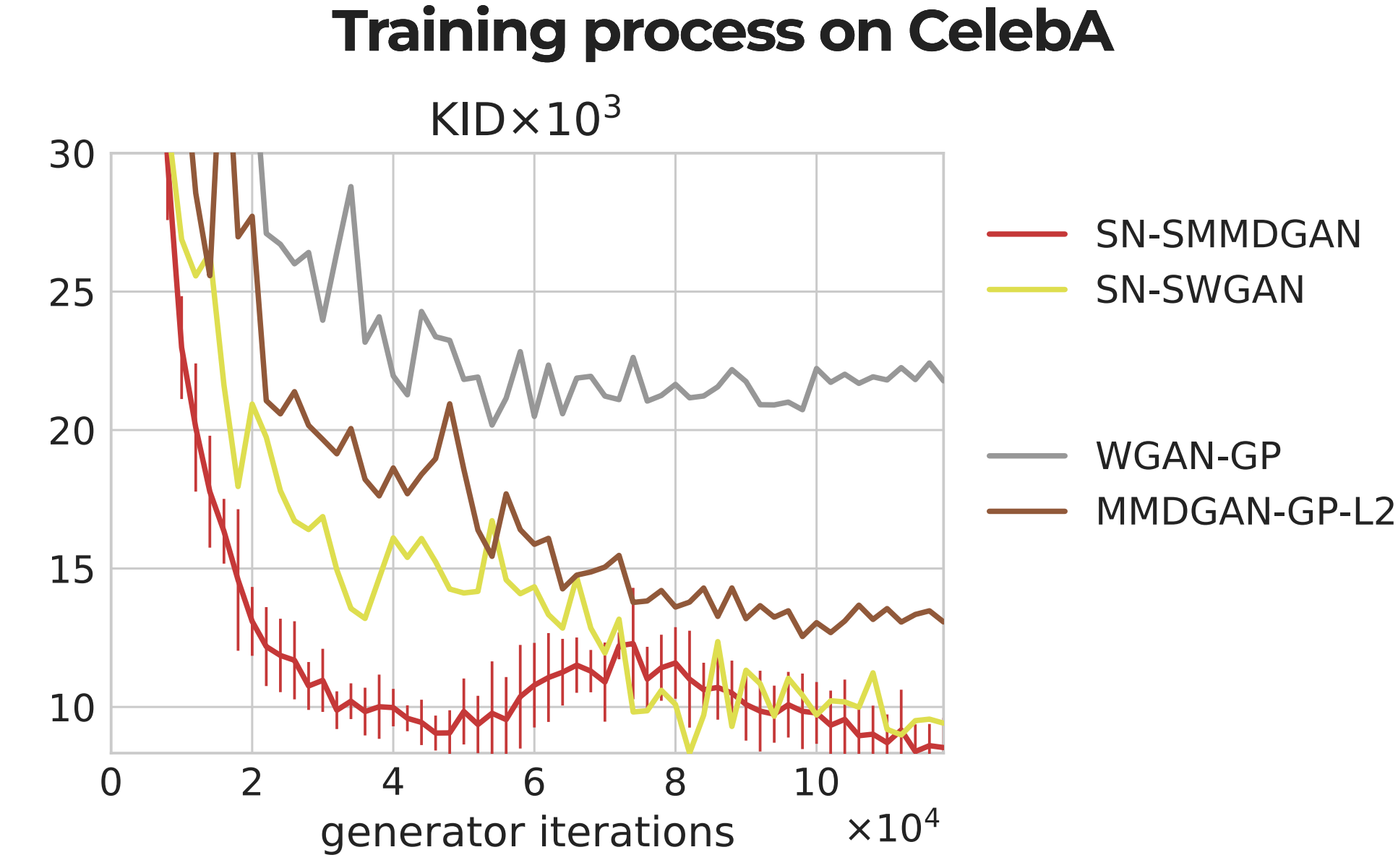
Evaluating generative models

- Human evaluation: good at precision, bad at recall
- Likelihood: hard for GANs, maybe not right thing anyway
- Two-sample tests: always reject!
- Most common: Fréchet Inception Distance, FID
  - Run pretrained featurizer on model and target
  - Model each as Gaussian; compute W_2
  - Strong bias, small variance: very misleading
  - Simple examples where $\mathbf{FID}(\mathbb{Q}_1) > \mathbf{FID}(\mathbb{Q}_2)$ but $\widehat{\mathbf{FID}}(\hat{\mathbb{Q}}_1) < \widehat{\mathbf{FID}}(\hat{\mathbb{Q}}_2)$ for reasonable sample size
- Our KID: $\mathbf{MMD}^2$ instead. Unbiased, asymptotically normal

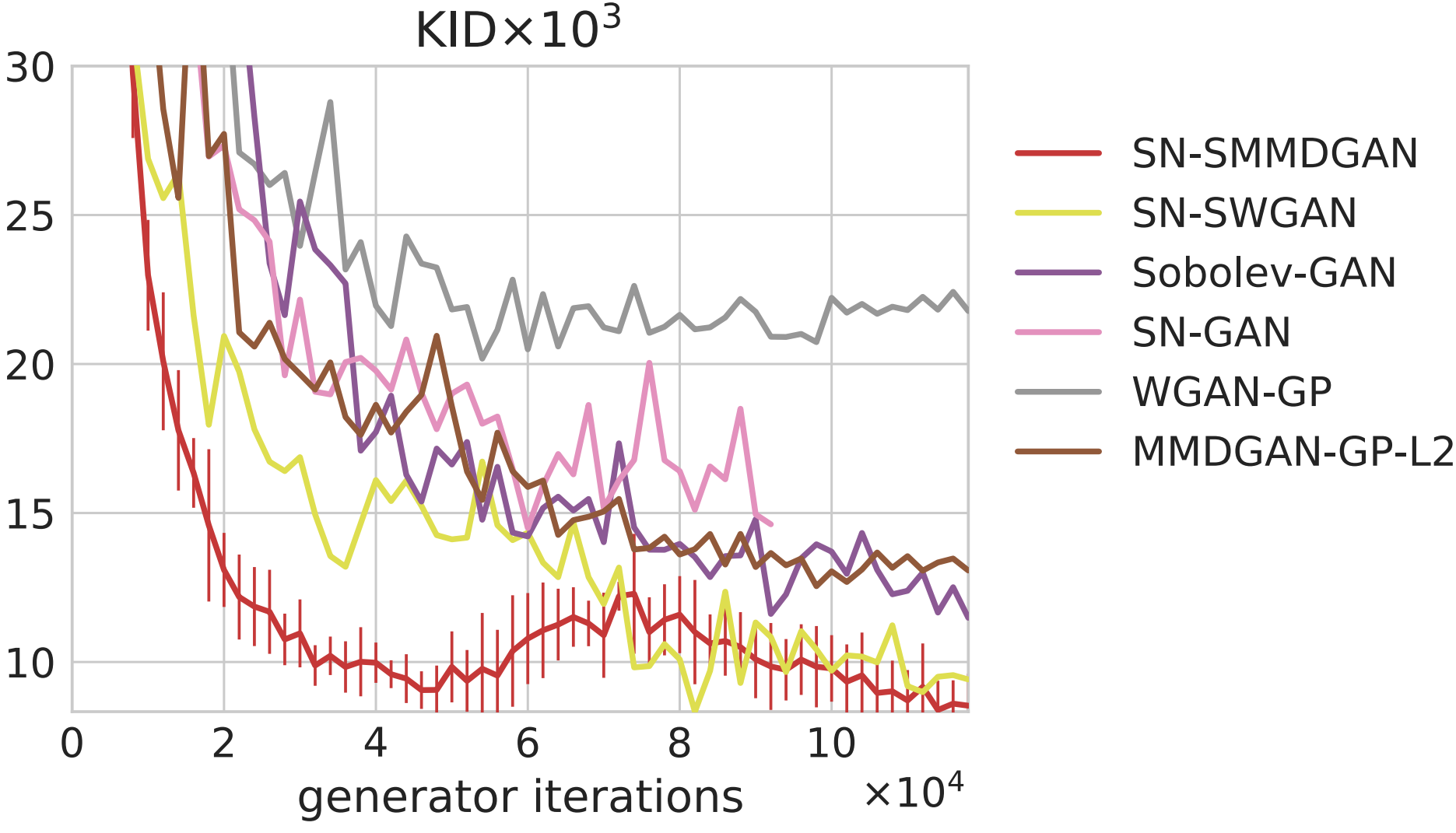

Training process on CelebA
KID×10^3
30
25
20
15
10
0
2
4
6
8
10
generator iterations
×10^4
WGAN-GP


Training process on CelebA
KID×10^3
30
25
20
15
10
0
2
4
6
8
10
generator iterations
×10^4
SN-SMMDGAN
WGAN-GP


Training process on CelebA
KID×10^3
30
25
20
15
10
0
2
4
6
8
10
generator iterations
×10^4
SN-SMMDGAN
WGAN-GP
MMDGAN-GP-L2


Training process on CelebA
KID×10^3
30
25
20
15
10
0
2
4
6
8
10
generator iterations
×10^4
SN-SMMDGAN
SN-SWGAN
WGAN-GP
MMDGAN-GP-L2

Training process on CelebA



IV: Unnormalized density/score estimation

- Problem: given samples $X_i \sim \mathbb{P}_0$ with density p_0
- Model is kernel exponential family: for any $f \in \mathcal{H}$,

$$p_f(x) = \exp(f(x))\, q(x)/Z(f)$$
$$= \exp\left(\langle \underbrace{f}_{\substack{\text{natural} \\ \text{parameter}}}, \underbrace{\phi(x)}_{\substack{\text{sufficient} \\ \text{statistic}}} \rangle_{\mathcal{H}}\right) \underbrace{q(x)}_{\substack{\text{base} \\ \text{measure}}} / \underbrace{Z(f)}_{\text{normalizer}}$$

IV: Unnormalized density/score estimation

- Problem: given samples $X_i \sim \mathbb{P}_0$ with density p_0
- Model is kernel exponential family: for any $f \in \mathcal{H}$,

$$p_f(x) = \exp(f(x))\, q(x)/Z(f)$$
$$= \exp\left(\langle \underbrace{f}_{\substack{\text{natural} \\ \text{parameter}}}, \underbrace{\phi(x)}_{\substack{\text{sufficient} \\ \text{statistic}}} \rangle_{\mathcal{H}}\right) \underbrace{q(x)}_{\substack{\text{base} \\ \text{measure}}} / \underbrace{Z(f)}_{\text{normalizer}}$$

i.e. any density with $\log p - \log q \in \mathcal{H}$

IV: Unnormalized density/score estimation

- Problem: given samples $X_i \sim \mathbb{P}_0$ with density p_0
- Model is kernel exponential family: for any $f \in \mathcal{H}$,

$$p_f(x) = \exp(f(x))\, q(x)/Z(f)$$
$$= \exp\left(\langle \underbrace{f}_{\substack{\text{natural}\\\text{parameter}}}, \underbrace{\phi(x)}_{\substack{\text{sufficient}\\\text{statistic}}} \rangle_{\mathcal{H}}\right) \underbrace{q(x)}_{\substack{\text{base}\\\text{measure}}} / \underbrace{Z(f)}_{\text{normalizer}}$$

  i.e. any density with $\log p - \log q \in \mathcal{H}$

- Gaussian k: dense in all continuous distributions on compact domains

Density estimation with KEFs

- Fitting with maximum likelihood is tough:
  - $Z(f), \nabla Z(f)$ are tough to compute
  - Likelihood equations ill-posed for *characteristic* kernels
- We choose to fit the unnormalized model
  - Could then estimate $Z(f)$ *once* after fitting if necessary

Unnormalized density / score estimation

- Don't necessarily need to compute $Z(f)$ afterwards
- $f + \log q = \log p_f + \log Z(f)$, the "energy", lets us:
  - Find modes (global or local)
  - Sample (with MCMC)
  - ...
- The score, $\nabla_x [f(x) + \log q(x)] = \nabla_x \log p_f(x)$, lets us:
  - Run HMC for targets whose gradients we can't evaluate
  - Construct Monte Carlo control functionals
  - ...

Score matching in KEFs [Sriperumbudur+ JMLR-17]

- Idea: minimize Fisher divergence $J(p_0 \| p_f)$

$$J(f) = \frac{1}{2} \int p_0(x) \| \nabla_x \log p_f(x) - \nabla_x \log p_0(x) \|^2 \mathrm{d}x$$

Score matching in KEFs [Sriperumbudur+ JMLR-17]

- Idea: minimize Fisher divergence $J(p_0 \| p_f)$

$$J(f) = \frac{1}{2} \int p_0(x) \| \nabla_x \log p_f(x) - \nabla_x \log p_0(x) \|^2 \mathrm{d}x$$

- Under mild assumptions, $J(f) = C(p_0) +$

$$\int p_0(x) \sum_{d=1}^{D} \left[\partial_d^2 \log p_f(x) + \frac{1}{2} (\partial_d \log p_f(x)))^2 \right] \mathrm{d}x$$

Score matching in KEFs [Sriperumbudur+ JMLR-17]

- Idea: minimize Fisher divergence $J(p_0 \| p_f)$

$$J(f) = \frac{1}{2} \int p_0(x) \| \nabla_x \log p_f(x) - \nabla_x \log p_0(x) \|^2 \mathrm{d}x$$

- Under mild assumptions, $J(f) = C(p_0) +$

$$\int p_0(x) \sum_{d=1}^{D} \left[\partial_d^2 \log p_f(x) + \frac{1}{2} (\partial_d \log p_f(x)))^2 \right] \mathrm{d}x$$

- Can estimate with Monte Carlo

Score matching in KEFs [Sriperumbudur+ JMLR-17]

- Minimize regularized loss function:

$$\hat{J}_\lambda(f) = \frac{1}{n}\sum_{a=1}^{n}\sum_{i=1}^{d}\left[\partial_i^2 f(X_a) + \frac{1}{2}(\partial_i f(X_a))^2\right] + \frac{1}{2}\lambda\|f\|_{\mathcal{H}}^2$$

Score matching in KEFs [Sriperumbudur+ JMLR-17]

- Minimize regularized loss function:

$$\hat{J}_\lambda(f) = \frac{1}{n}\sum_{a=1}^{n}\sum_{i=1}^{d}\left[\partial_i^2 f(X_a) + \frac{1}{2}(\partial_i f(X_a))^2\right] + \frac{1}{2}\lambda\|f\|_{\mathcal{H}}^2$$

- Representer theorem tells us minimizer of $\hat{J}_\lambda$ over $\mathcal{H}$ is

$$f_{\lambda,x} \in \mathbf{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \mathbf{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$$

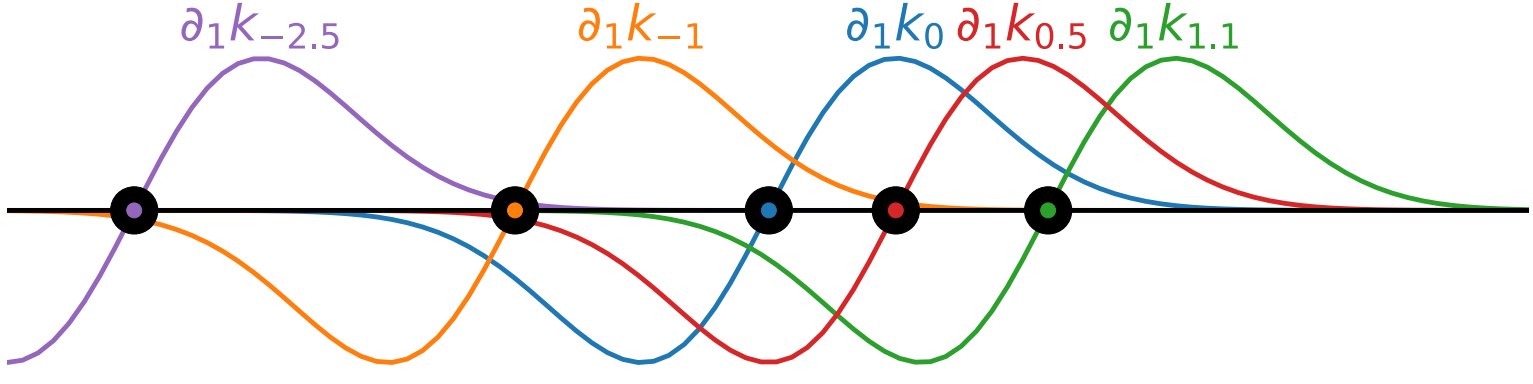
Score matching in KEFs [Sriperumbudur+ JMLR-17]

- Minimize regularized loss function:

$$\hat{J}_\lambda(f) = \frac{1}{n}\sum_{a=1}^{n}\sum_{i=1}^{d}\left[\partial_i^2 f(X_a) + \frac{1}{2}(\partial_i f(X_a))^2\right] + \frac{1}{2}\lambda\|f\|_{\mathcal{H}}^2$$

- Representer theorem tells us minimizer of $\hat{J}_\lambda$ over $\mathcal{H}$ is

$$f_{\lambda,x} \in \mathbf{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \mathbf{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$$



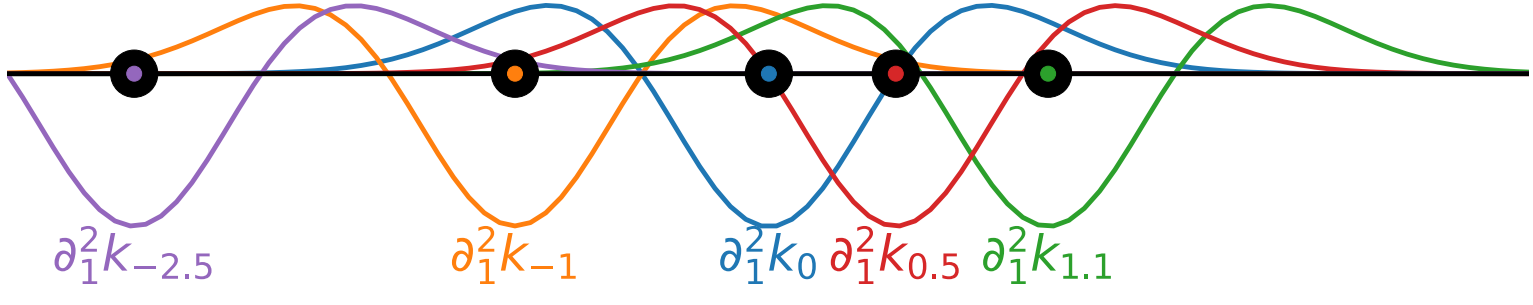
Score matching in KEFs [Sriperumbudur+ JMLR-17]

- Minimize regularized loss function:

$$\hat{J}_\lambda(f) = \frac{1}{n}\sum_{a=1}^{n}\sum_{i=1}^{d}\left[\partial_i^2 f(X_a) + \frac{1}{2}(\partial_i f(X_a))^2\right] + \frac{1}{2}\lambda\|f\|_{\mathcal{H}}^2$$

- Representer theorem tells us minimizer of $\hat{J}_\lambda$ over $\mathcal{H}$ is

$$f_{\lambda,x} \in \mathbf{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \mathbf{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$$



Score matching in a subspace

- Best $f \in \mathcal{H}$ is in

$$\mathcal{H}_{\text{full}} = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \operatorname{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$$

Score matching in a subspace

- Best $f \in \mathcal{H}$ is in

$$\mathcal{H}_{\text{full}} = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \operatorname{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$$

- Find best f in dim M subspace in $\mathcal{O}(ndM^2 + M^3)$ time

Score matching in a subspace

- Best $f \in \mathcal{H}$ is in

$$\mathcal{H}_{\text{full}} = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \operatorname{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$$

- Find best f in dim M subspace in $\mathcal{O}(ndM^2 + M^3)$ time

$$\beta = -(\frac{1}{n}\underbrace{B_{XY}^{\mathsf{T}}}_{M\times nd}\underbrace{B_{XY}}_{nd\times M} + \lambda\underbrace{G_{YY}}_{M\times M})^{\dagger}\underbrace{h_Y}_{M\times 1}$$

Score matching in a subspace

- Best $f \in \mathcal{H}$ is in

$$\mathcal{H}_{\text{full}} = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \operatorname{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$$

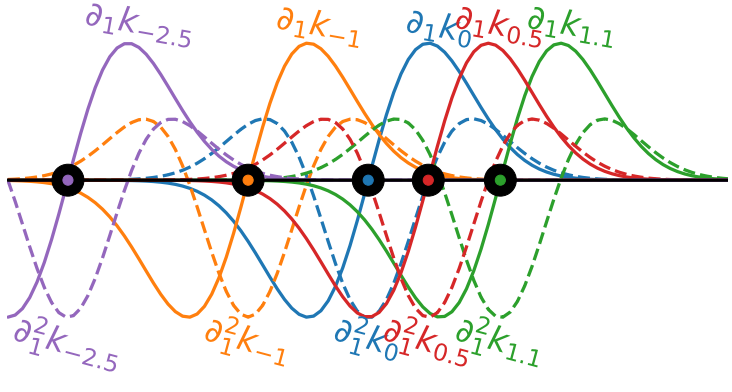
- Find best f in dim M subspace in $\mathcal{O}(ndM^2 + M^3)$ time

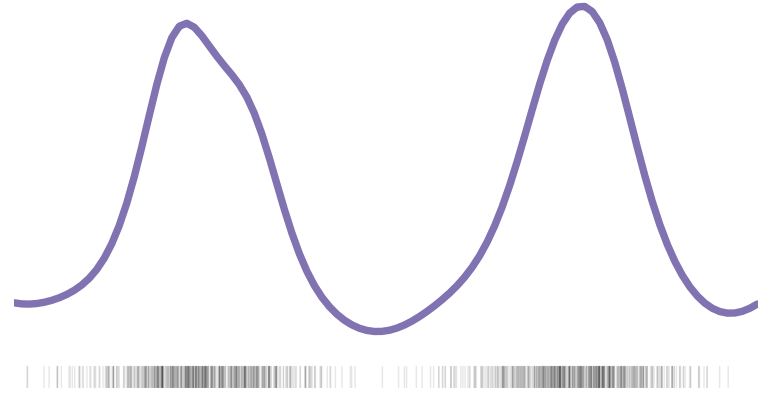
$$\beta = -(\tfrac{1}{n}\underbrace{B_{XY}^{\mathsf{T}}}_{M\times nd}\underbrace{B_{XY}}_{nd\times M} + \lambda\underbrace{G_{YY}}_{M\times M})^{\dagger}\underbrace{h_Y}_{M\times 1}$$

  - $\mathcal{H}_{\text{full}}$: $M = 2nd$, $\mathcal{O}(n^3d^3)$ time!

Nyström approximation [Sutherland+ AISTATS-18]

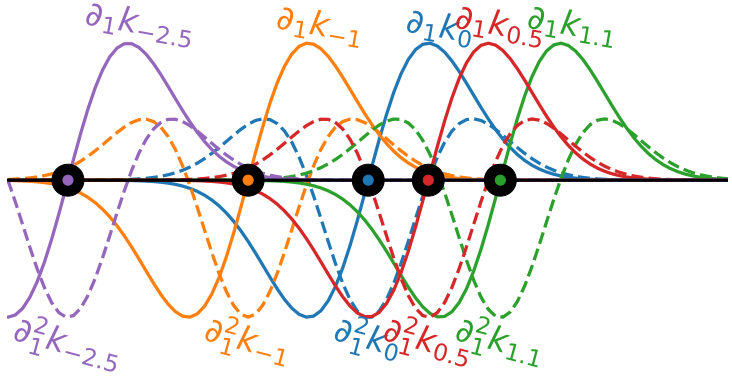
- $\mathcal{H}_{\text{full}} = \text{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \text{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$

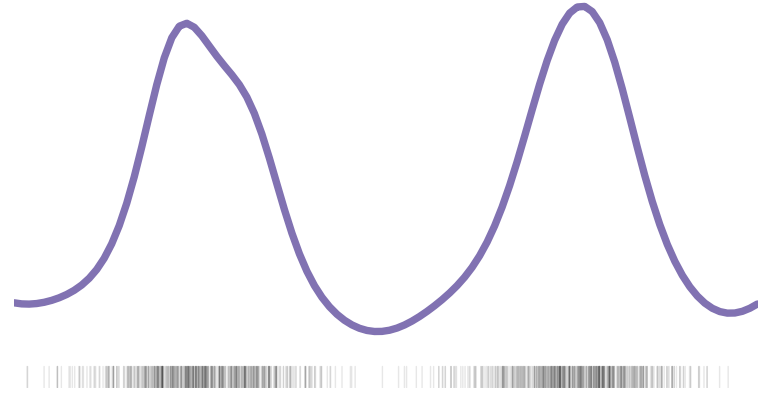




Nyström approximation [Sutherland+ AISTATS-18]

- $\mathcal{H}_{\text{full}} = \text{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \text{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$
- Nyström approximation: find fit in different (smaller) $\mathcal{H}_Y$



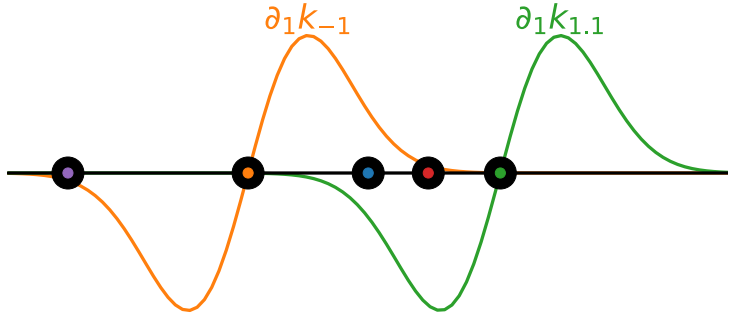


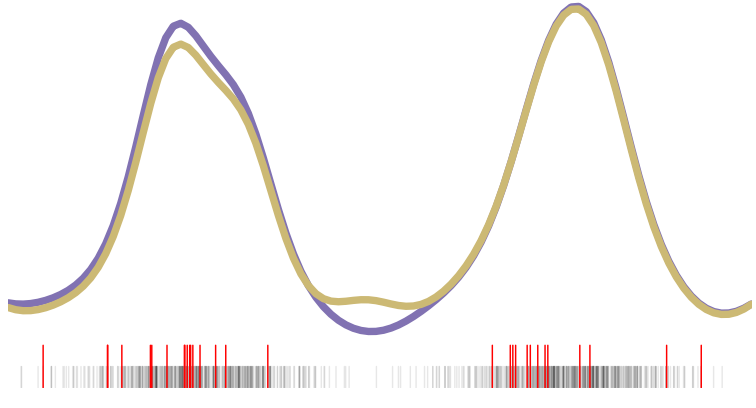
Nyström approximation [Sutherland+ AISTATS-18]

- $\mathcal{H}_{\text{full}} = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \operatorname{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$
- Nyström approximation: find fit in different (smaller) $\mathcal{H}_Y$
- One choice: pick $Y \subset [n]$, $|Y| = m$ at random, then

  $\mathcal{H}_{\text{nys}}^{Y} = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in Y}^{i\in[d]}$ $\qquad \mathcal{O}(nm^2d^3)$ time

  Get the same rates with $m = \sqrt{n}\log n$ (sometimes less)





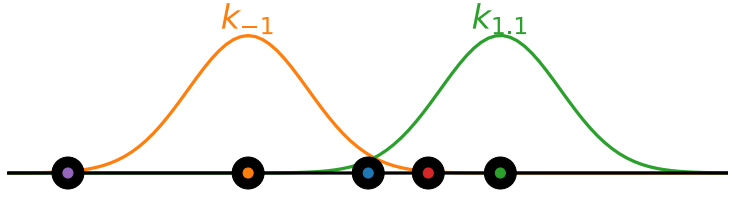
Nyström approximation [Sutherland+ AISTATS-18]

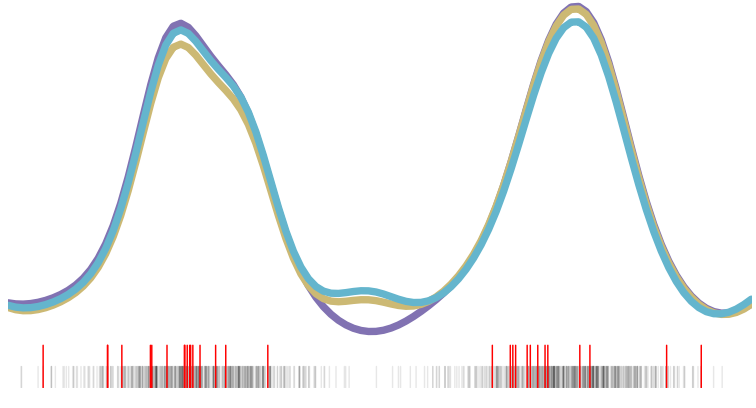
- $\mathcal{H}_{\text{full}} = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in[n]}^{i\in[d]} \cup \operatorname{span}\left\{\partial_i^2 k_{X_a}\right\}_{a\in[n]}^{i\in[d]}$
- Nyström approximation: find fit in different (smaller) $\mathcal{H}_Y$
- One choice: pick $Y \subset [n]$, $|Y| = m$ at random, then

  $\mathcal{H}_{\text{nys}}^Y = \operatorname{span}\left\{\partial_i k_{X_a}\right\}_{a\in Y}^{i\in[d]}$ $\quad \mathcal{O}(nm^2d^3)$ time

  Get the same rates with $m = \sqrt{n}\log n$ (sometimes less)
- *"lite"*: pick Y at random, then

  $\mathcal{H}_{\text{lite}}^Y = \operatorname{span}\left\{k_{X_a}\right\}_{a\in Y}$ $\quad \mathcal{O}(nm^2d)$ time





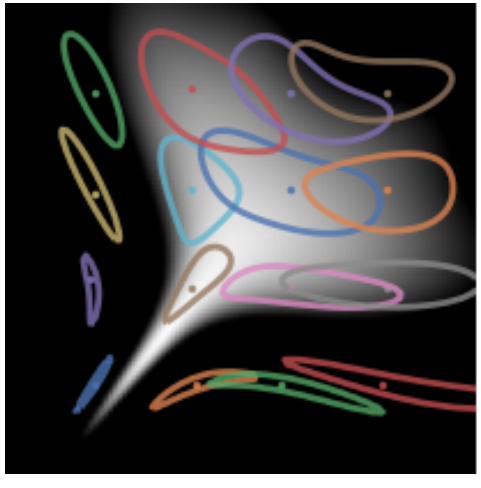
Meta-learning a kernel

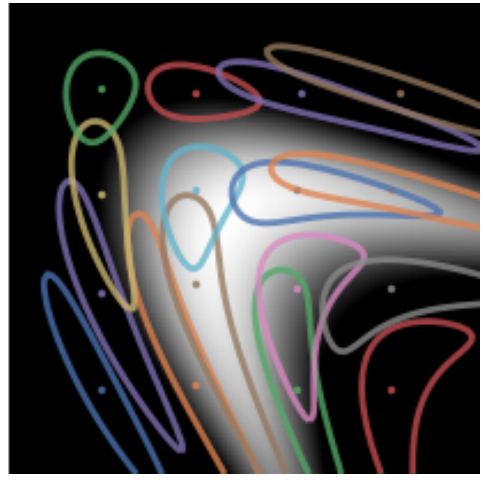
Meta-learning a kernel

Meta-learning a kernel

Results

- Learns local dataset geometry: better fits

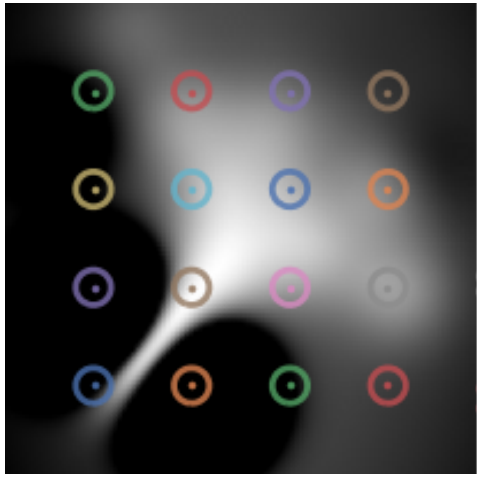
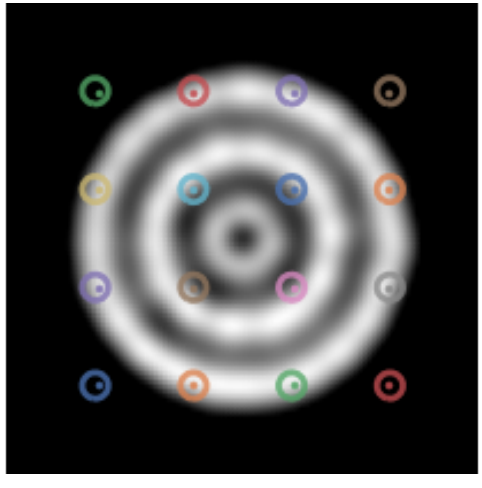




- On real data: slightly worse likelihoods, maybe better “shapes” than deep likelihood models

Results

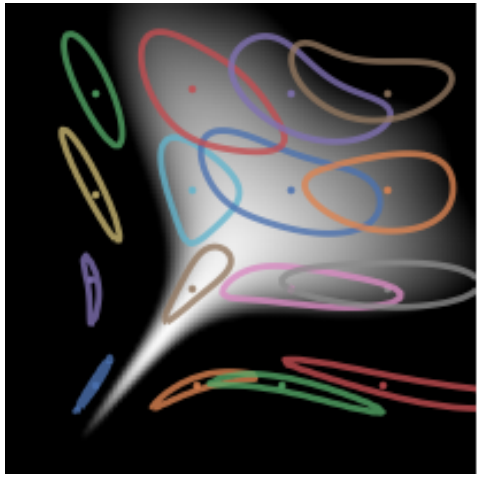
- Learns local dataset geometry: better fits

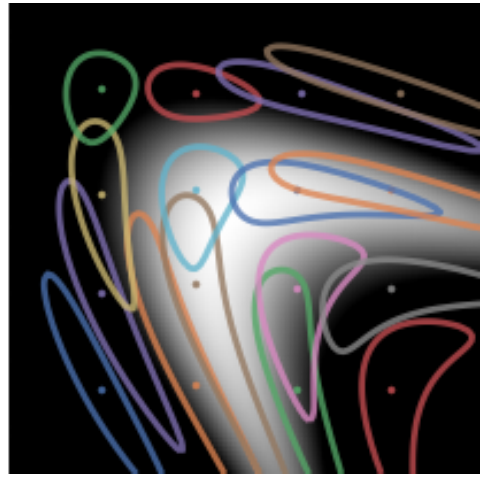




- On real data: slightly worse likelihoods, maybe better “shapes” than deep likelihood models

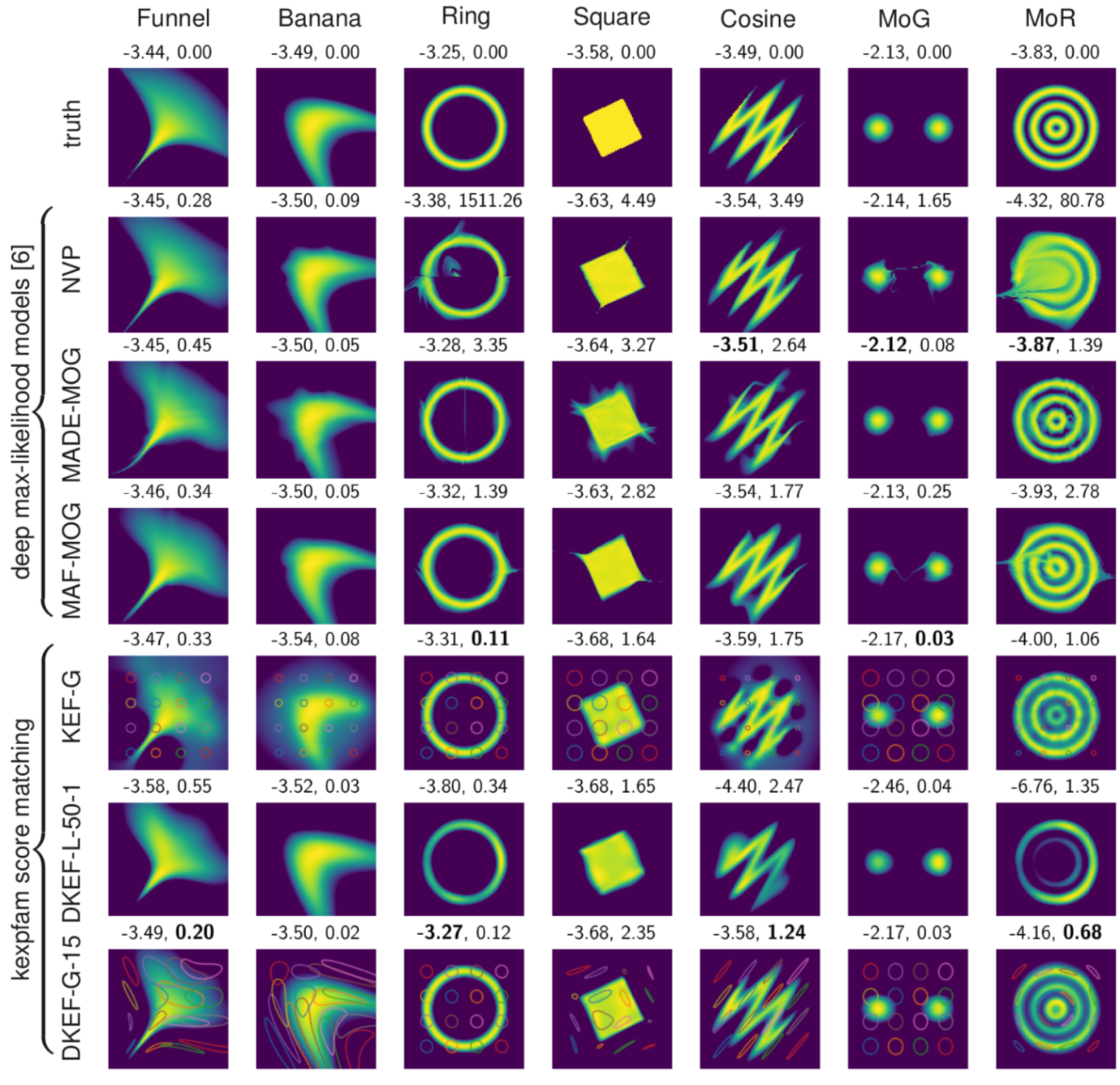
Results

- Learns local dataset geometry: better fits





- On real data: slightly worse likelihoods, maybe better “shapes” than deep likelihood models


Funnel
Banana
Ring
Square
Cosine
MoG
MoR
truth
-3.44, 0.00
-3.49, 0.00
-3.25, 0.00
-3.58, 0.00
-3.49, 0.00
-2.13, 0.00
-3.83, 0.00
deep max-likelihood models [6]
NVP
-3.45, 0.28
-3.50, 0.09
-3.38, 1511.26
-3.63, 4.49
-3.54, 3.49
-2.14, 1.65
-4.32, 80.78
MADE-MOG
-3.45, 0.45
-3.50, 0.05
-3.28, 3.35
-3.64, 3.27
-3.51, 2.64
-2.12, 0.08
-3.87, 1.39
MAF-MOG
-3.46, 0.34
-3.50, 0.05
-3.32, 1.39
-3.63, 2.82
-3.54, 1.77
-2.13, 0.25
-3.93, 2.78
kexpfam score matching
KEF-G
-3.47, 0.33
-3.54, 0.08
-3.31, 0.11
-3.68, 1.64
-3.59, 1.75
-2.17, 0.03
-4.00, 1.06
DKEF-L-50-1
-3.58, 0.55
-3.52, 0.03
-3.80, 0.34
-3.68, 1.65
-4.40, 2.47
-2.46, 0.04
-6.76, 1.35
DKEF-G-15
-3.49, 0.20
-3.50, 0.02
-3.27, 0.12
-3.68, 2.35
-3.58, 1.24
-2.17, 0.03
-4.16, 0.68

Recap

Combining a deep architecture with a kernel machine that takes the higher-level learned representation as input can be quite powerful.

— Y. Bengio & Y. LeCun (2007), "Scaling Learning Algorithms towards AI"

Recap

Combining a deep architecture with a kernel machine that takes the higher-level learned representation as input can be quite powerful.

— Y. Bengio & Y. LeCun (2007), "Scaling Learning Algorithms towards AI"

- Two-sample testing [ICLR-17, ICML-20, NeurIPS-21]
  - ψ maximizing power criterion, for one task or many
- Self-supervised learning with HSIC [NeurIPS-21]
  - Much better understanding of what's going on!
- Generative modeling with MMD GANs [ICLR-18, NeurIPS-18]
  - Need a smooth loss function for the generator
- Score matching in exponential families [AISTATS-18, ICML-19]
  - Avoid overfitting with closed-form fit on held-out data