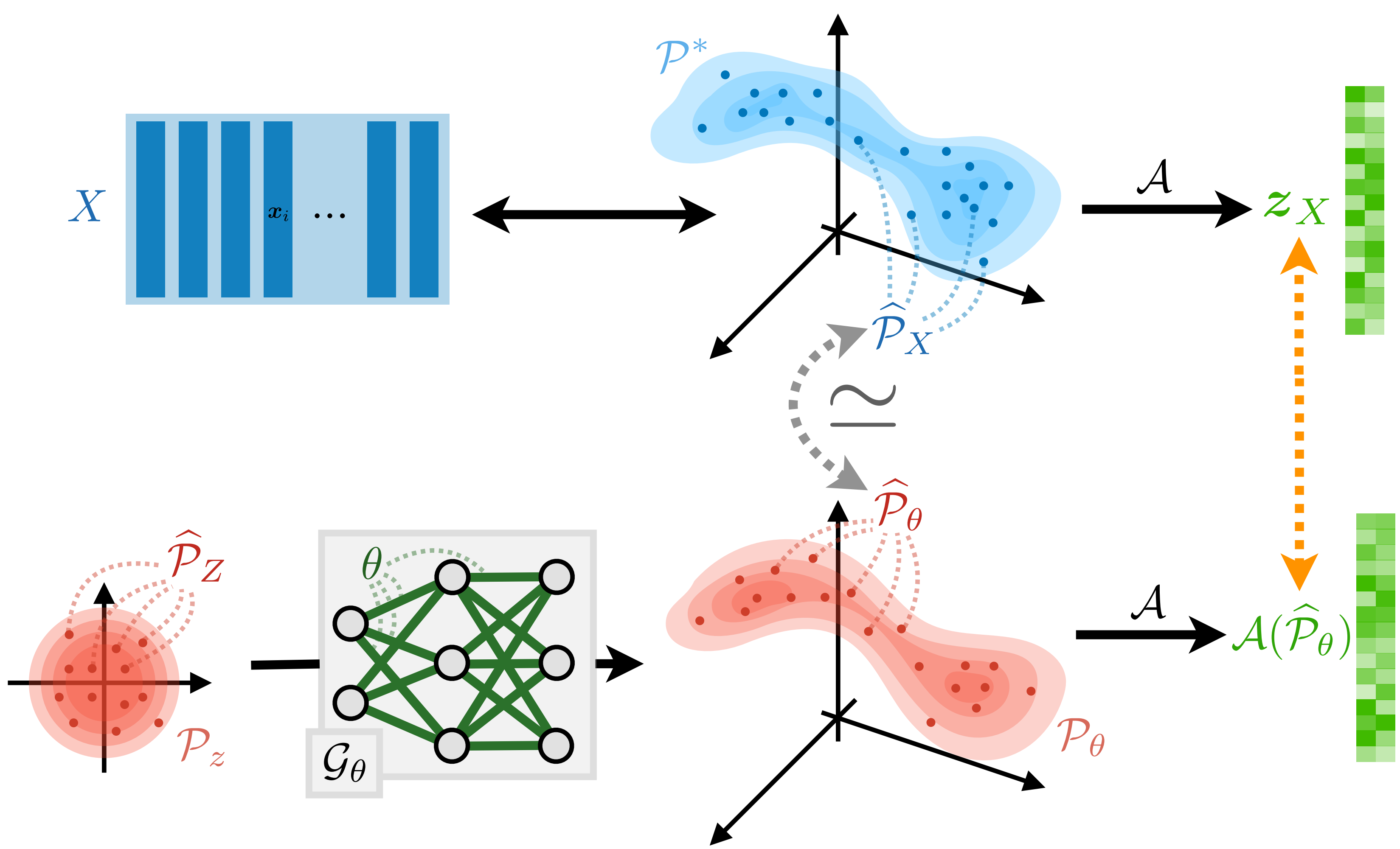
Compressive Learning of Generative Networks

Vincent Schellekens & Laurent Jacques
UCLouvain

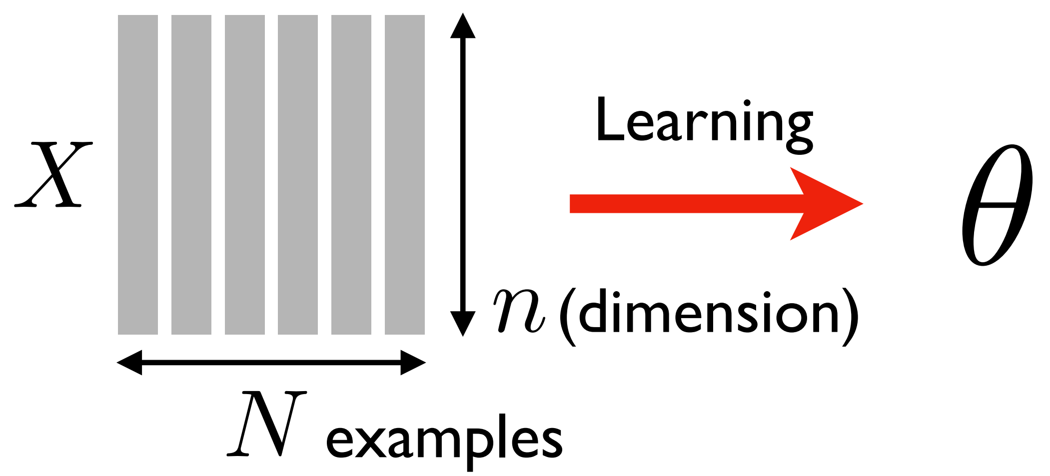


UCLouvain icteam fnrs LA LIBERTÉ DE CHERCHER

Compressive Learning of Generative Networks

Compressive Learning: Why?

Usual machine learning

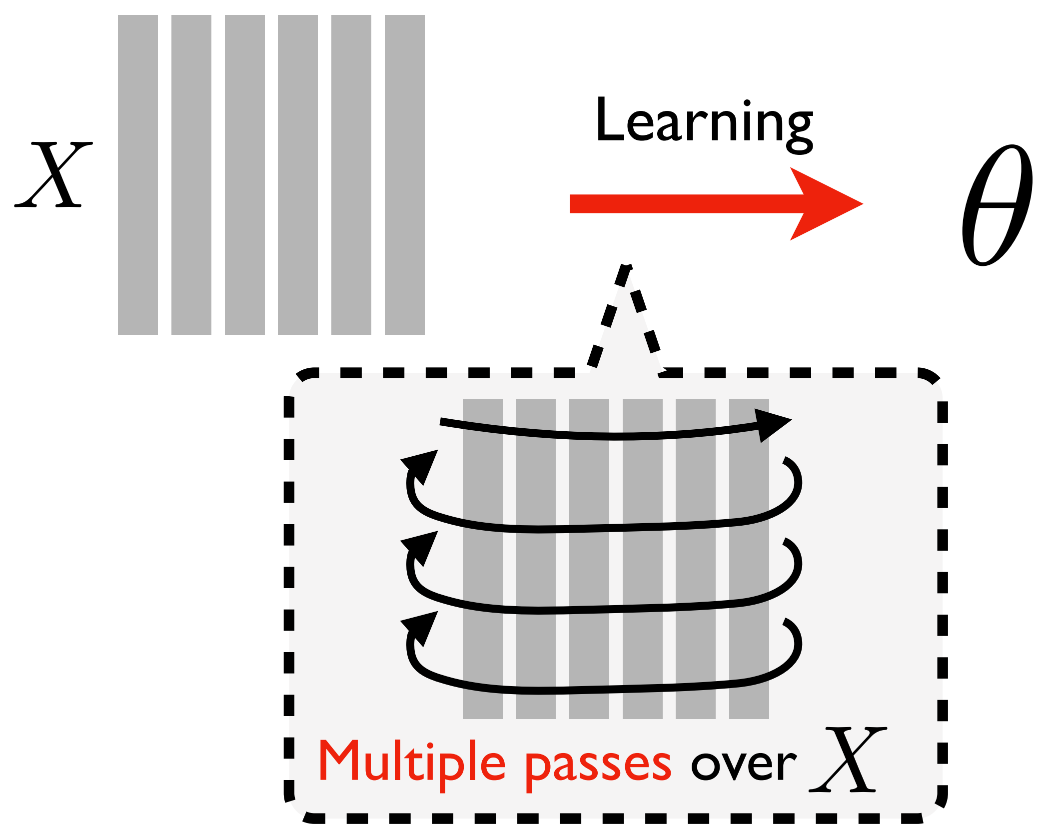


Dataset

Parameters

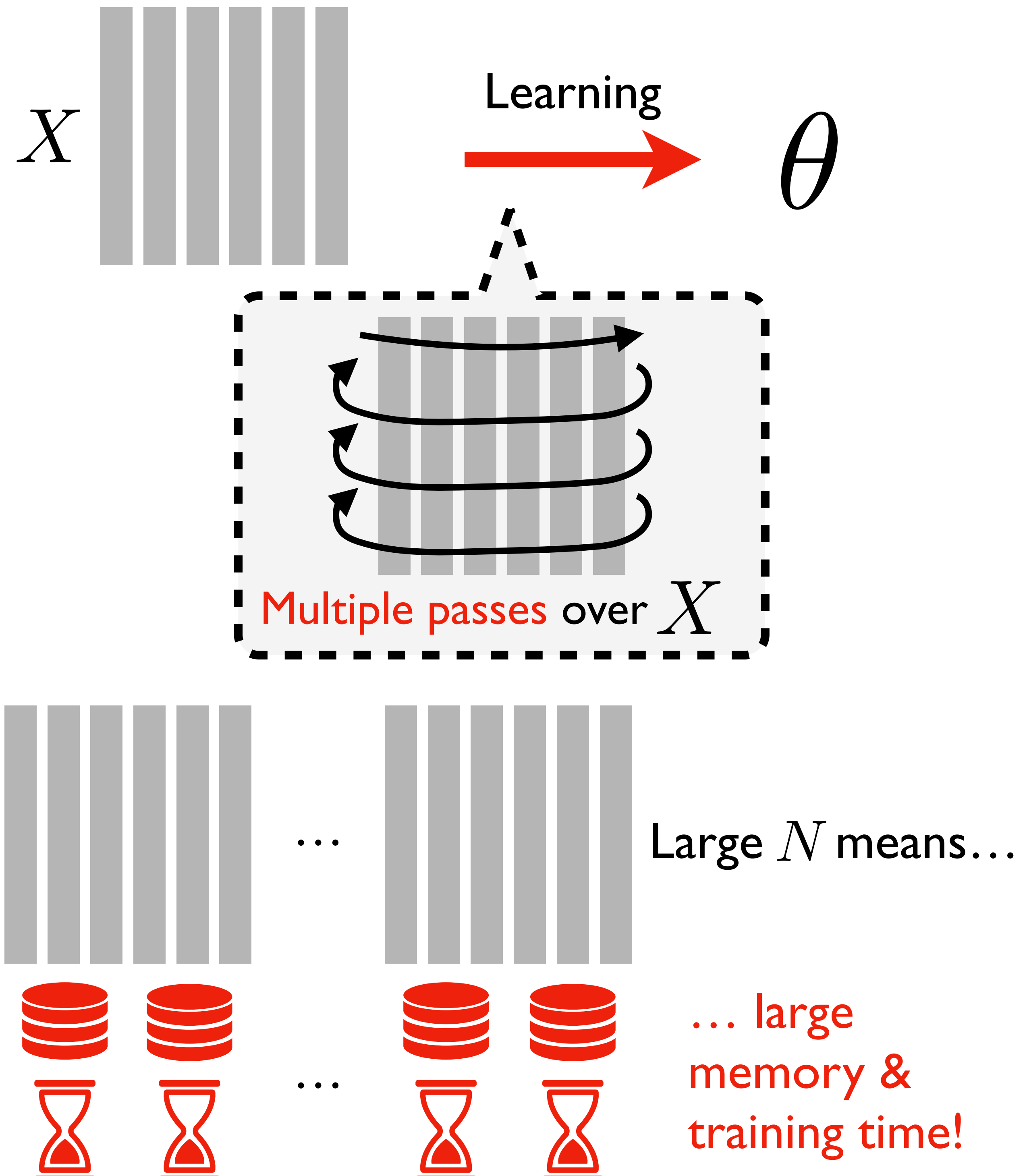
Compressive Learning: Why?

Usual machine learning



Compressive Learning: Why?

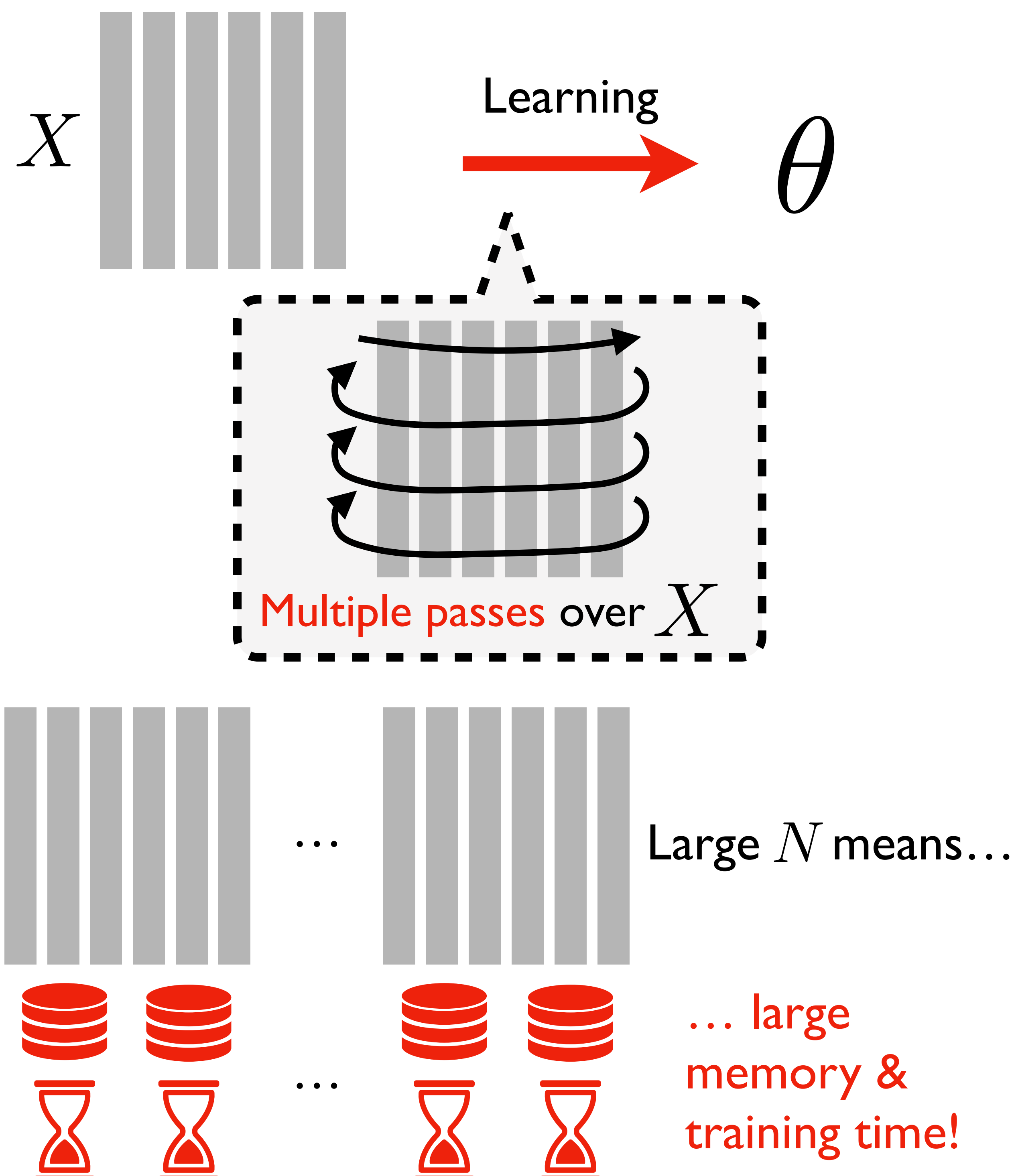
Usual machine learning



Context: <u>large-scale learning</u>

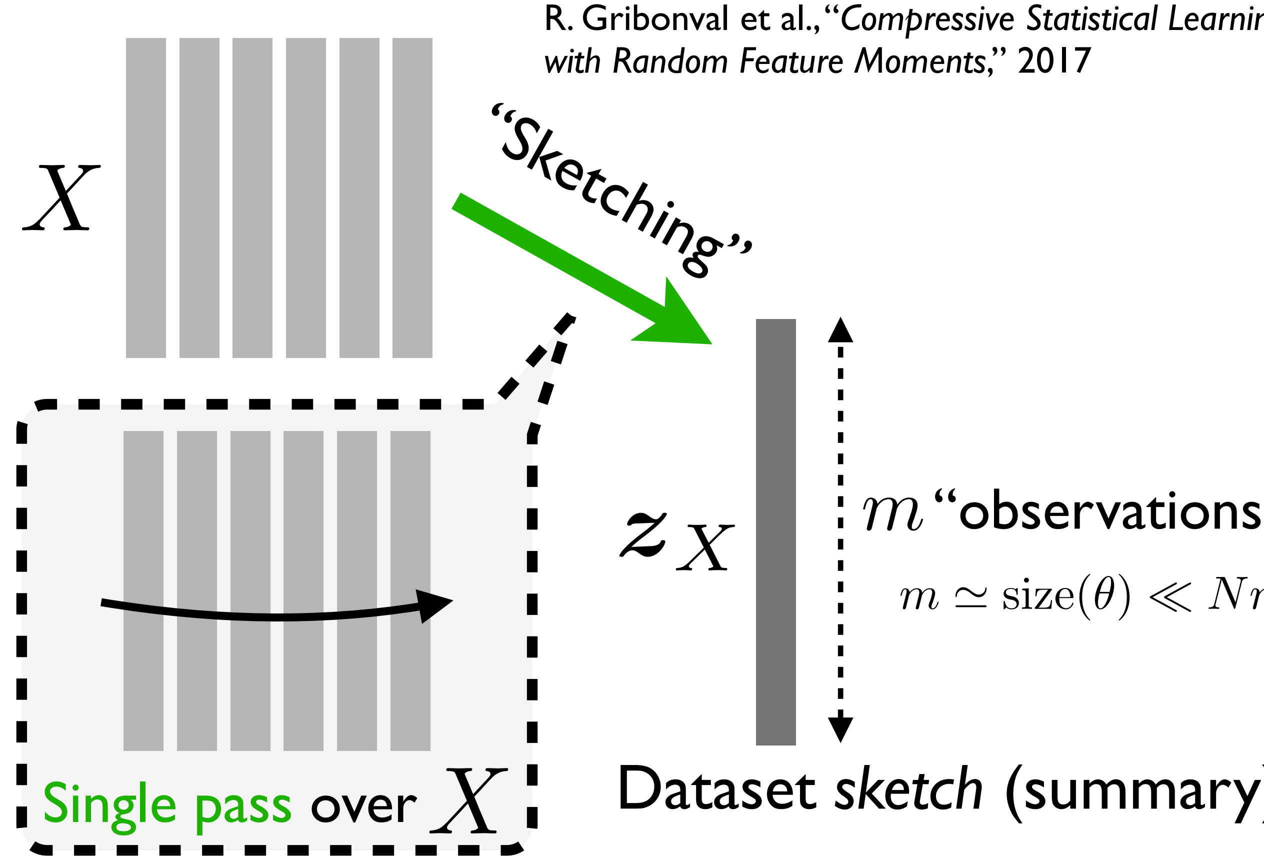
Compressive Learning: What?

Usual machine learning



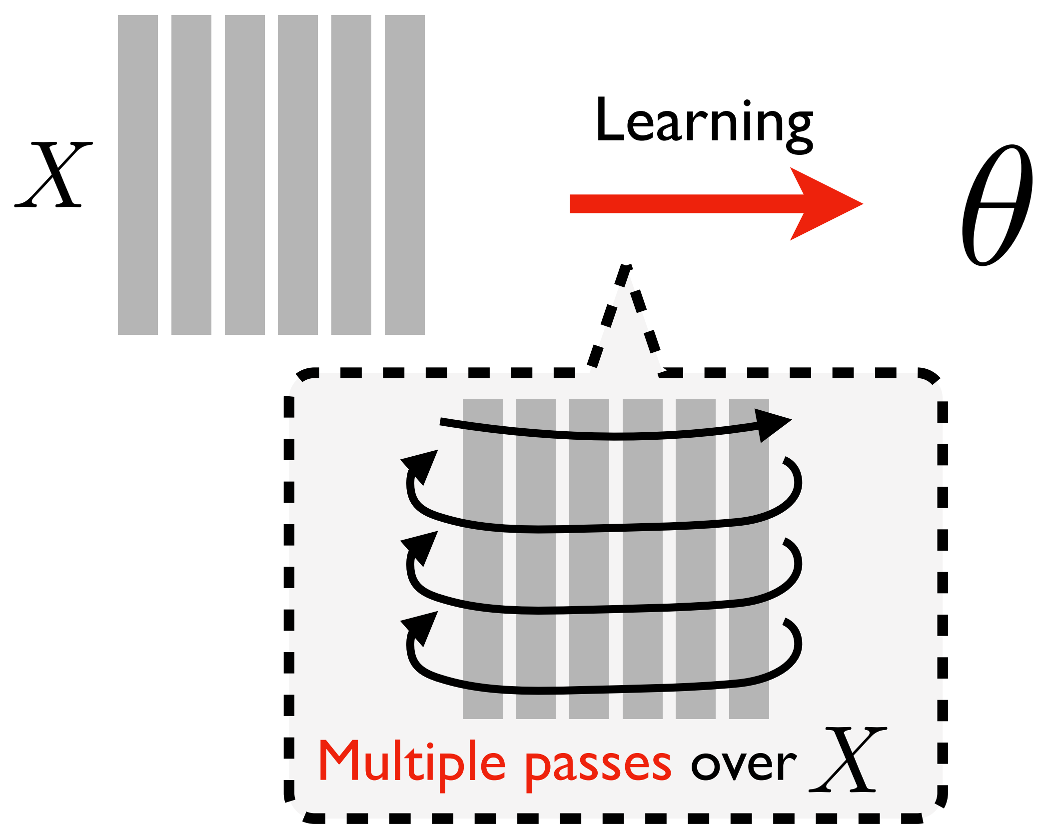
Compressive Learning

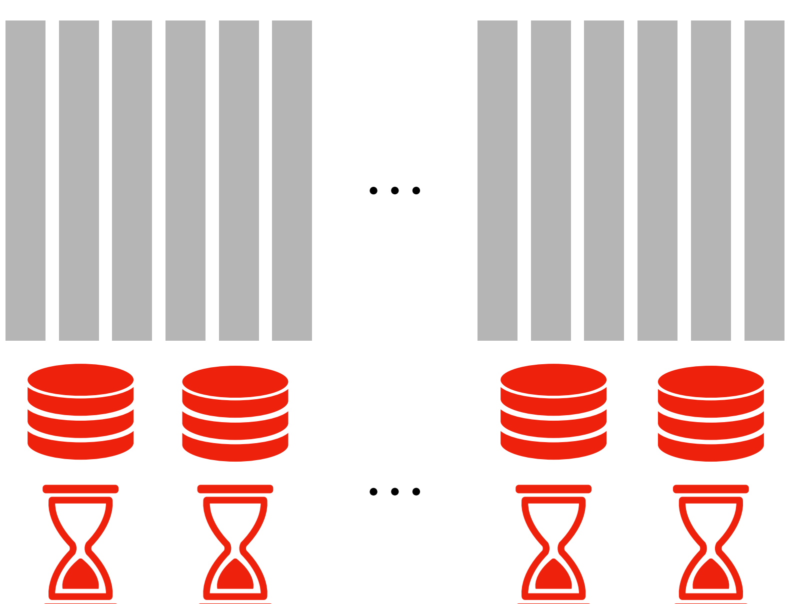
R. Gribonval et al., "*Compressive Statistical Learning with Random Feature Moments*," 2017



Compressive Learning: What?

Usual machine learning



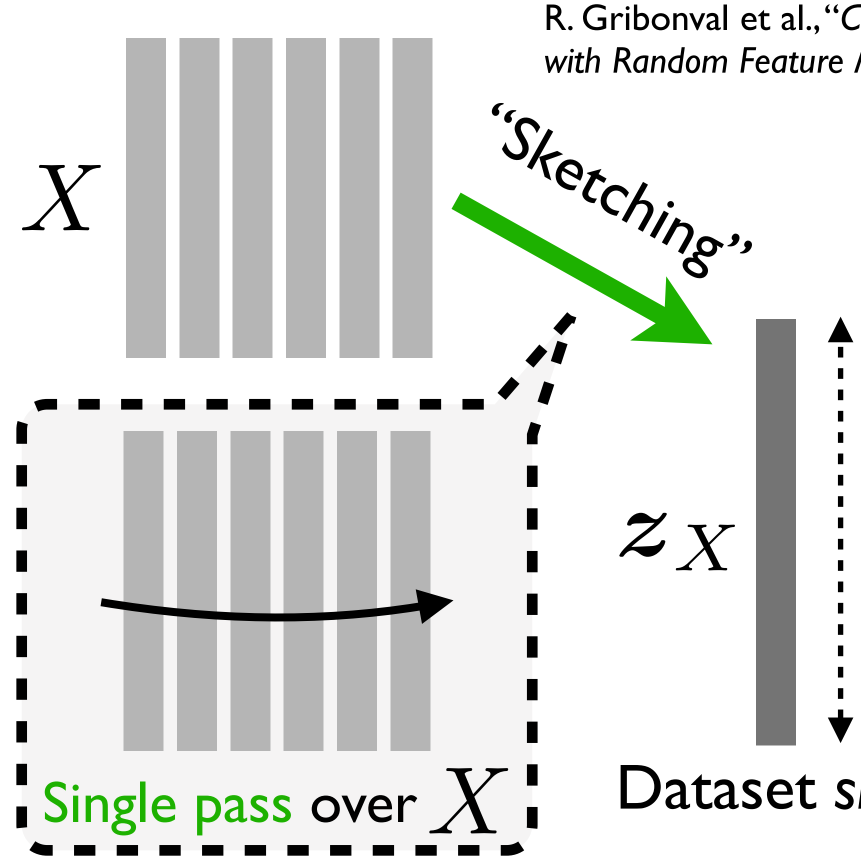


Large N means…

… large memory & training time!

Compressive Learning

R. Gribonval et al., "*Compressive Statistical Learning with Random Feature Moments*," 2017

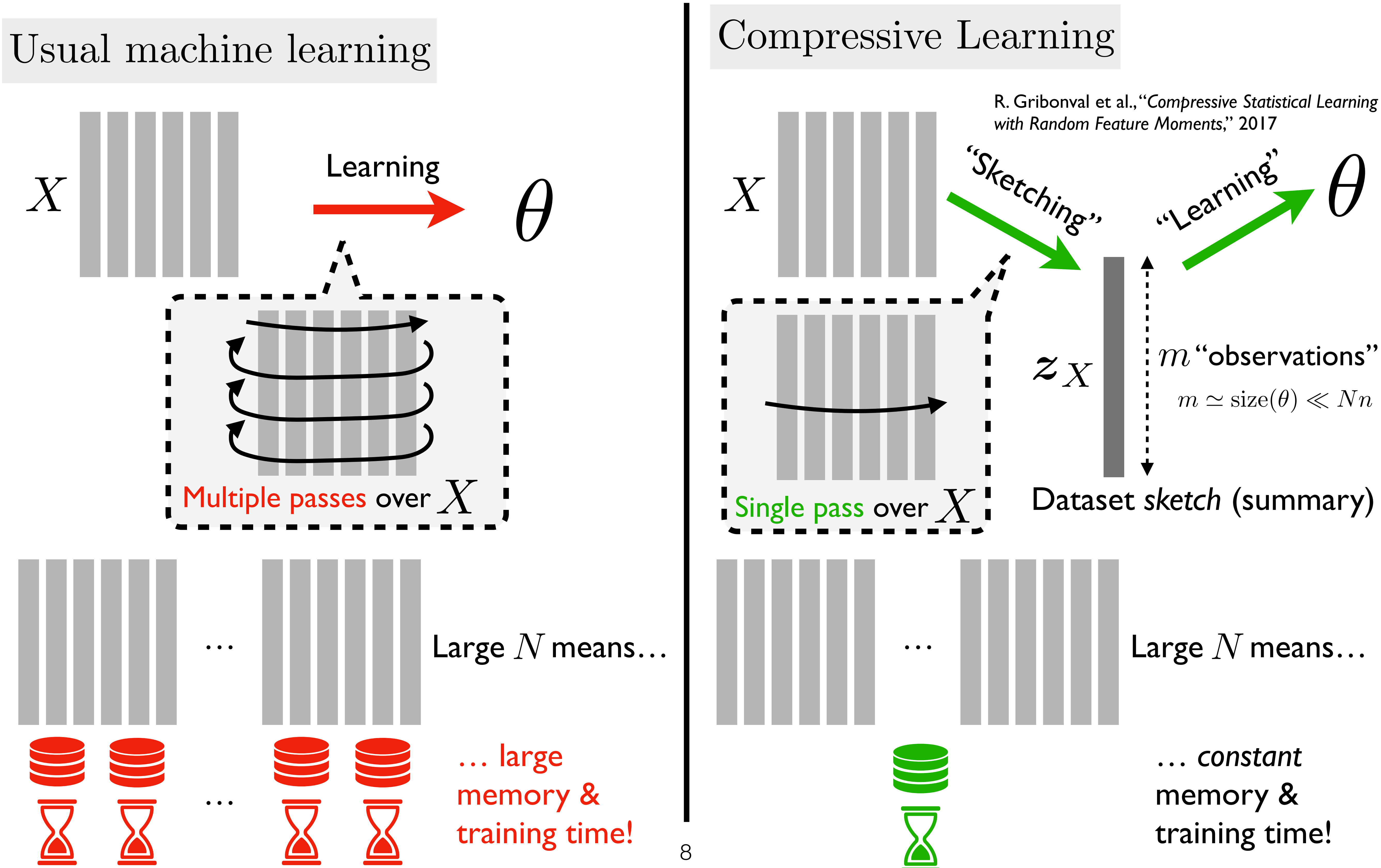


Dataset *sketch* (summary)

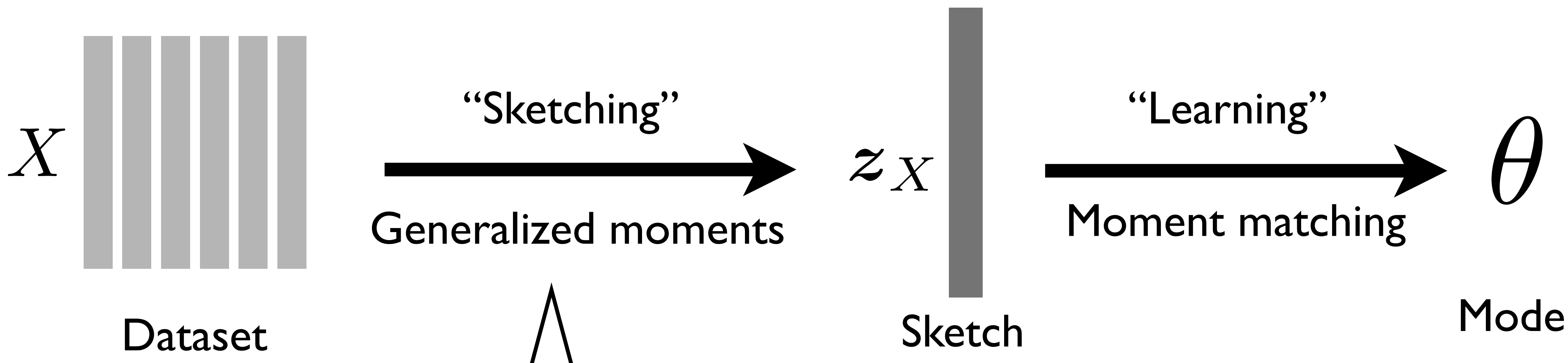
$$\boldsymbol{z}_X = \frac{1}{N}\sum_{i=1}^{N} \Phi(\boldsymbol{x}_i)$$

"Generalized moments"
(average of features of the examples)

Compressive Learning: What?



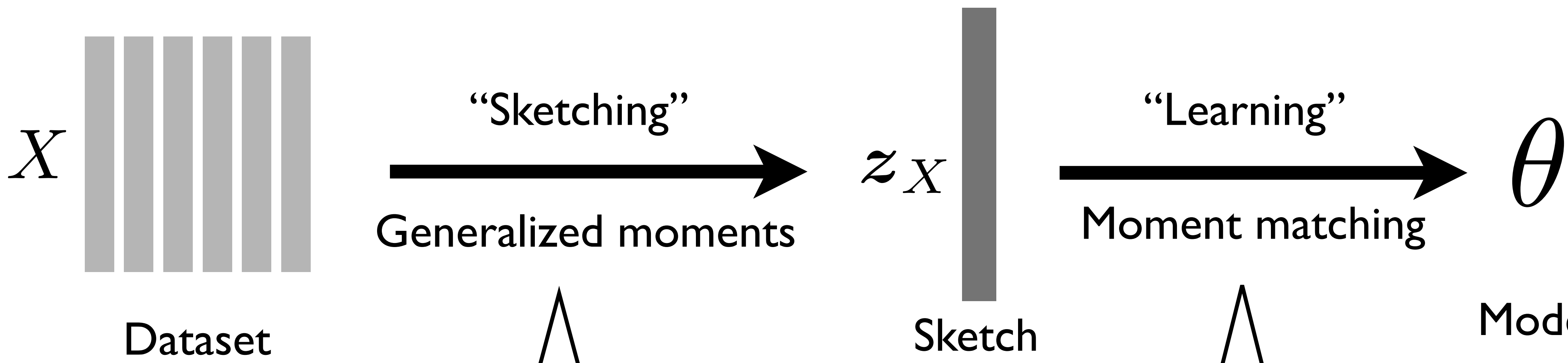
Compressive Learning in a nutshell



Advantages:

- **Harsh compression** (ideal for large-scale datasets, data streams)
- Only a **single pass on dataset** (parallelizable, distributable)
- Handy for **privacy preservation**

Compressive Learning in a nutshell

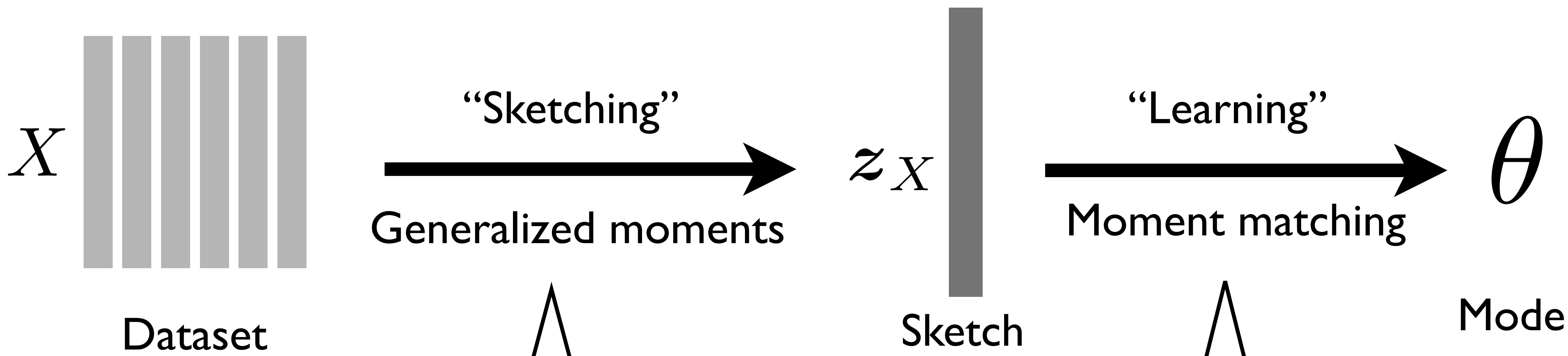


Advantages:
- **Harsh compression** (ideal for large-scale datasets, data streams)
- Only a **single pass on dataset** (parallelizable, distributable)
- Handy for **privacy preservation**

Existing algorithms for:
- k-means clustering
- mixture model estimation: GMM, alpha-stable distributions
- Principal Component Analysis
- Independent Component Analysis

Difficult to extend the framework to new tasks!

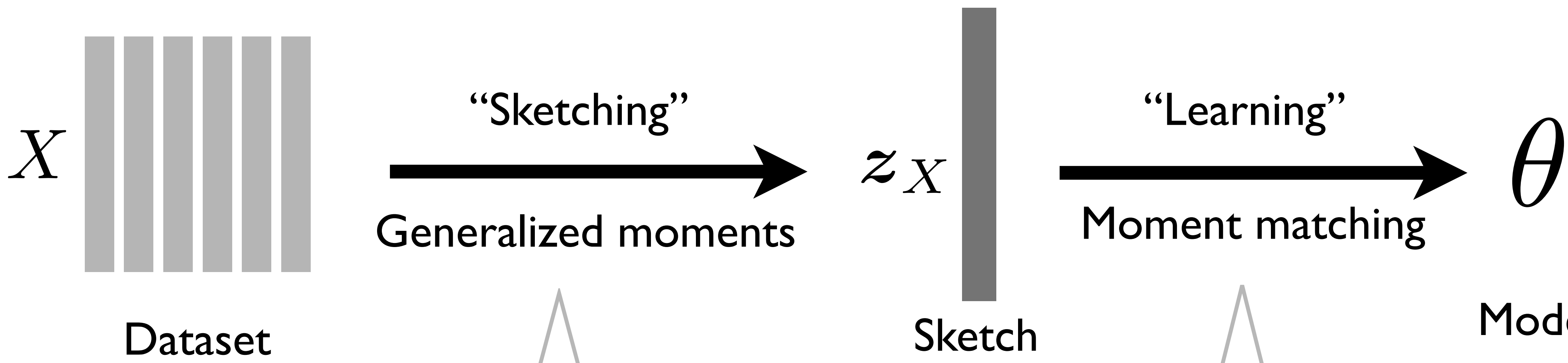
Compressive Learning in a nutshell



Advantages:
- **Harsh compression** (ideal for large-scale datasets, data streams)
- Only a **single pass on dataset** (parallelizable, distributable)
- Handy for **privacy preservation**

Existing algorithms for:
- k-means clustering
- mixture model estimation: GMM, alpha-stable distributions
- Principal Component Analysis
- Independent Component Analysis
- **This work: Generative NN**

Compressive Learning in a nutshell



Advantages:

- **Harsh compression** (ideal for large-scale datasets, data streams)
- Only a **single pass on dataset** (parallelizable, distributable)
- Handy for **privacy preservation**

Existing algorithms for:

- k-means clustering
- mixture model estimation: GMM, alpha-stable distributions
- Principal Component Analysis
- Independent Component Analysis
- **This work: Generative NN**

Contributions:

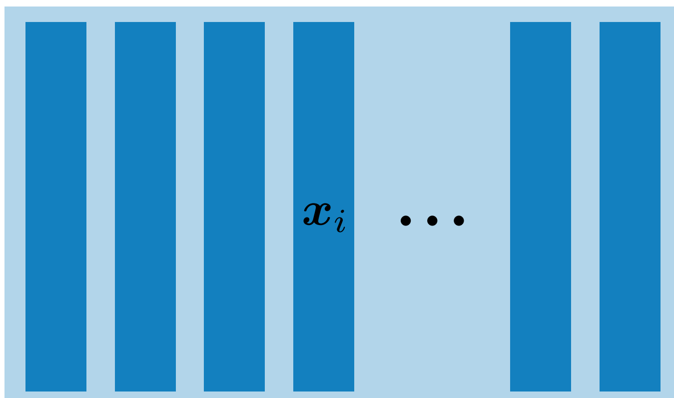
- [x] <u>Connecting literature</u> (Compressive Learning & Generative Networks)
- [x] Practical <u>learning algorithm</u>
- [x] Qualitative <u>proof-of-concept</u> (toy example)
- [ ] Quantitative experimental validation (real data)

Compressive Learning of
Generative Networks

Generative Networks: What?

Learn a generative network $\mathcal{G}_\theta$

X

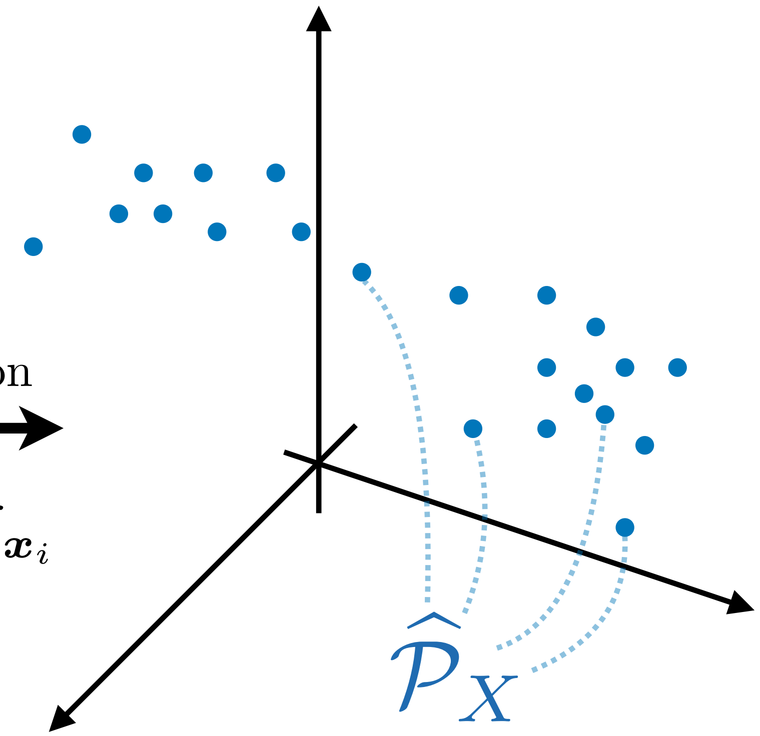


Dataset

We have **samples** (signals)

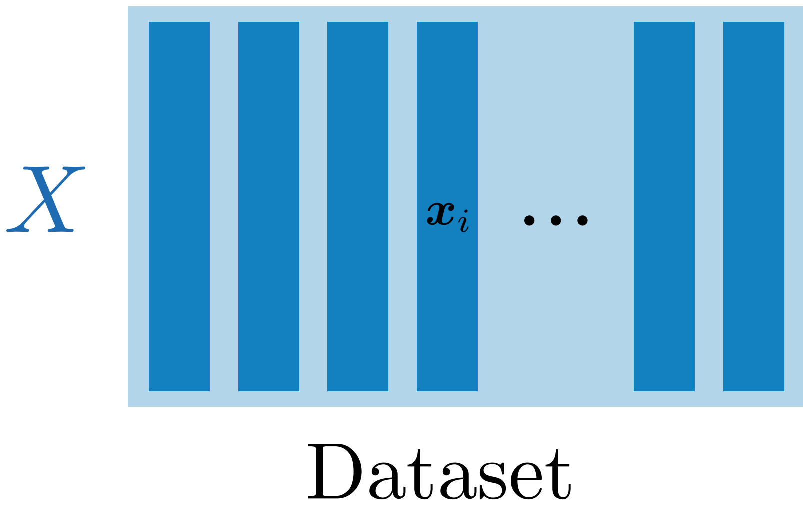
Empirical distribution

$$\widehat{\mathcal{P}}_X = \frac{1}{N} \sum_{\boldsymbol{x}_i \in X} \delta_{\boldsymbol{x}_i}$$



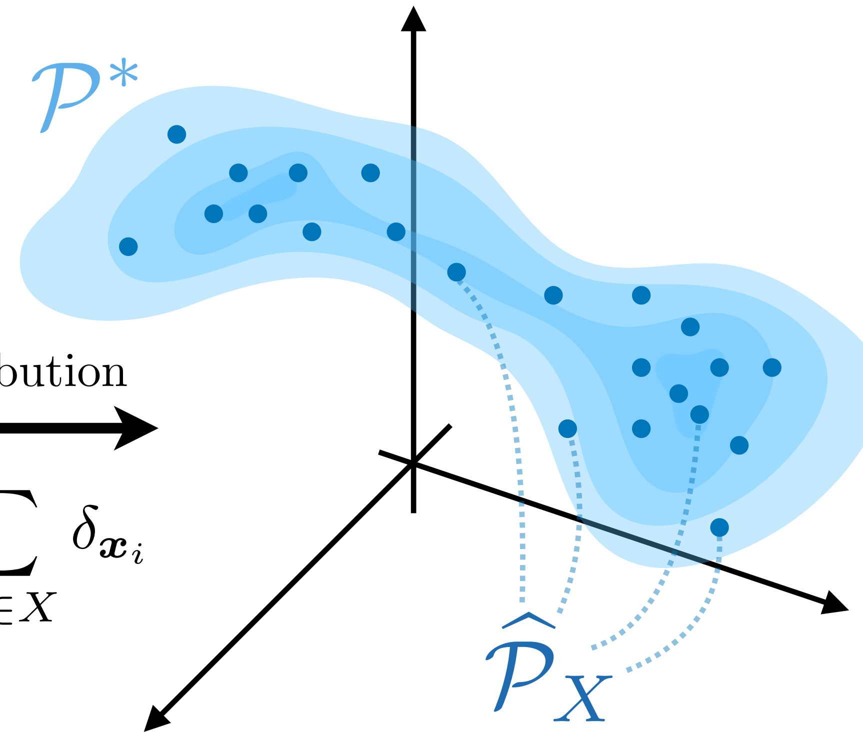
Generative Networks: What?

Learn a <u>generative network</u> $\mathcal{G}_\theta$



Empirical distribution

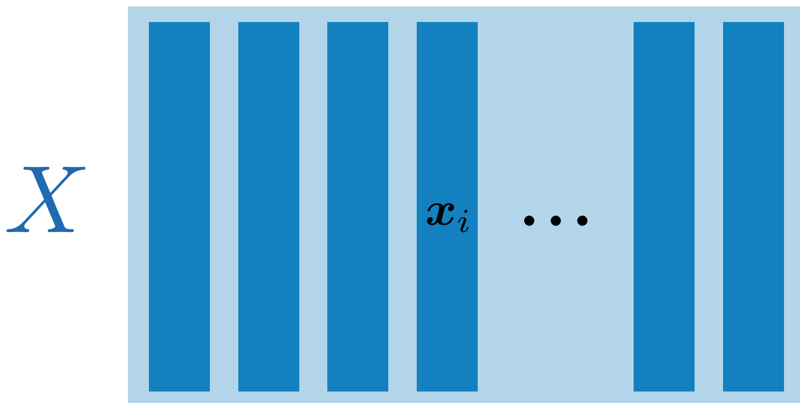
$$\widehat{\mathcal{P}}_X = \frac{1}{N} \sum_{\boldsymbol{x}_i \in X} \delta_{\boldsymbol{x}_i}$$



We have **samples** (signals) from a high-dimensional **distribution**...

Generative Networks: What?

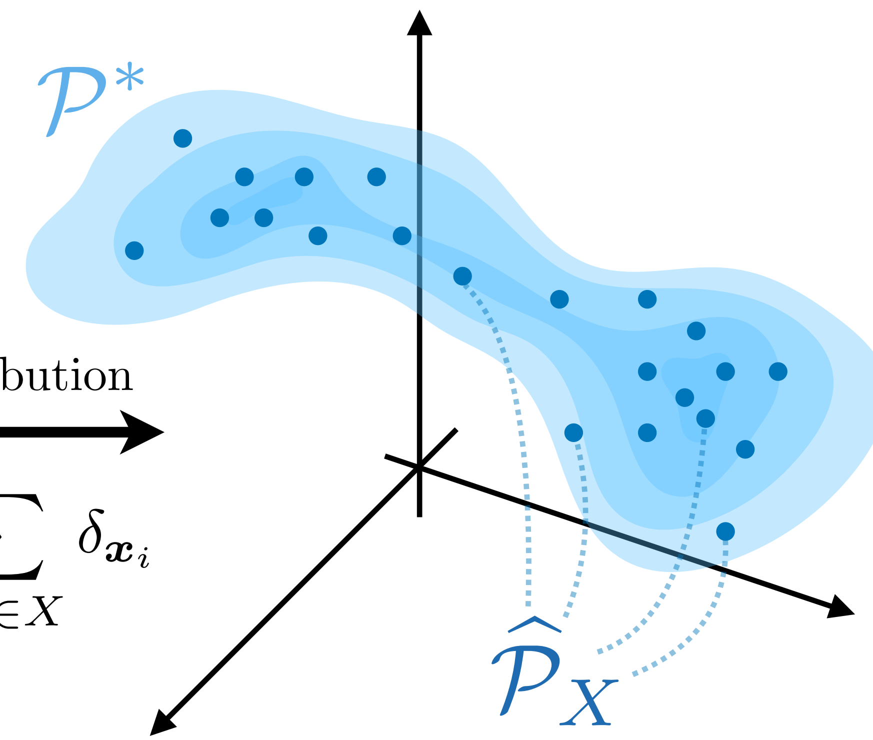
Learn a <u>generative network</u> $\mathcal{G}_\theta$



Dataset

Empirical distribution

$$\widehat{\mathcal{P}}_X = \frac{1}{N} \sum_{\boldsymbol{x}_i \in X} \delta_{\boldsymbol{x}_i}$$

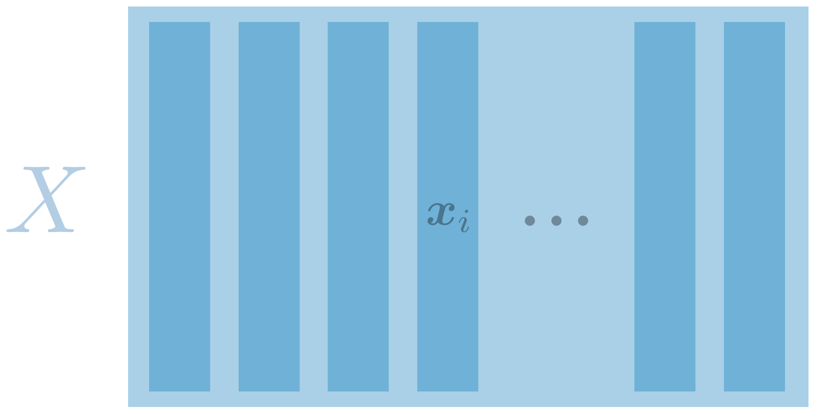


We have **samples** (signals) from a high-dimensional **distribution**...

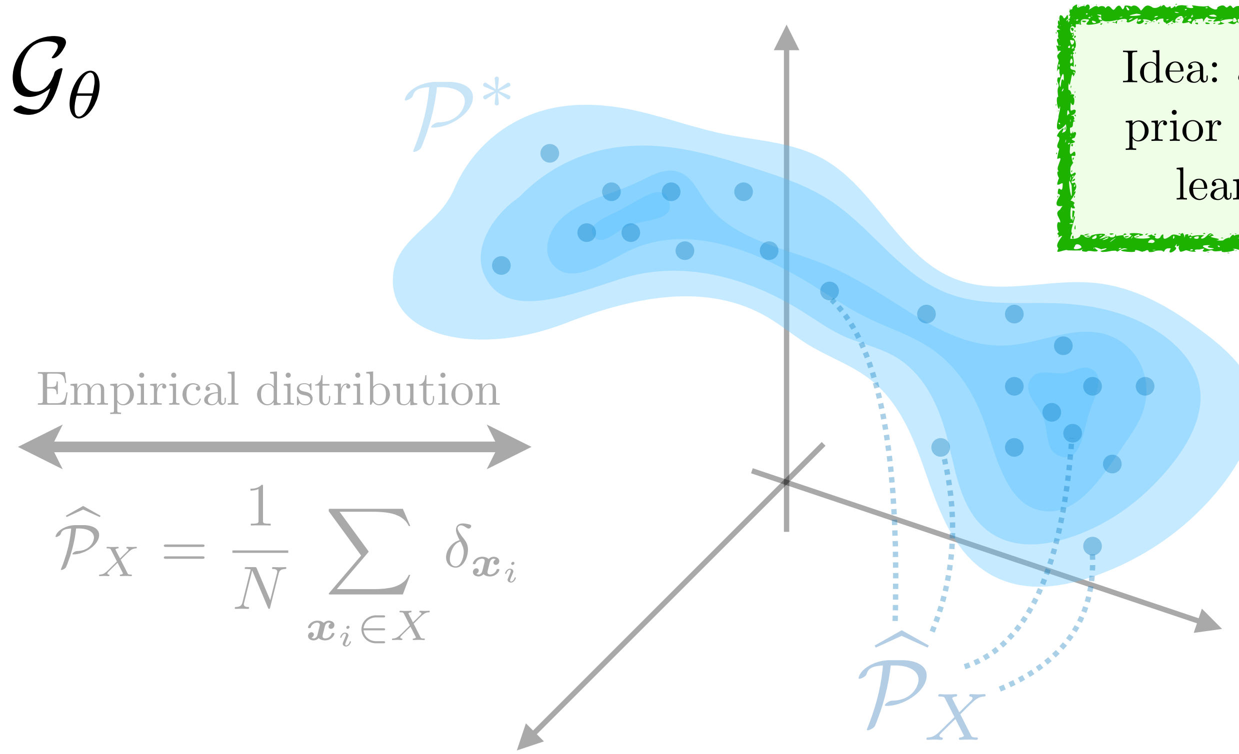
Idea: approximate true prior distribution from learning examples

Generative Networks: What?

Learn a <u>generative network</u> $\mathcal{G}_\theta$



Dataset

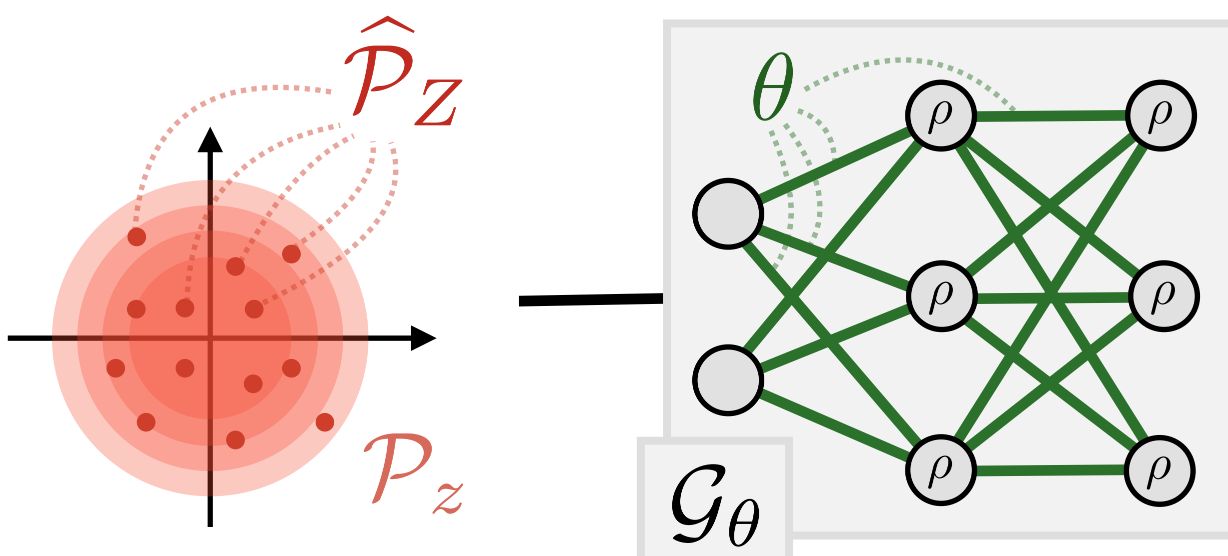


Idea: approximate true prior distribution from learning examples

We have **samples** (signals) from a high-dimensional **distribution**...

... and a way to generate “artificial” samples...

$$\mathcal{G}_\theta(\boldsymbol{z}) = \rho(\Theta_L \cdot \rho(\Theta_{L-1} \cdots \rho(\Theta_2 \cdot \rho(\Theta_1 \cdot \boldsymbol{z}))))$$



Latent space

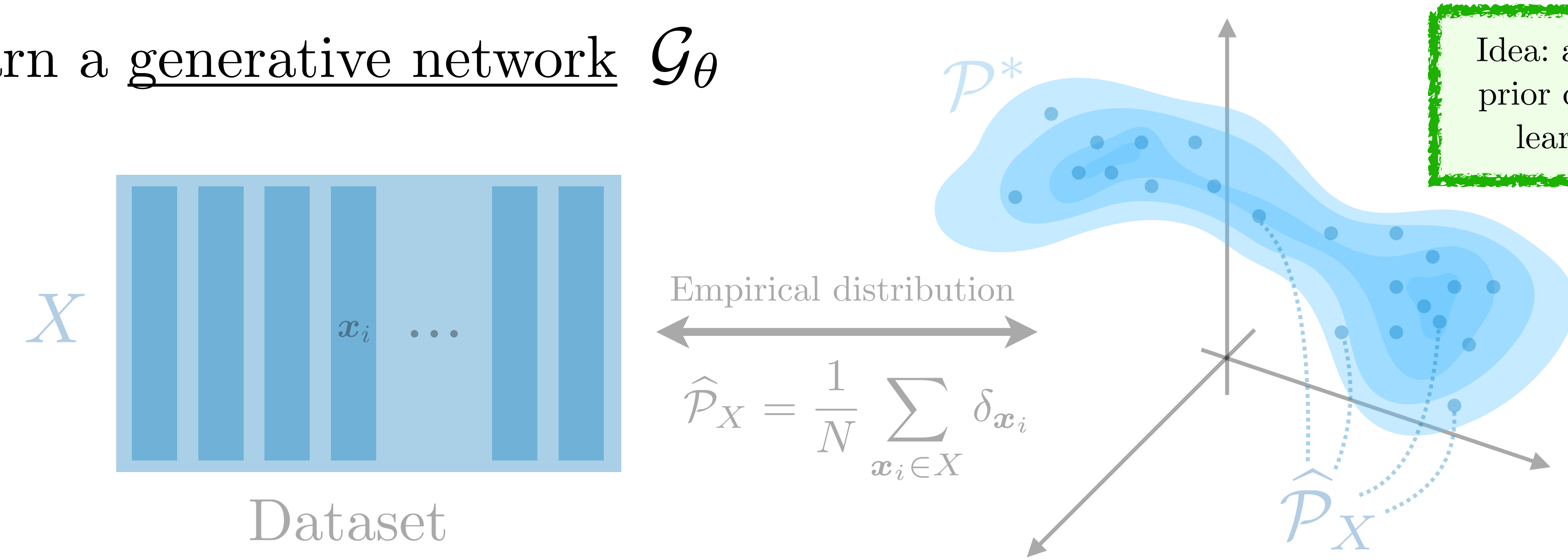
Random “noise”

Generative network

Generative Networks: What?

Learn a <u>generative network</u> $\mathcal{G}_\theta$

Idea: approximate true prior distribution from learning examples

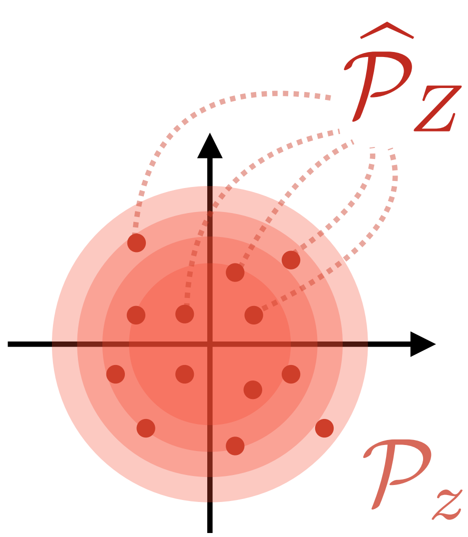


Dataset

We have **samples** (signals) from a high-dimensional **distribution**...

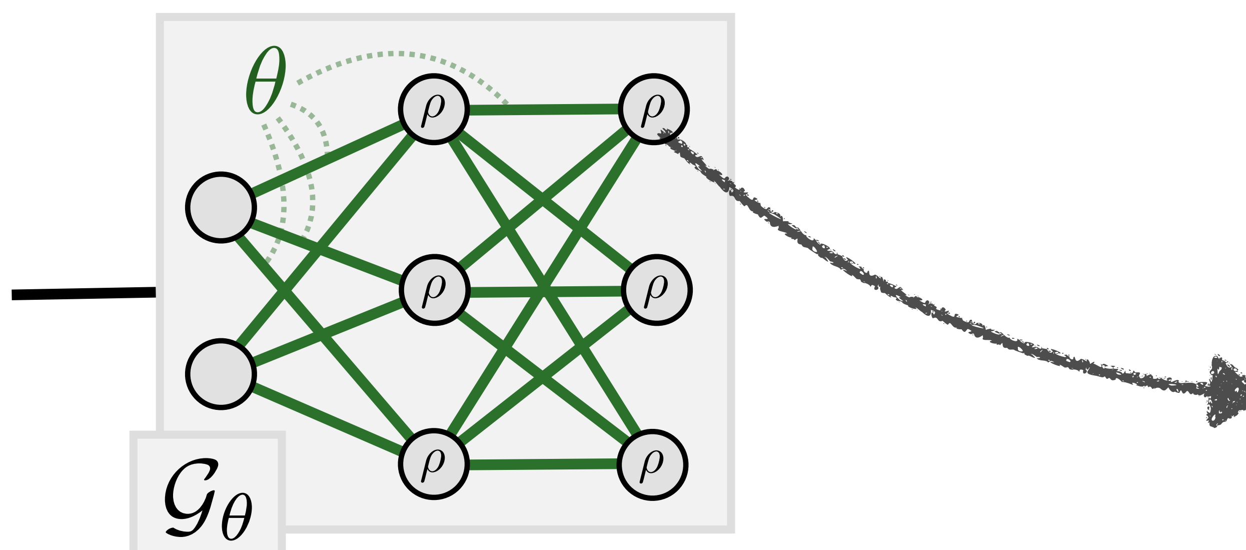
... and a way to generate “artificial” samples...

$$\mathcal{G}_\theta(\boldsymbol{z}) = \rho(\Theta_L \cdot \rho(\Theta_{L-1} \cdots \rho(\Theta_2 \cdot \rho(\Theta_1 \cdot \boldsymbol{z}))))$$

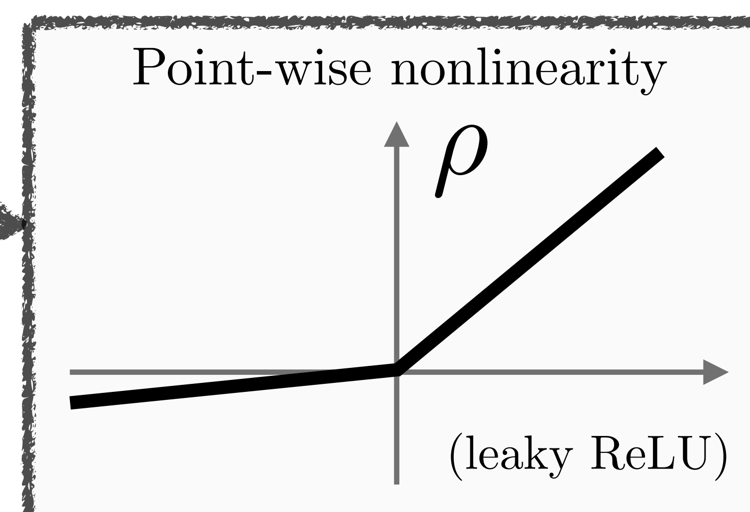


Latent space

Random “noise”



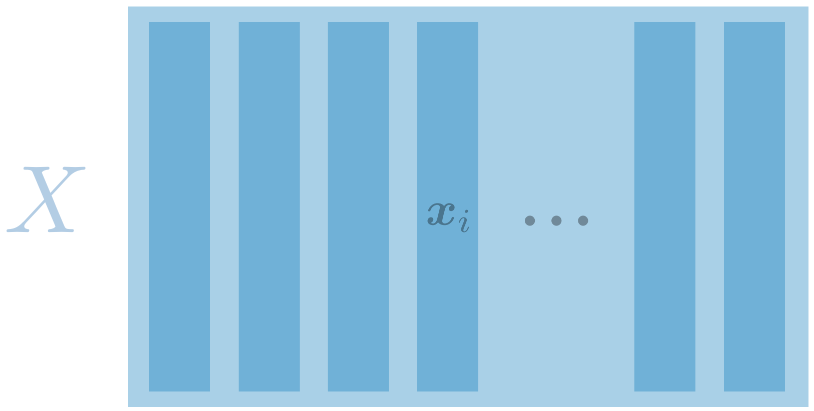
Generative network



Generative Networks: What?

Learn a <u>generative network</u> $\mathcal{G}_\theta$

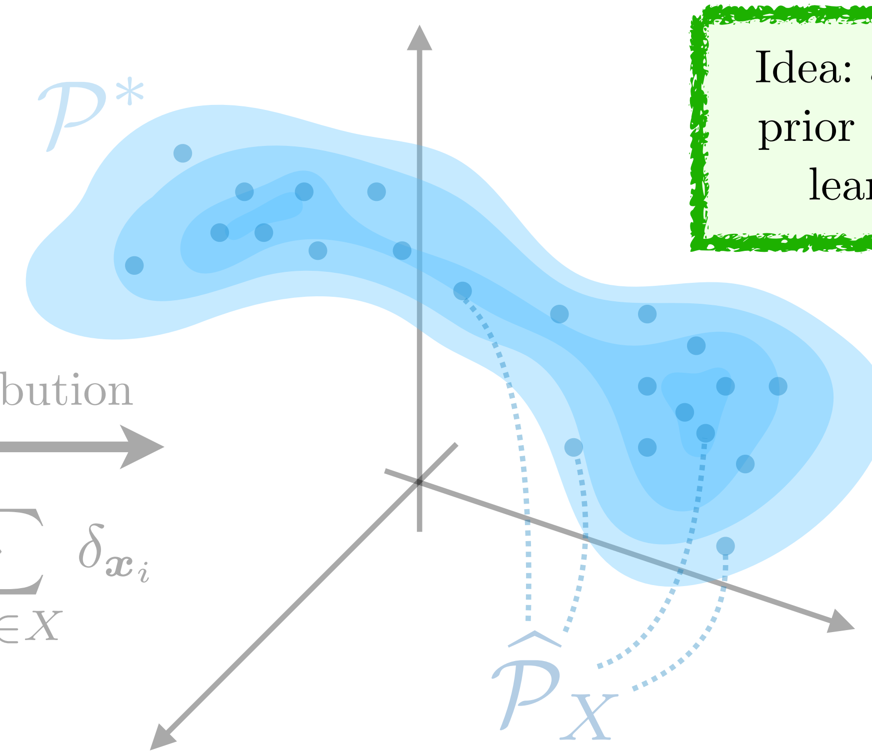
Idea: approximate true prior distribution from learning examples



Dataset

Empirical distribution

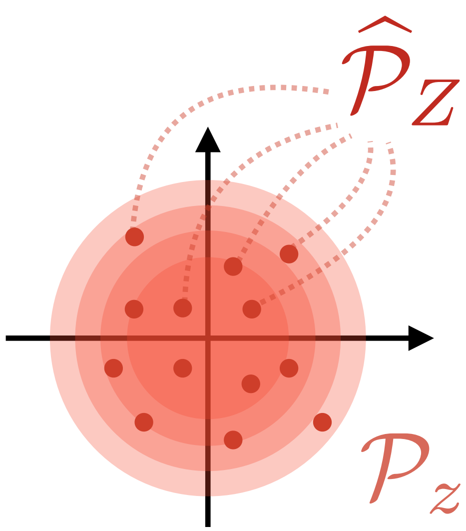
$$\widehat{\mathcal{P}}_X = \frac{1}{N} \sum_{x_i \in X} \delta_{x_i}$$



We have **samples** (signals) from a high-dimensional **distribution**...

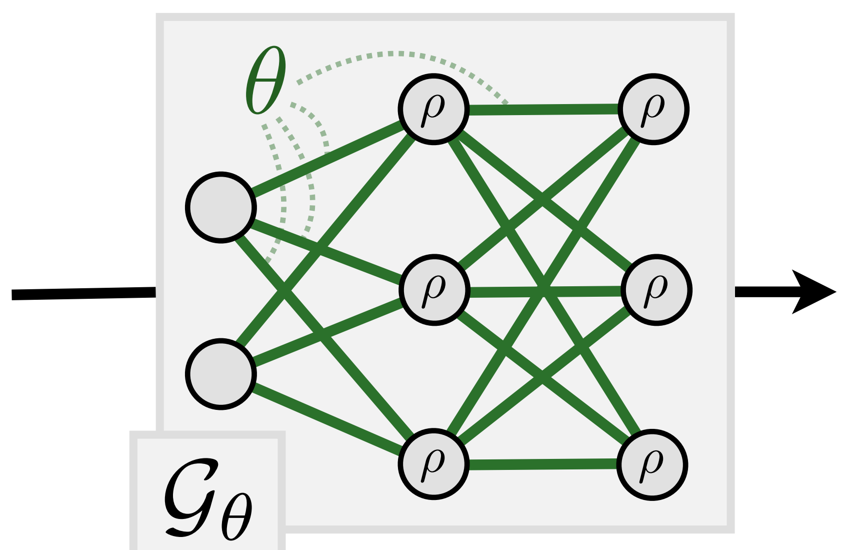
... and a way to generate "artificial" samples...

$$\mathcal{G}_\theta(z) = \rho(\Theta_L \cdot \rho(\Theta_{L-1} \cdots \rho(\Theta_2 \cdot \rho(\Theta_1 \cdot z))))$$

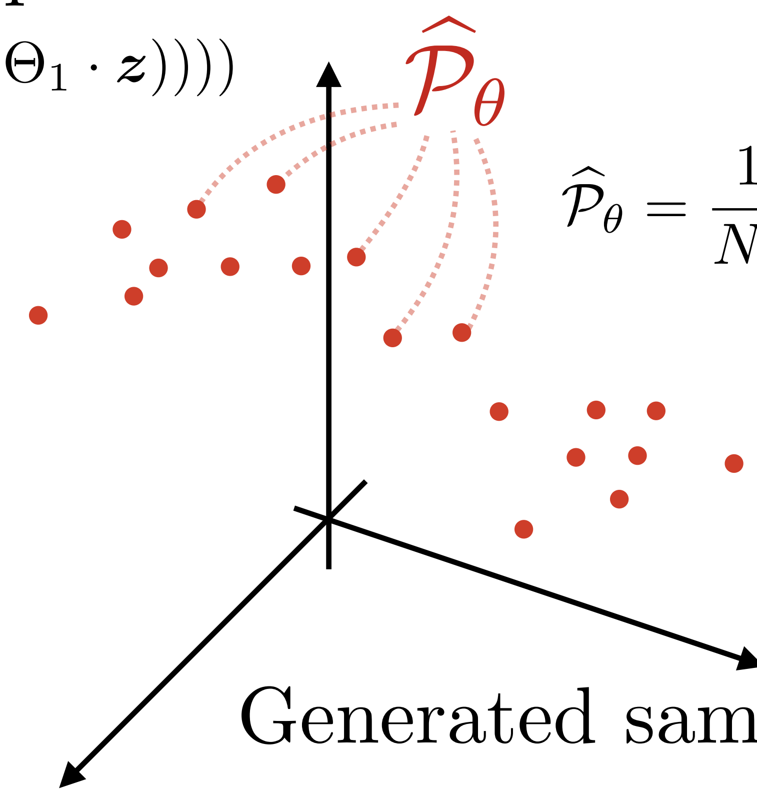


Latent space

Random "noise"



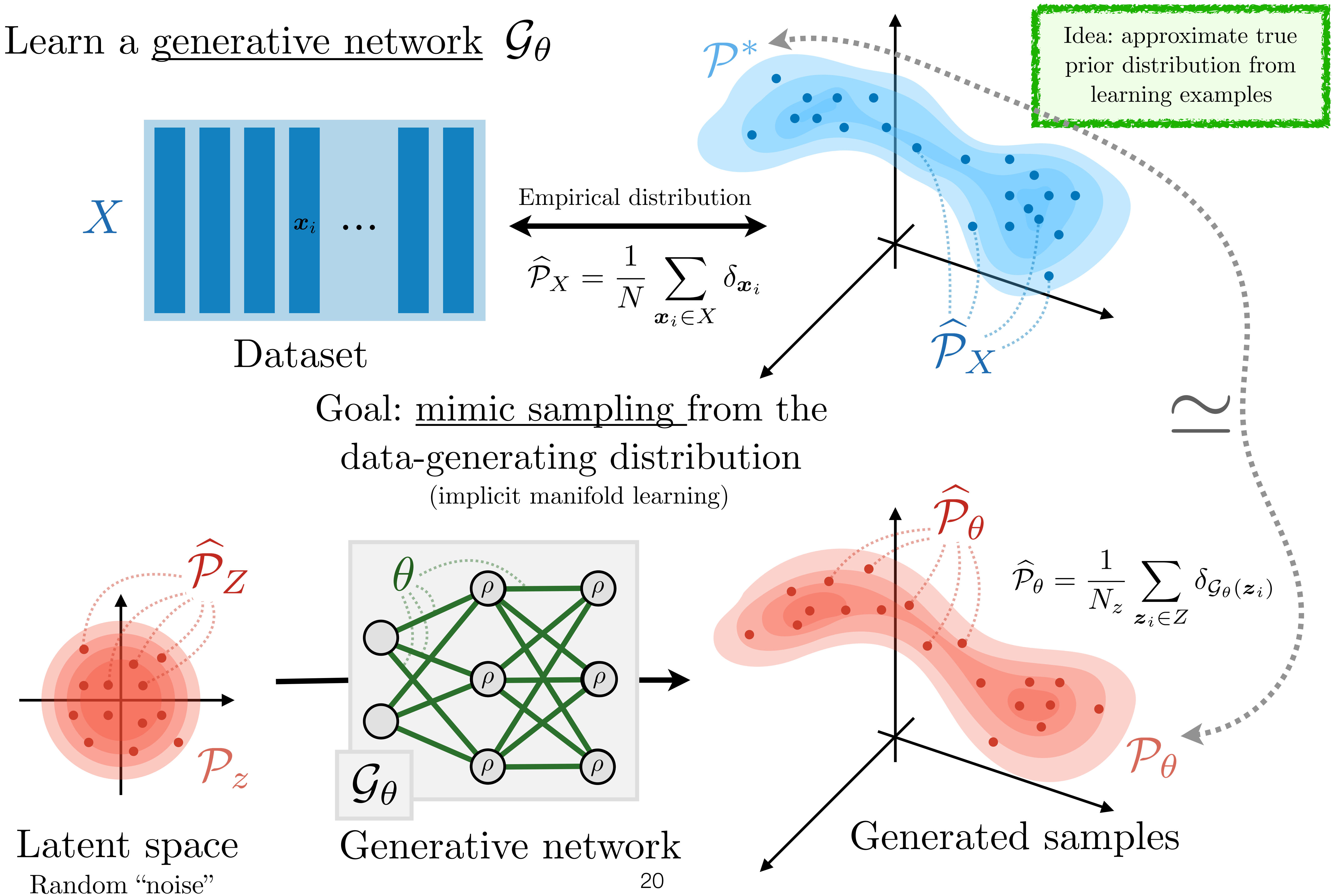
Generative network



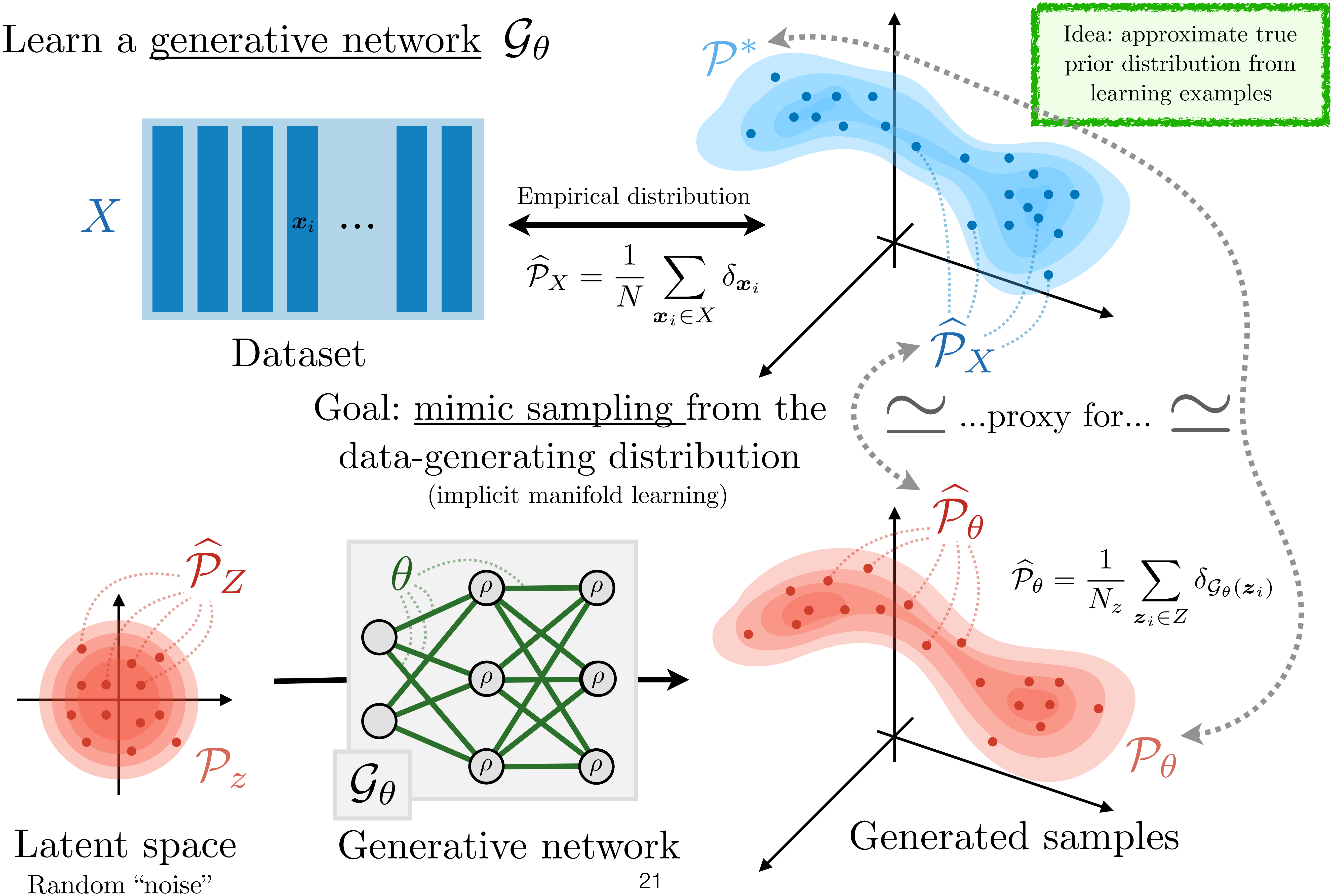
$$\widehat{\mathcal{P}}_\theta = \frac{1}{N_z} \sum_{z_i \in Z} \delta_{\mathcal{G}_\theta(z_i)}$$

Generated samples

Generative Networks: What?



Generative Networks: What?



Generative Networks: Why?

Motivation (amongst others): priors in inverse problems

SUNLayer: Stable denoising with generative networks

Dustin G. Mixon* Soledad Villar†

Abstract

It has been experimentally established that deep neural netw
for real world data. It has also been established that such genera
problems like compressed sensing and super resolution. In this
lem of image denoising. We propose a theoretical setting that u
properties of the activation functions will allow signal denoisin

1 Introduction

Deep neural networks, in particular generative adversarial netw
used to produce generative models for real world data that can
for natural images (see for instance [Nguyen et al., 2016]). T
efficiently solve classical inverse problems in signal processing
pressed sensing ([Bora et al., 2017]). The latter numerically de
solve the compressed sensing problem with ten times fewer me
requires. Follow-up work by [Hand and Voroninski, 2017] rec
empirical risk minimization) in the compressed sensing task b
network with random weights and ReLU activation functions.

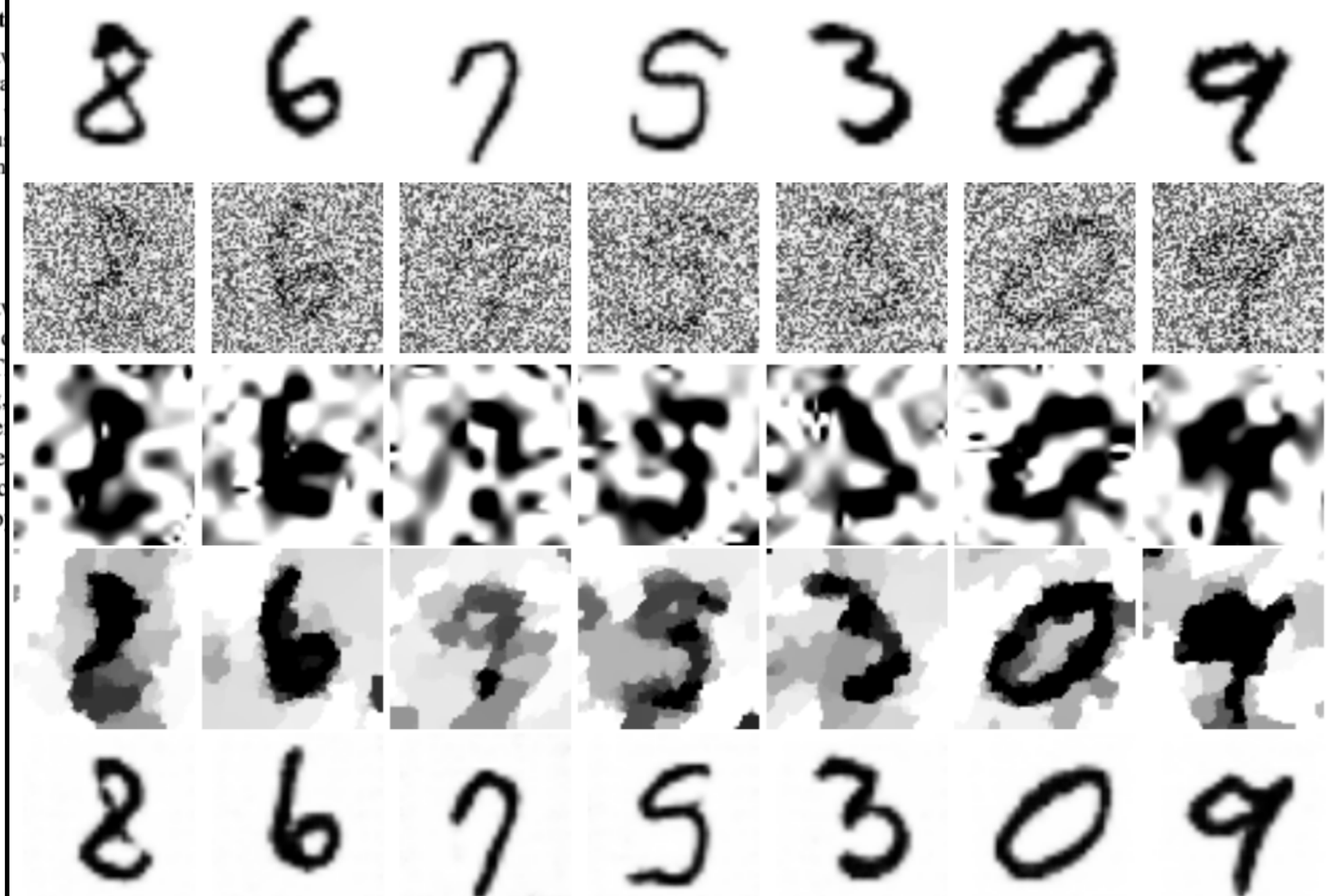


Figure 1: Denoising with generative priors
(First line) Digits from the MNIST test set ([LeCun, 1998]). **(Second line)** random noise is added to the digits. **(Third line)** Denoising of images by shrinkage in wavelet domain ([Donoho and Johnstone, 1994]). **(Fourth line)** Denoising by minimizing total variation ([Rudin et al., 1992]). **(Fifth line)** We train a GAN using the training set of MNIST to obtain a generative model G. We denoise by finding the closest element in the image of G using stochastic grading descent.

Generative Networks: Why?

Motivation (amongst others): priors in inverse problems

Blind Image Deconvolution using Deep Generative Priors

Muhammad Asim* , Fahad Shamshad*, and Ali Ahmed

***Abstract*—This paper proposes a novel approach to regularize the *ill-posed* and *non-linear* blind image deconvolution (blind deblurring) using deep generative networks as priors. We employ two separate generative models — one trained to produce sharp images while the other trained to generate blur kernels from lower-dimensional parameters. To deblur, we propose an alternating gradient descent scheme operating in the latent lower-dimensional space of each of the pretrained generative models. Our experiments show promising deblurring results on images even under large blurs, and heavy noise. To address the shortcomings of generative models such as mode collapse, we augment our generative priors with classical image priors and report improved performance on complex image datasets. The deblurring performance depends on how well the range of the generator spans the image class. Interestingly, our experiments show that even an untrained structured (convolutional) generative networks acts as an image prior in the image deblurring context allowing us to extend our results to more diverse natural image datasets.**

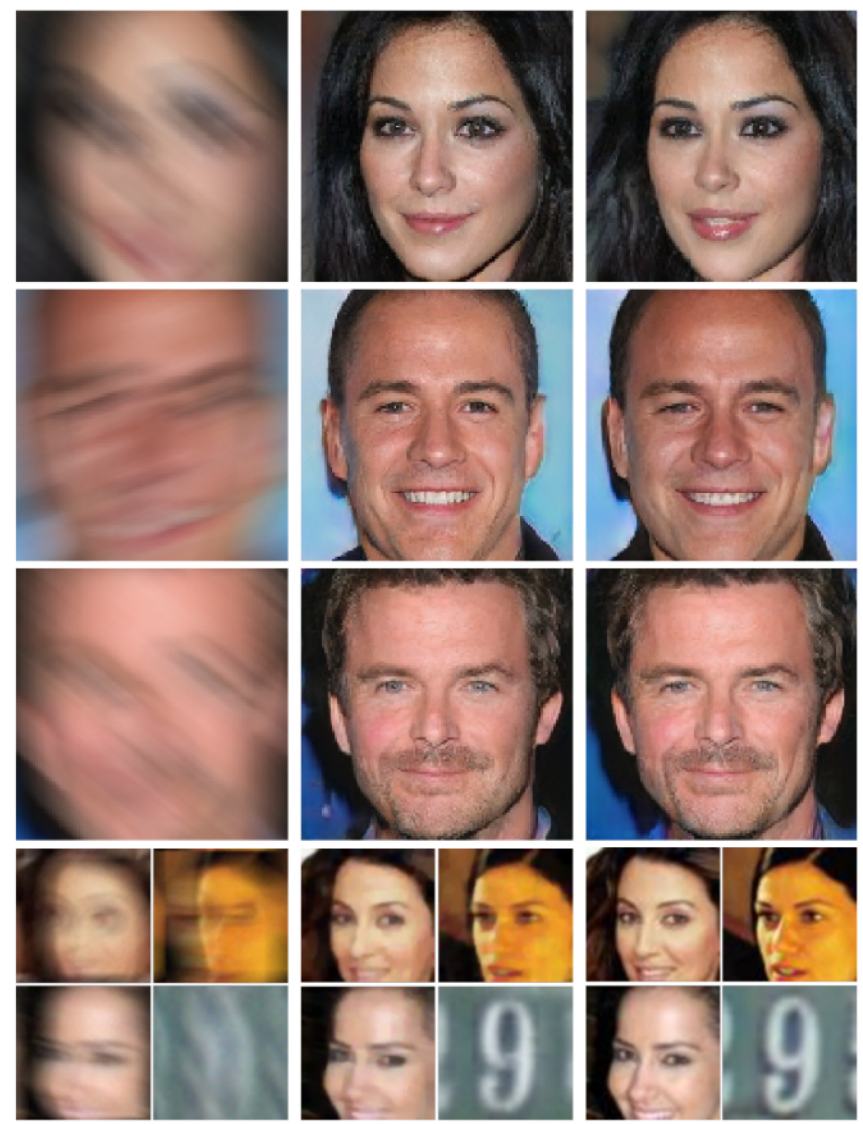
***Index Terms*—Blind image deblurring, generative adversarial networks, variational autoencoders, deep image prior.**

I. Introduction

BLIND image deblurring aims to recover a true image i and a blur kernel k from blurry and possibly noisy observation y. For a uniform and spatially invariant blur, it can be mathematically formulated as

$$y = i \otimes k + n, \qquad (1)$$

where $\otimes$ is a convolution operator and n is an additive Gaussian noise. In its full generality, the inverse problem (1) is severely ill-posed as many different instances of i, and k fit the observation y, see [1], [2] for a thorough discussion on

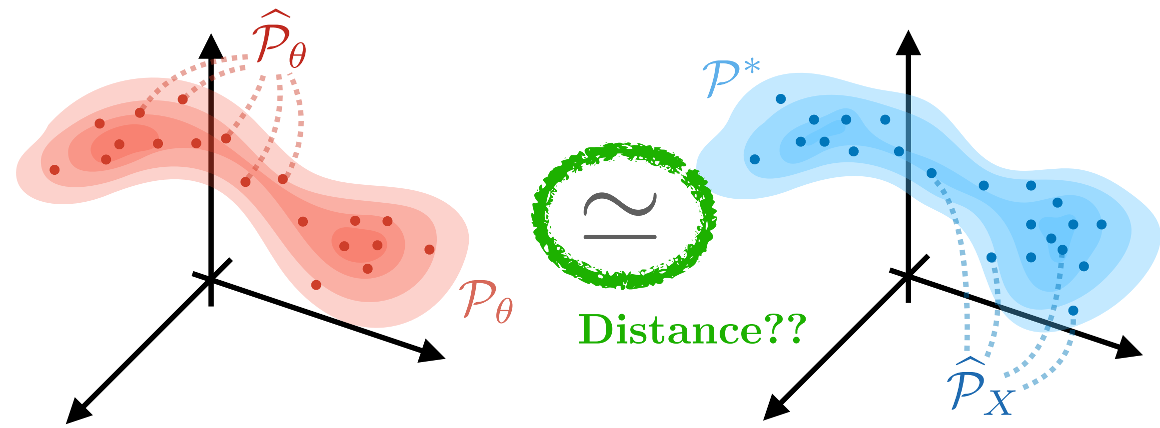


(a) Blurry (b) Ours (c) Original

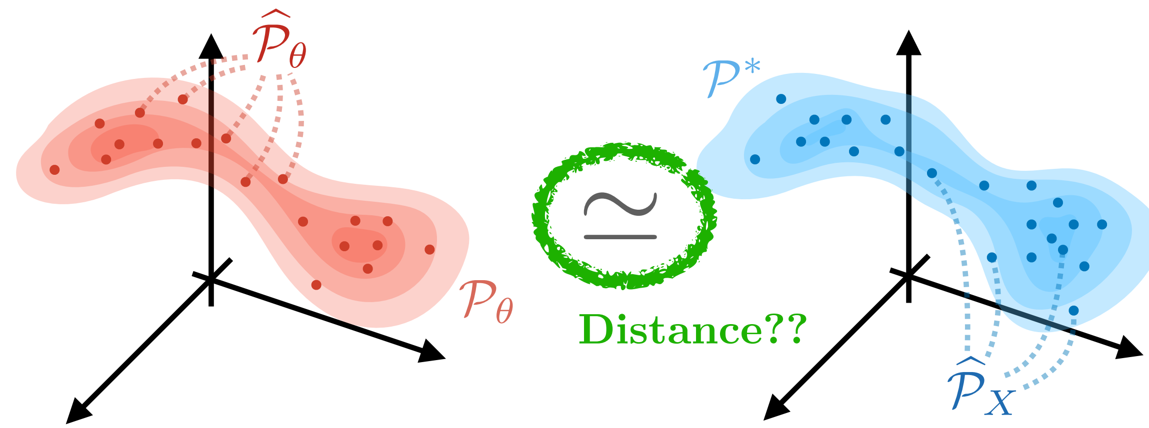
Fig. 1: Blind image deblurring using deep generative priors.

Generative Networks: How?

How to learn the generative network $\mathcal{G}_\theta$



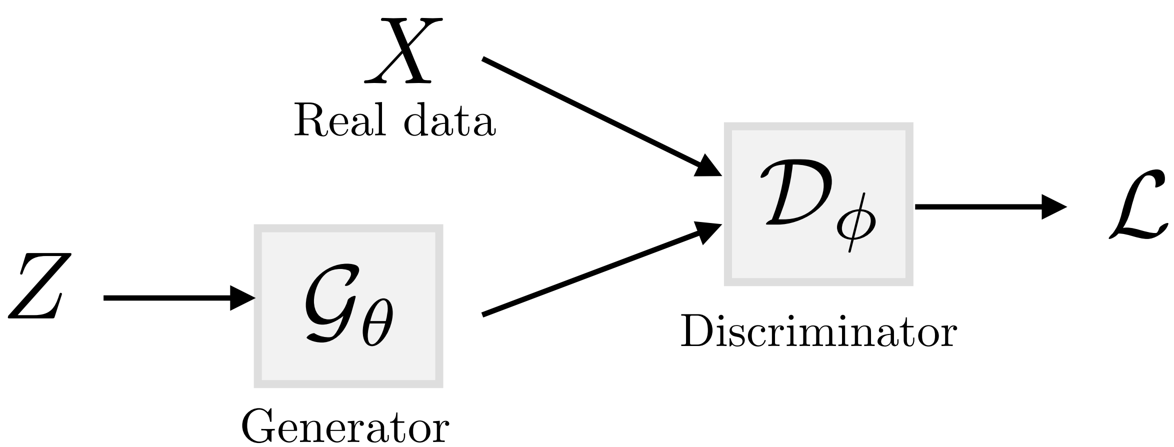
Generative Networks: How?



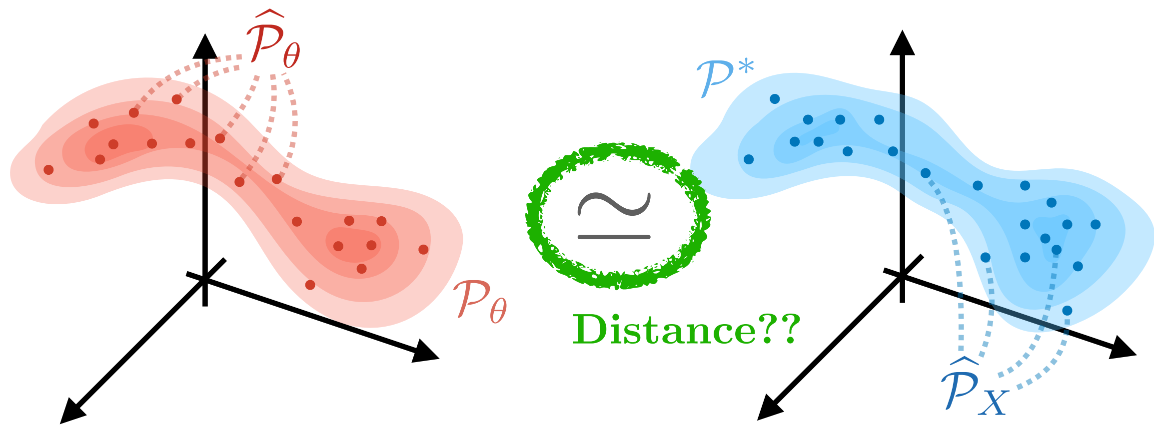
<u>How</u> to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator



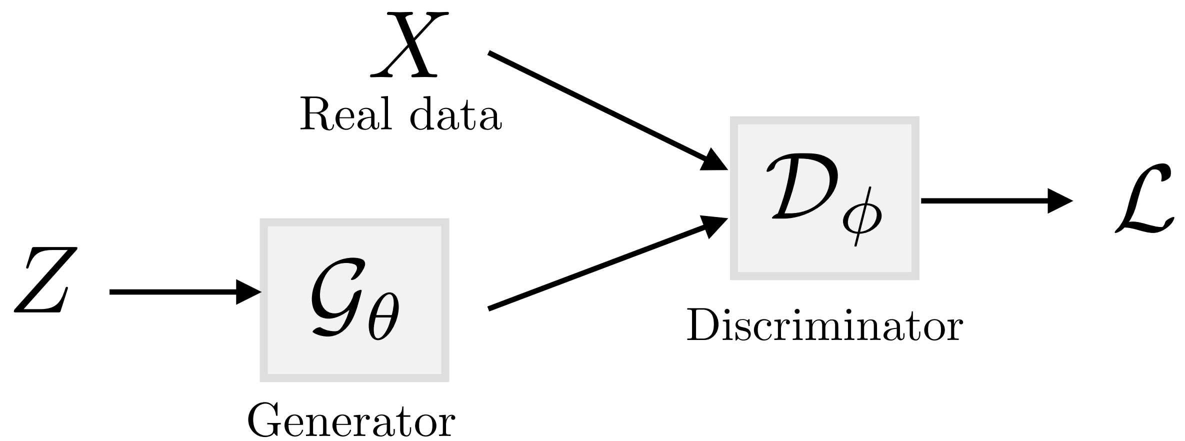
Generative Networks: How?



<u>How</u> to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator

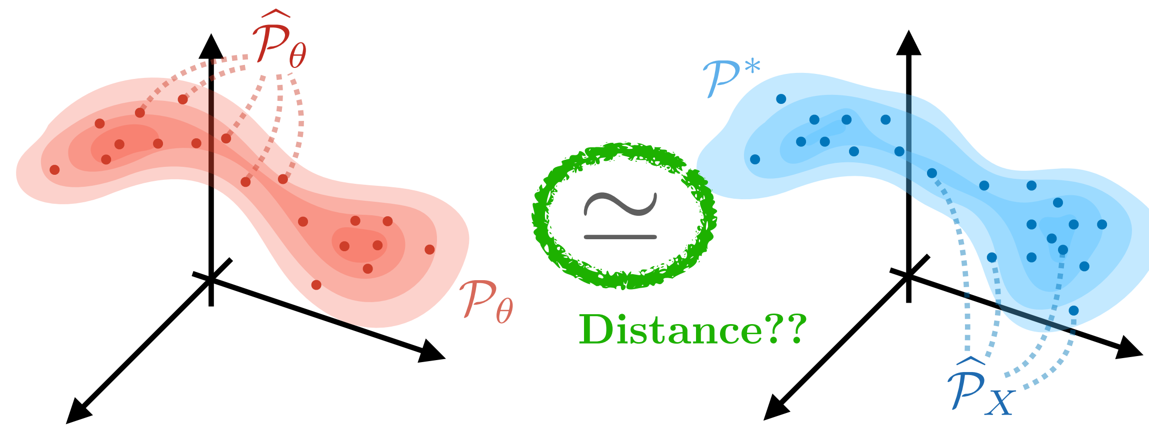


<u>Very difficult</u> to train (due to balancing of training discriminator/generator)

$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$$

Lots of research interest in other (easier) methods

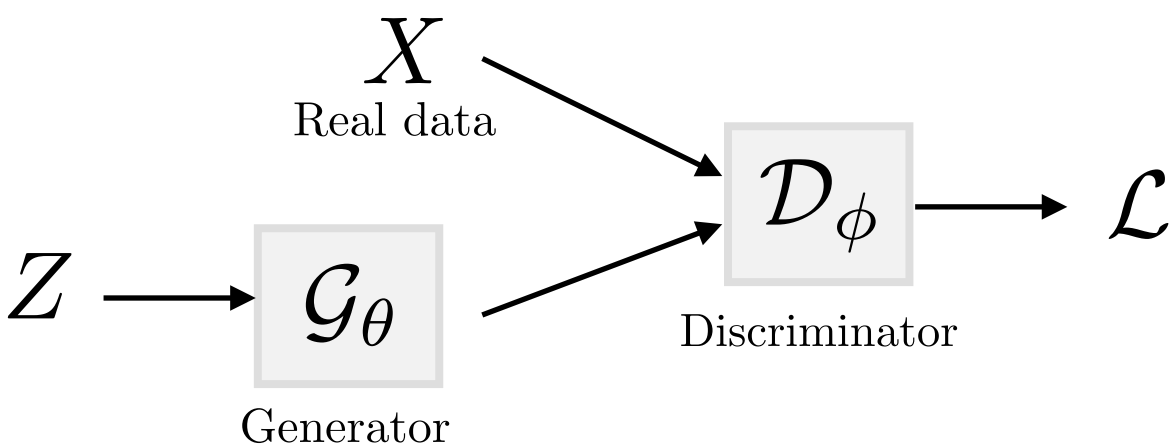
Generative Networks: How?



<u>How</u> to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator



<u>Very difficult</u> to train (due to balancing of training discriminator/generator)

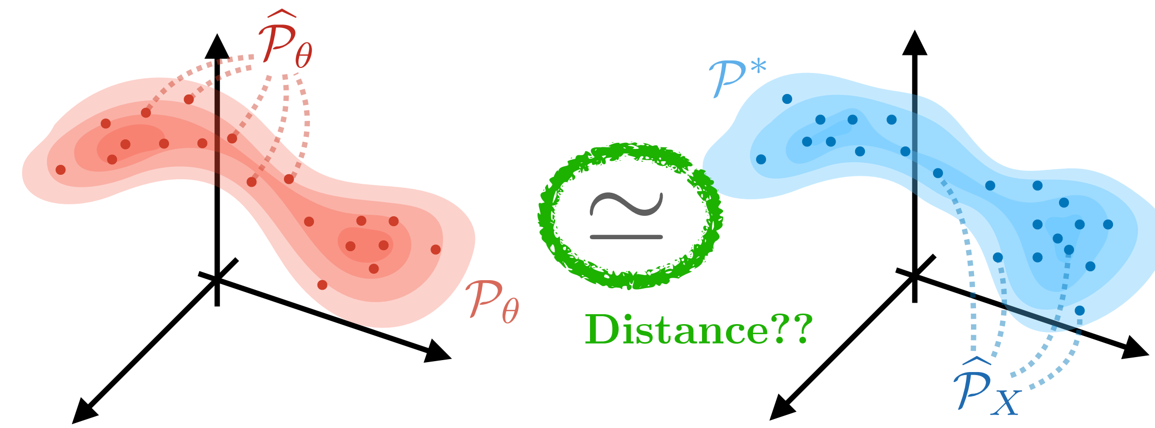
$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$$

2) Maximum Mean Discrepancy

$$\min_\theta \ \mathrm{MMD}_\kappa(\widehat{\mathcal{P}}_X, \widehat{\mathcal{P}}_\theta)$$

Easier to train (no balancing)...

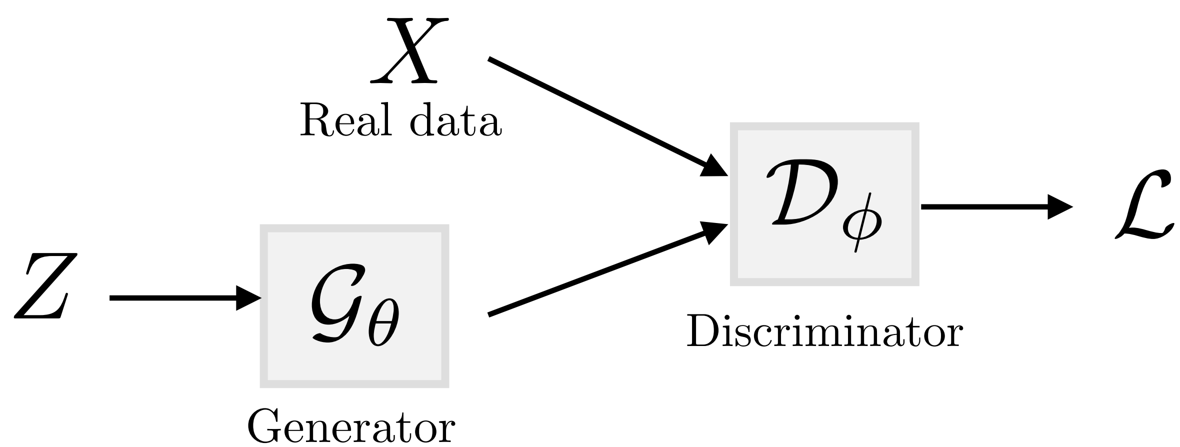
Generative Networks: How?



How to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator



Very difficult to train (due to balancing of training discriminator/generator)

$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$$

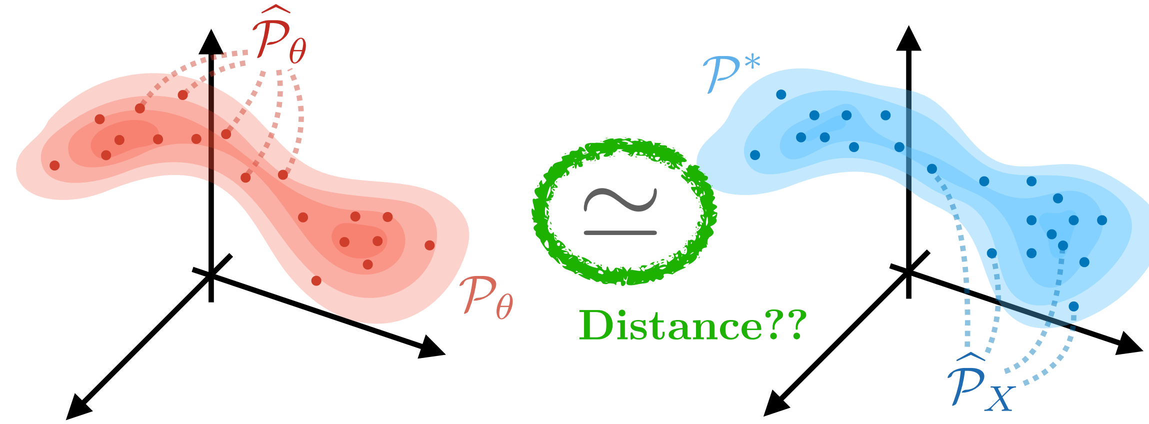
2) Maximum Mean Discrepancy

$$\sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{x}_j \in X}} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{z}_j \in Z}} \kappa(\boldsymbol{x}_i, \mathcal{G}_\theta(\boldsymbol{z}_j)) + \sum_{\substack{\boldsymbol{z}_i \in Z \\ \boldsymbol{z}_j \in Z}} \kappa(\mathcal{G}_\theta(\boldsymbol{z}_i), \mathcal{G}_\theta(\boldsymbol{z}_j))$$

$$\min_\theta \ \mathrm{MMD}_\kappa(\widehat{\mathcal{P}}_X, \widehat{\mathcal{P}}_\theta)$$

Easier to train (no balancing)...

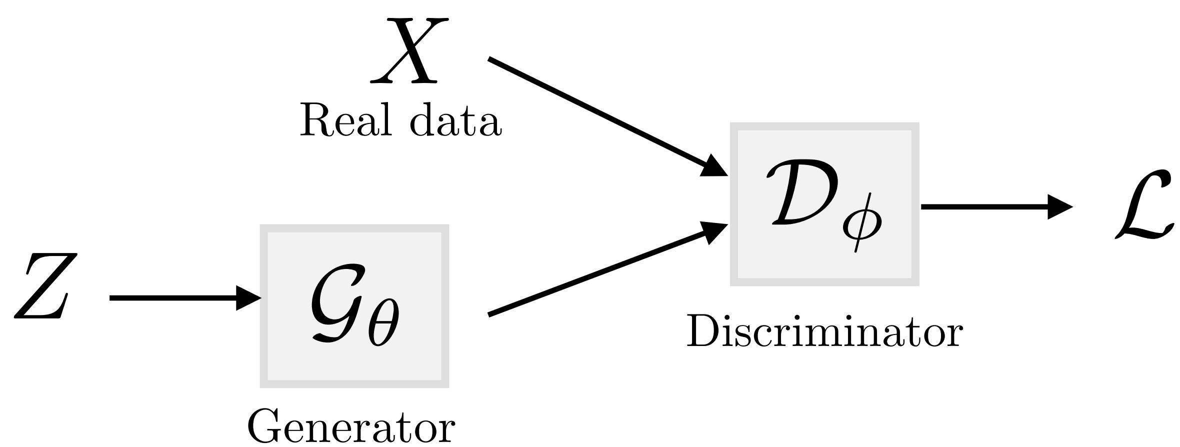
Generative Networks: How?



How to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator



Very difficult to train (due to balancing of training discriminator/generator)

$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$$

2) Maximum Mean Discrepancy

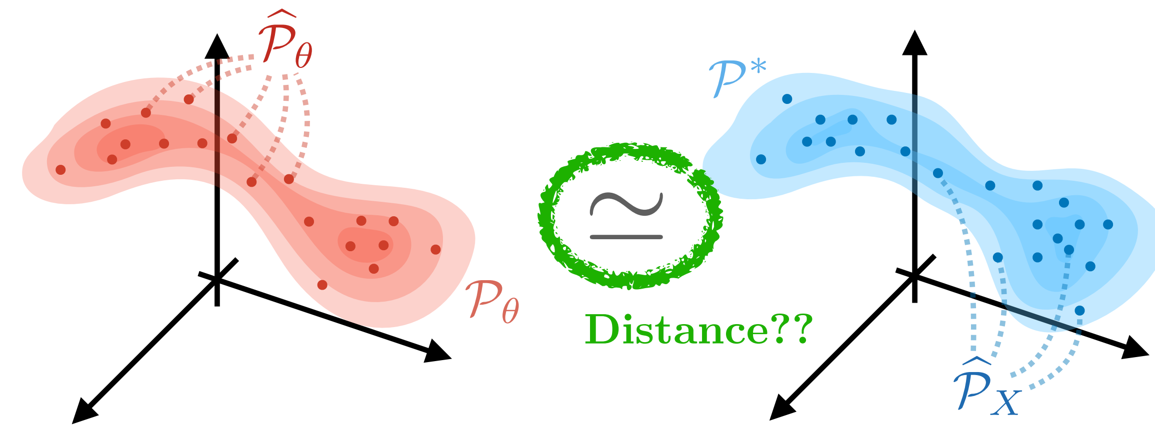
$$\min_\theta \ \mathrm{MMD}_\kappa(\widehat{\mathcal{P}}_X, \widehat{\mathcal{P}}_\theta)$$

Similarity between samples

$$\sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{x}_j \in X}} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{z}_j \in Z}} \kappa(\boldsymbol{x}_i, \mathcal{G}_\theta(\boldsymbol{z}_j)) + \sum_{\substack{\boldsymbol{z}_i \in Z \\ \boldsymbol{z}_j \in Z}} \kappa(\mathcal{G}_\theta(\boldsymbol{z}_i), \mathcal{G}_\theta(\boldsymbol{z}_j))$$

Easier to train (no balancing)...

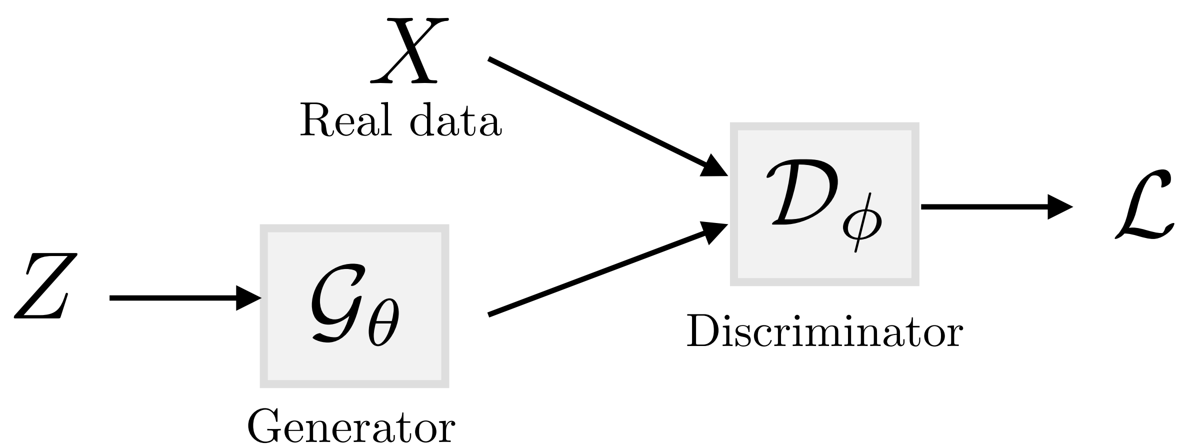
Generative Networks: How?



How to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator



Very difficult to train (due to balancing of training discriminator/generator)

$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$$

2) Maximum Mean Discrepancy

$$\min_\theta \ \mathrm{MMD}_\kappa(\widehat{\mathcal{P}}_X, \widehat{\mathcal{P}}_\theta)$$

Similarity between samples

$$\sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{x}_j \in X}} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{z}_j \in Z}} \kappa(\boldsymbol{x}_i, \mathcal{G}_\theta(\boldsymbol{z}_j)) + \sum_{\substack{\boldsymbol{z}_i \in Z \\ \boldsymbol{z}_j \in Z}} \kappa(\mathcal{G}_\theta(\boldsymbol{z}_i), \mathcal{G}_\theta(\boldsymbol{z}_j))$$

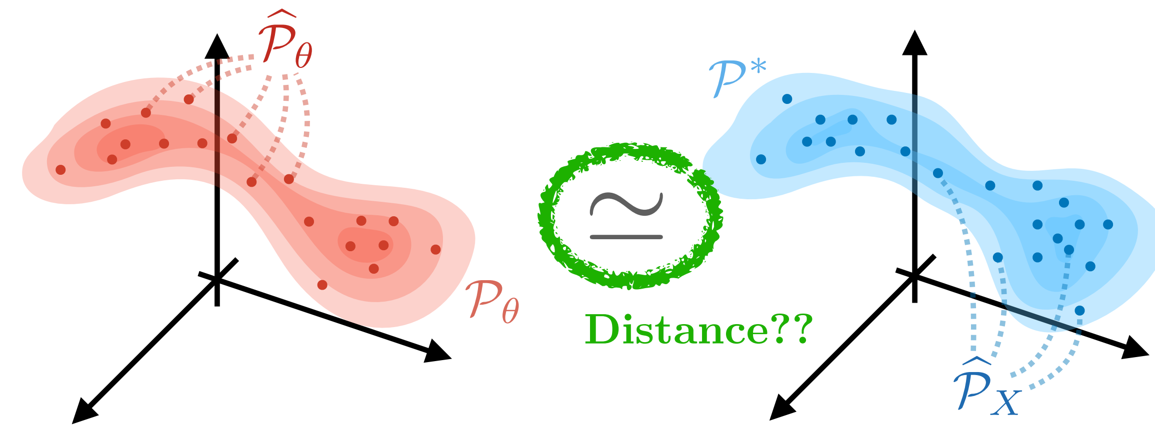
equivalent to (for later)

$$\mathbb{E}_{\boldsymbol{\omega} \sim \Lambda} \left| \sum_{\boldsymbol{x}_i \in X} e^{\mathrm{i}\boldsymbol{\omega}^T \boldsymbol{x}_i} - \sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\boldsymbol{\omega}^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \right|^2$$

Easier to train (no balancing)...

With $\Lambda = \mathcal{F}\kappa$

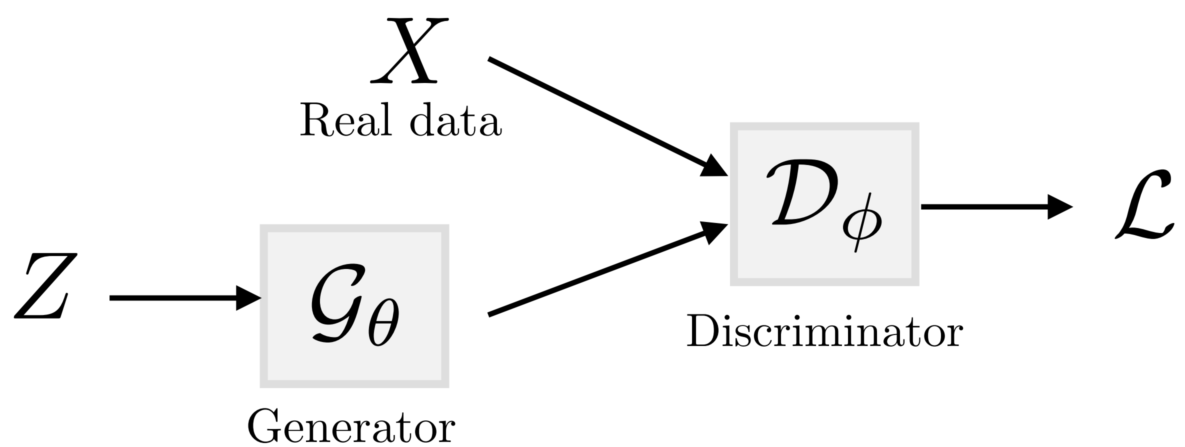
Generative Networks: How?



<u>How</u> to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator



<u>Very difficult</u> to train (due to balancing of training discriminator/generator)

$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$$

2) Maximum Mean Discrepancy

$$\min_\theta \ \mathrm{MMD}_\kappa(\widehat{\mathcal{P}}_X, \widehat{\mathcal{P}}_\theta)$$

Similarity between samples

$$\sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{x}_j \in X}} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{z}_j \in Z}} \kappa(\boldsymbol{x}_i, \mathcal{G}_\theta(\boldsymbol{z}_j)) + \sum_{\substack{\boldsymbol{z}_i \in Z \\ \boldsymbol{z}_j \in Z}} \kappa(\mathcal{G}_\theta(\boldsymbol{z}_i), \mathcal{G}_\theta(\boldsymbol{z}_j))$$

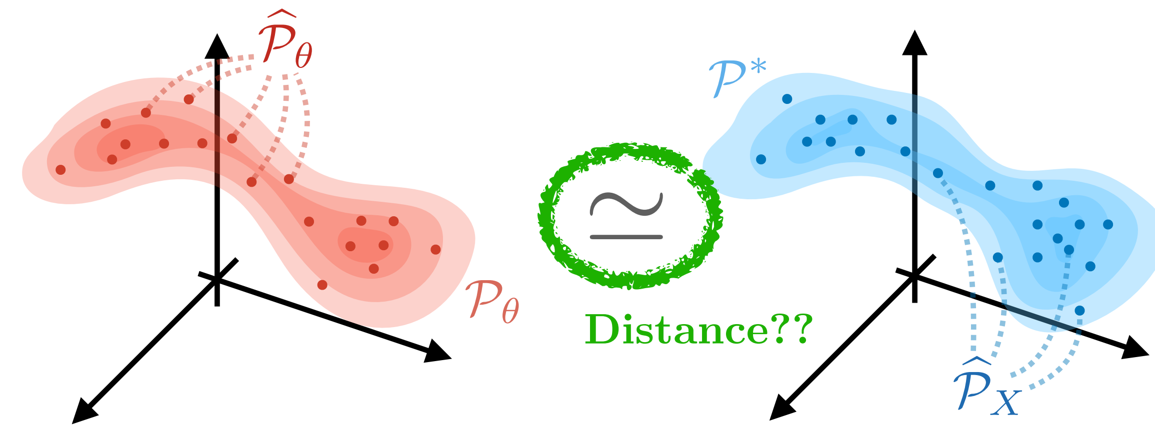
equivalent to (for later)

$$\mathbb{E}_{\boldsymbol{\omega} \sim \Lambda} \left| \sum_{\boldsymbol{x}_i \in X} e^{\mathrm{i}\boldsymbol{\omega}^T \boldsymbol{x}_i} - \sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\boldsymbol{\omega}^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \right|^2$$

With $\Lambda = \mathcal{F}\kappa$

Easier to train (no balancing) but <u>quadratic complexity</u>...

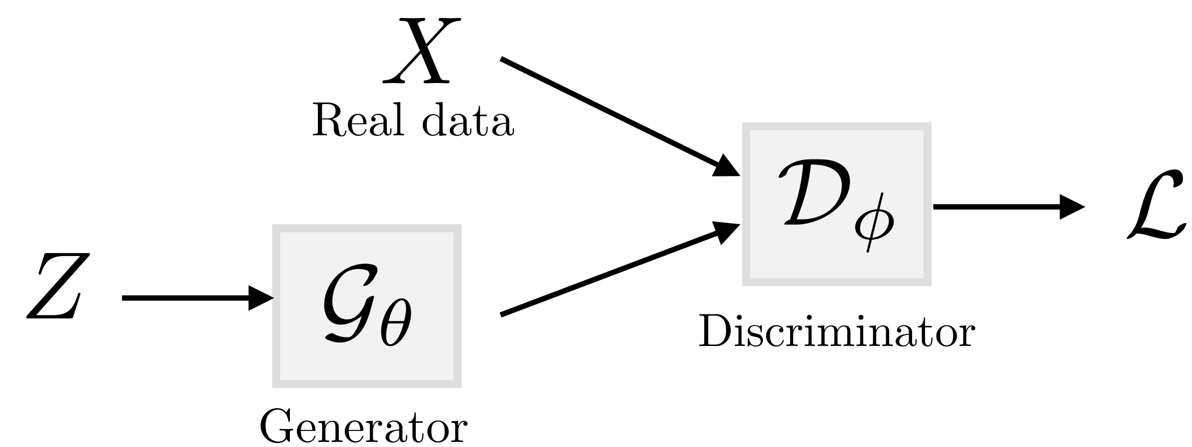
Generative Networks: How?



How to learn the generative network $\mathcal{G}_\theta$

1) Golden standard: Generative Adversarial Networks

Learn a second "discriminator" network that classifies real/fake at the same time as the generator



Very difficult to train (due to balancing of training discriminator/generator)

$$\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$$

2) Maximum Mean Discrepancy

$$\min_\theta \ \mathrm{MMD}_\kappa(\widehat{\mathcal{P}}_X, \widehat{\mathcal{P}}_\theta)$$

Similarity between samples

$$\sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{x}_j \in X}} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2 \sum_{\substack{\boldsymbol{x}_i \in X \\ \boldsymbol{z}_j \in Z}} \kappa(\boldsymbol{x}_i, \mathcal{G}_\theta(\boldsymbol{z}_j)) + \sum_{\substack{\boldsymbol{z}_i \in Z \\ \boldsymbol{z}_j \in Z}} \kappa(\mathcal{G}_\theta(\boldsymbol{z}_i), \mathcal{G}_\theta(\boldsymbol{z}_j))$$

equivalent to (for later)

$$\sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\boldsymbol{\omega}^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \Big|^2$$

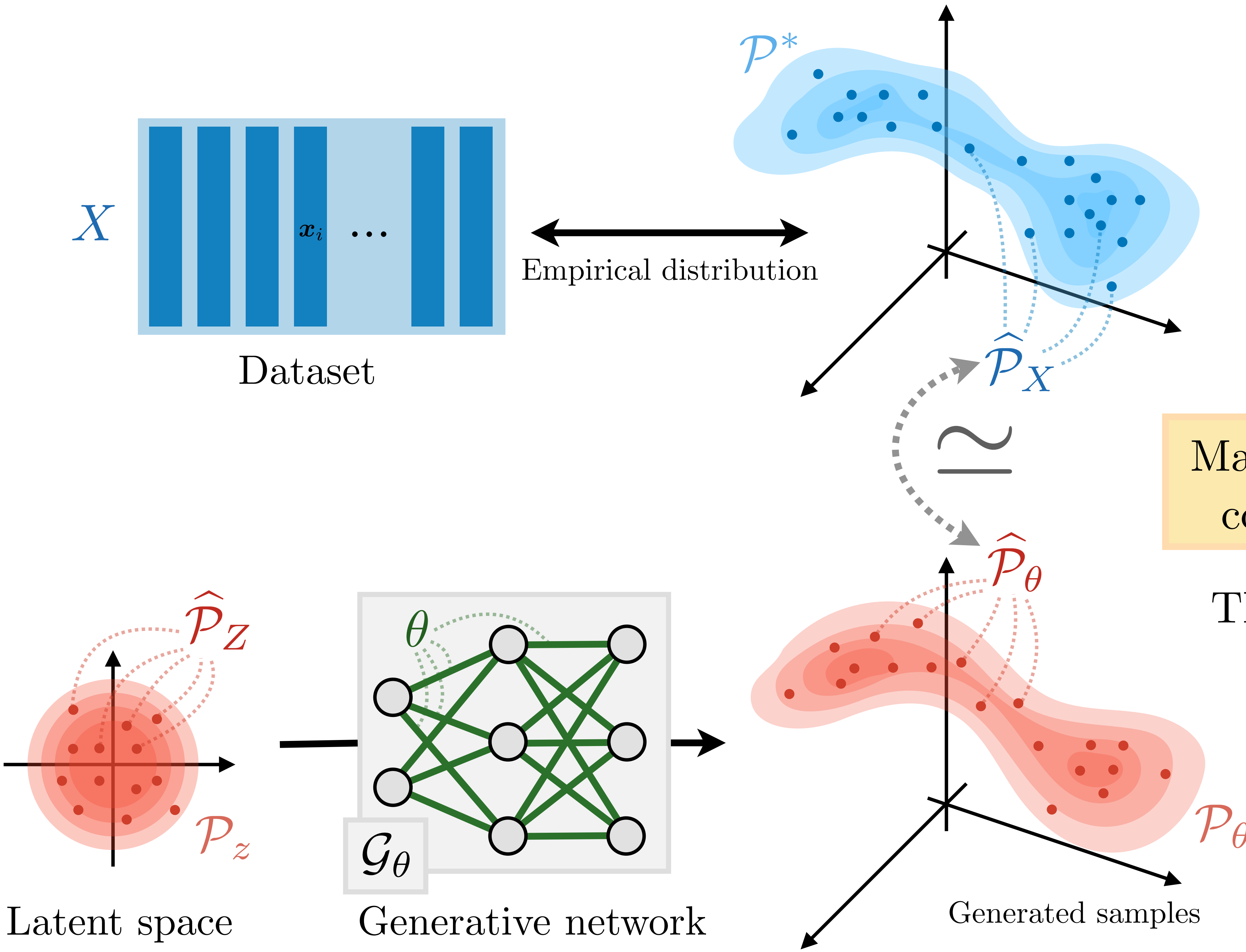
Training generative networks typically requires massive amounts of data!

Easier to train (no balancing) but quadratic complexity...

$$\mathcal{O}(N'(N + N'))$$

Compressive Learning of Generative Networks

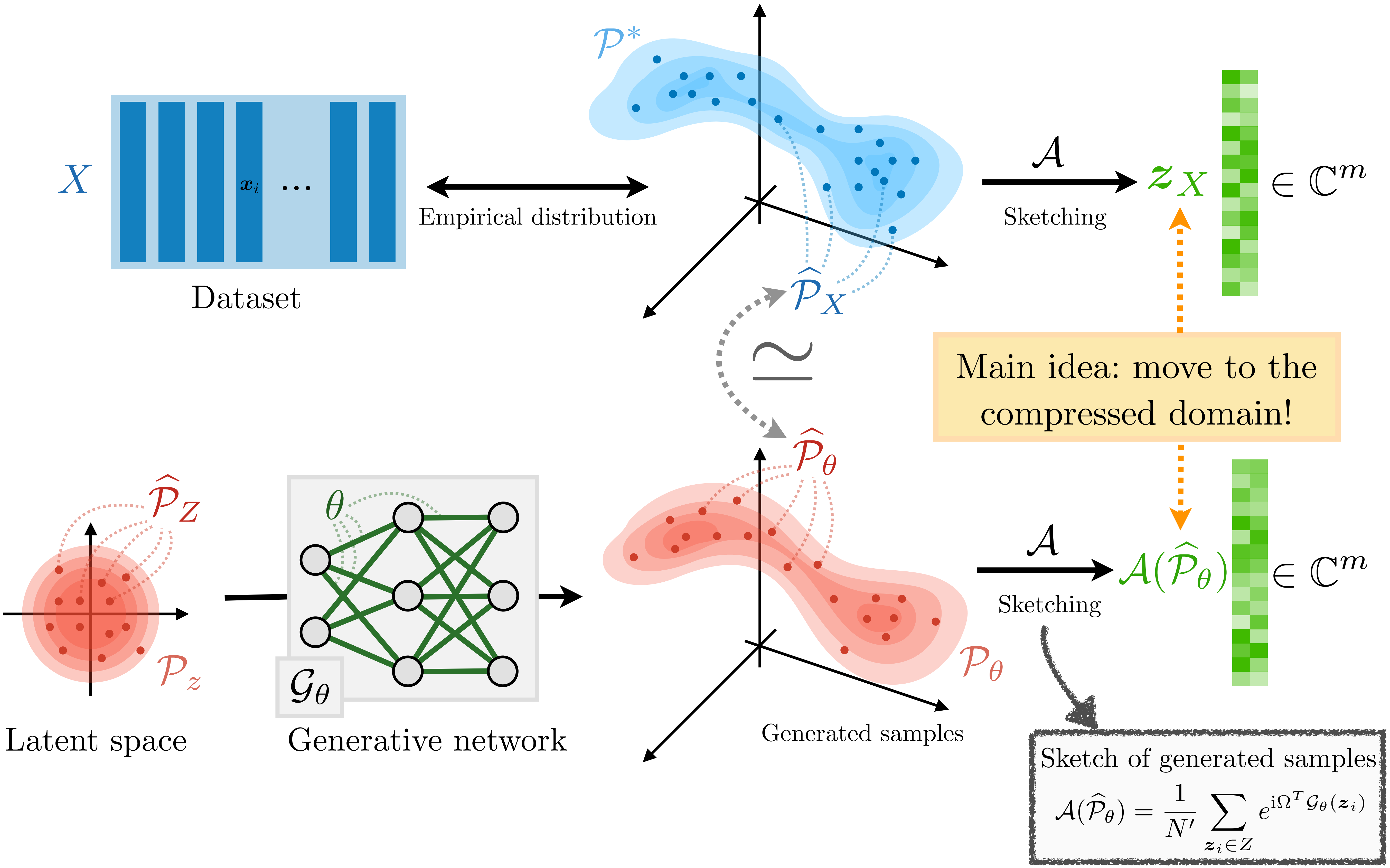
Compressively learning generative networks



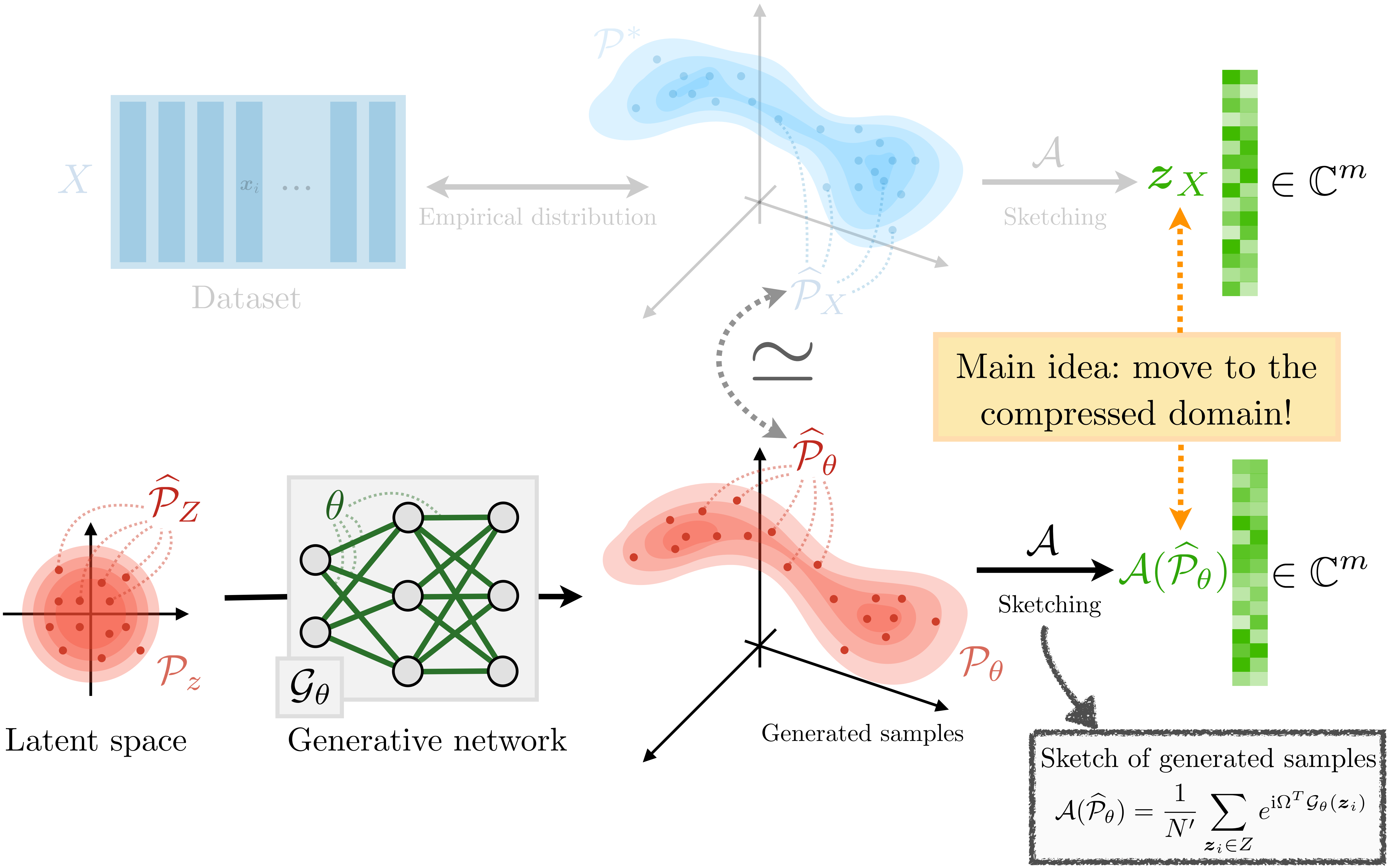
Main idea: move to the compressed domain!

Thanks to sketching

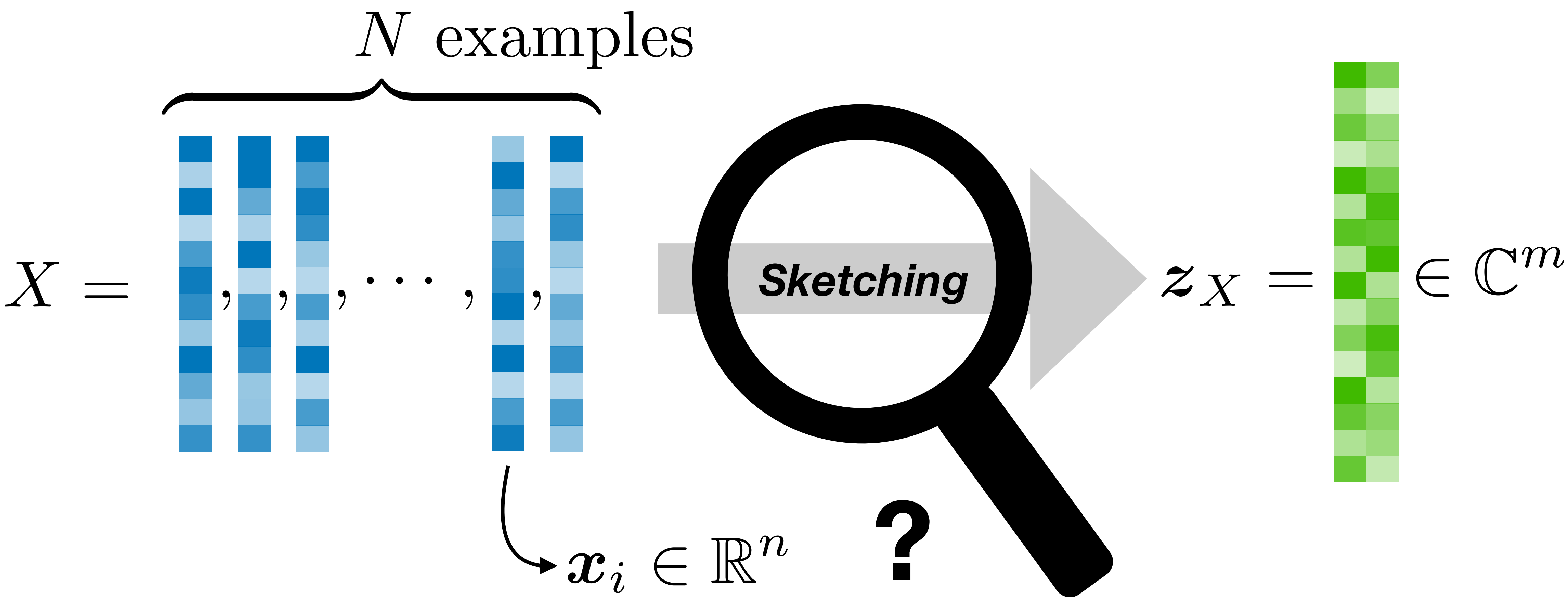
Compressively learning generative networks



Compressively learning generative networks

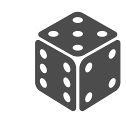


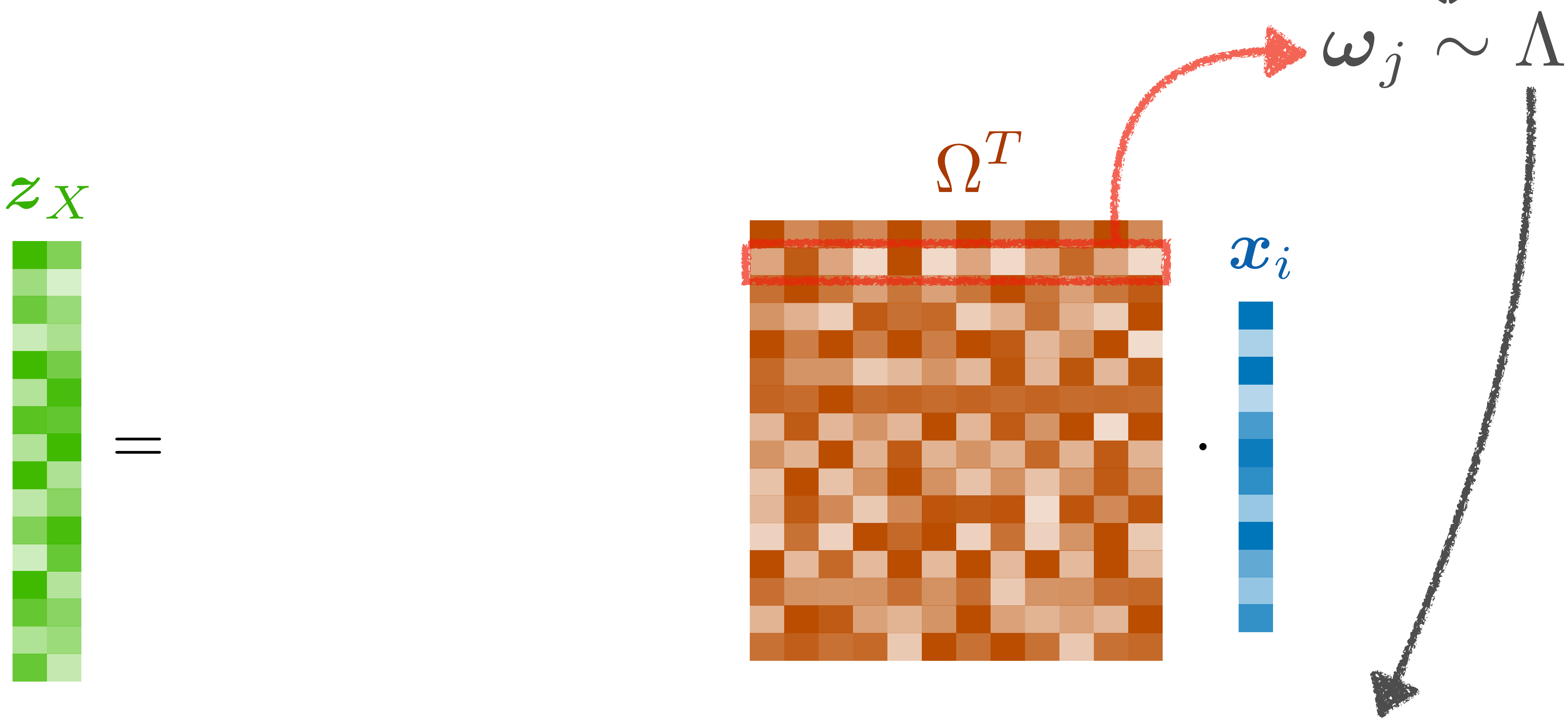
Sketching a dataset



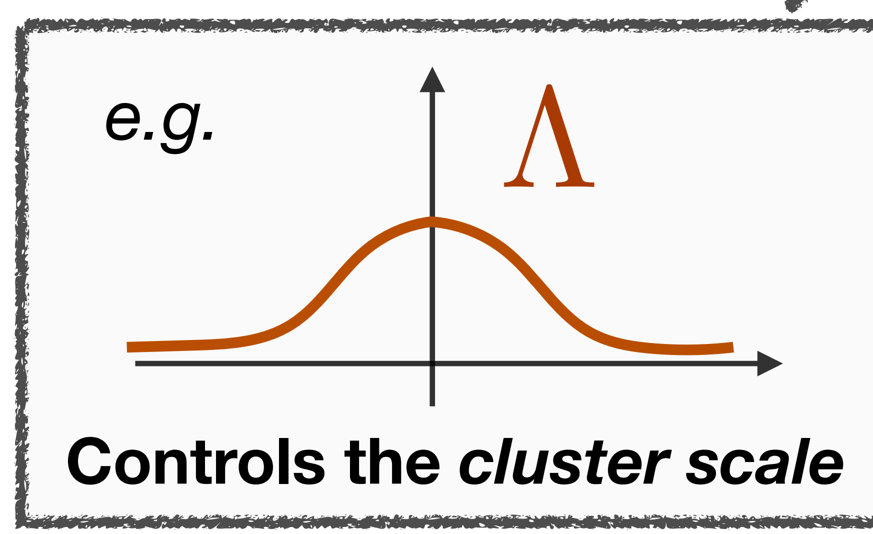
[Gribonval17]

Sketching a dataset





1. Project on m (random) vectors

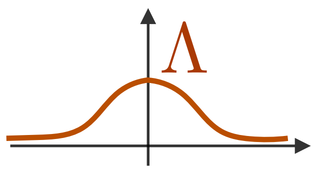


Sketching a dataset

$$\boldsymbol{z}_X = \exp\left(\mathrm{i}\, \Omega^T \cdot \boldsymbol{x}_i \right)$$

$\boldsymbol{\omega}_j \sim \Lambda$

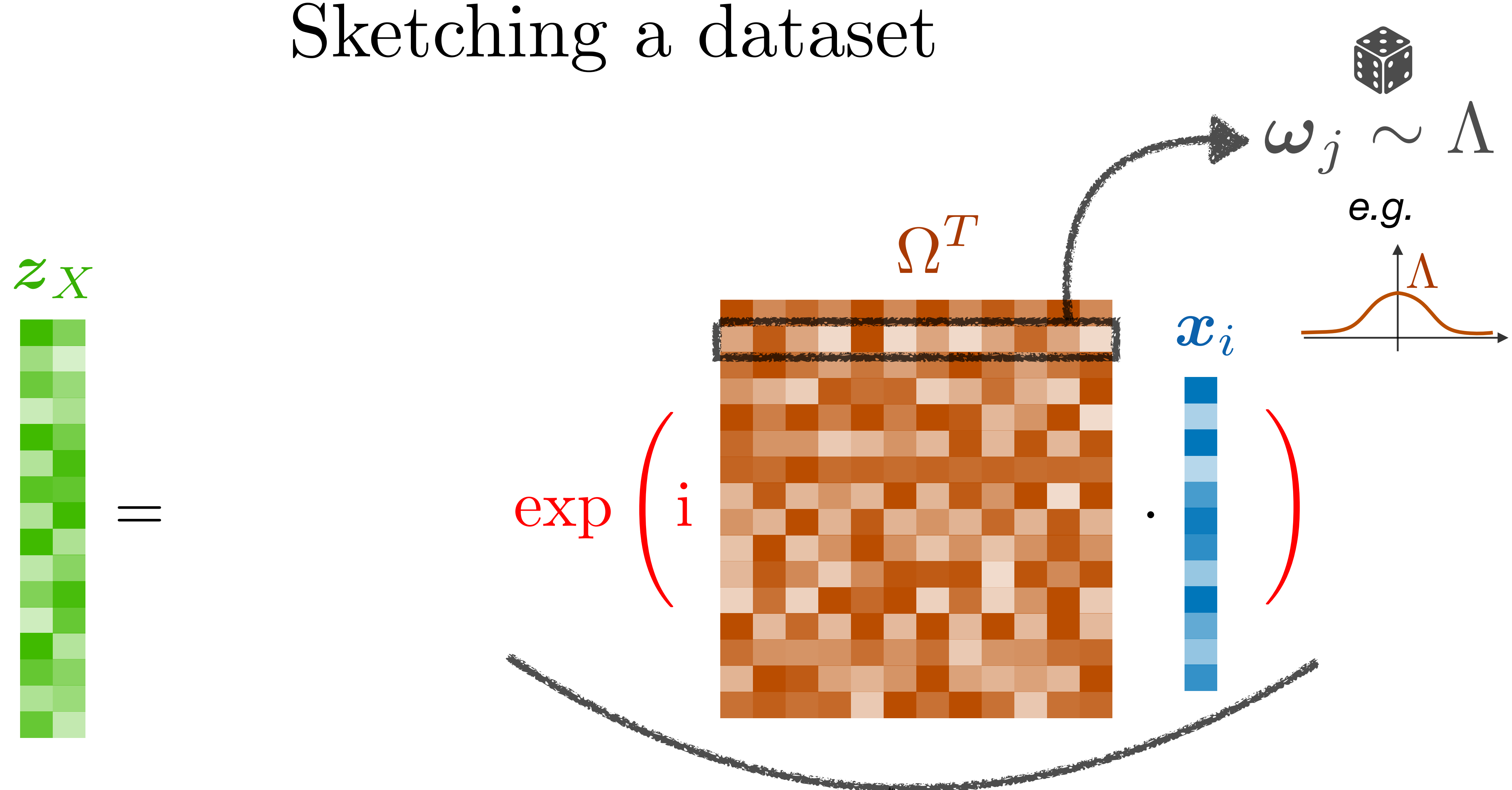
e.g.



1. Project on m (random) vectors

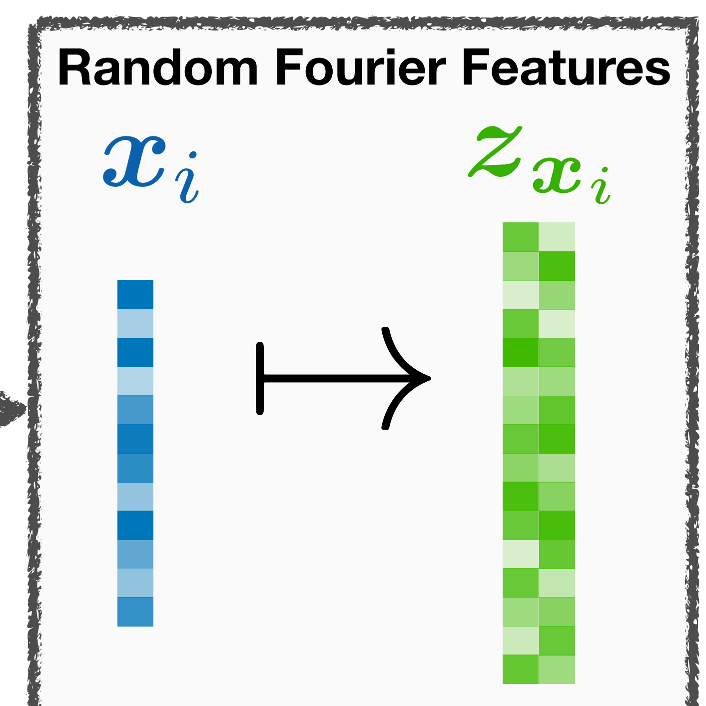
2. Nonlinear periodic signature function

Sketching a dataset



1. Project on m (random) vectors

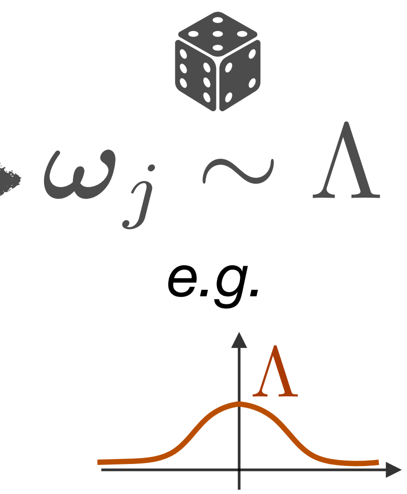
2. Nonlinear periodic signature function

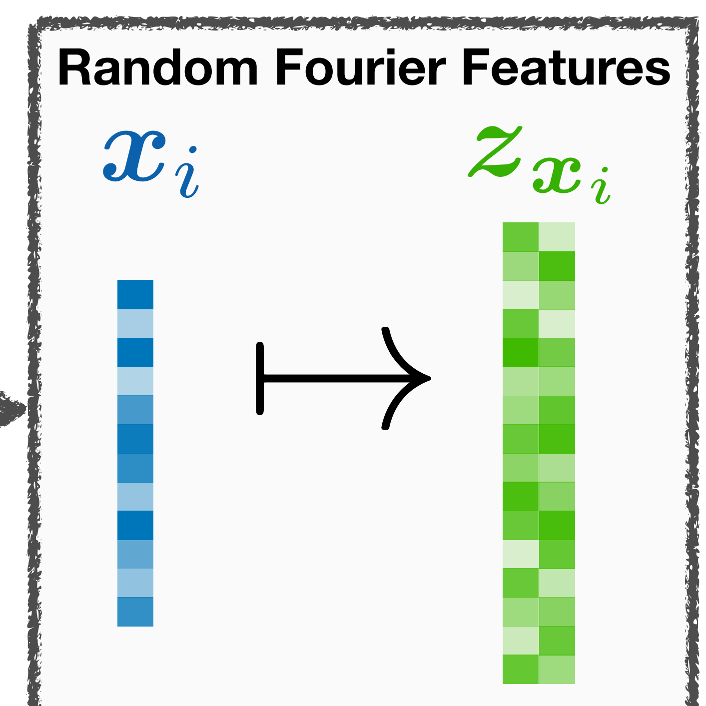


[A. Rahimi, B. Recht, "*Random Features for Large-scale Kernel Machines*",NIPS, 2007]

Sketching a dataset

$$z_X = \frac{1}{N} \sum_{x_i \in X} \exp\left(\mathrm{i}\, \Omega^T \cdot x_i \right)$$

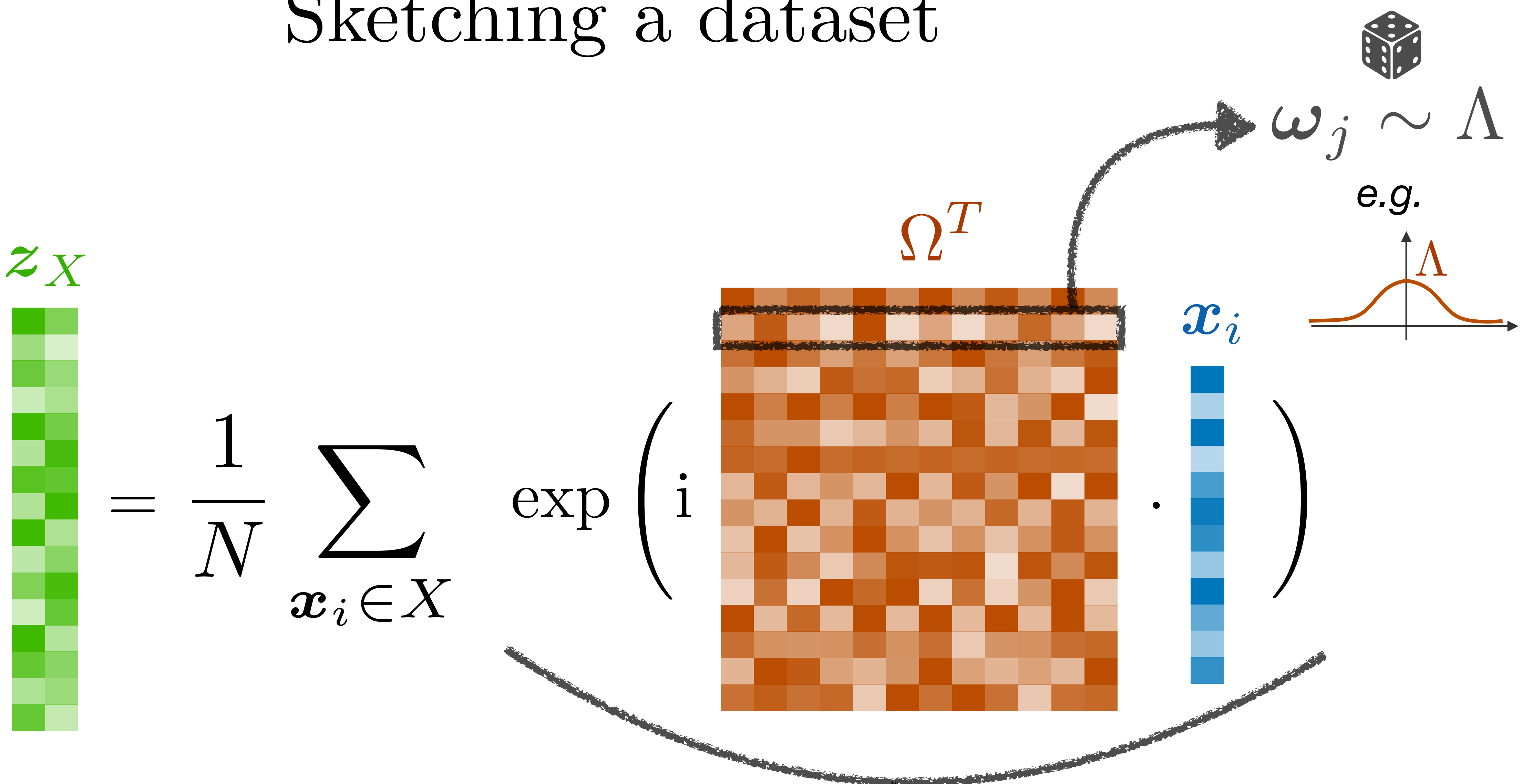




1. **Project on m (random) vectors**
2. **Nonlinear periodic signature function**
3. **Pooling (average)**

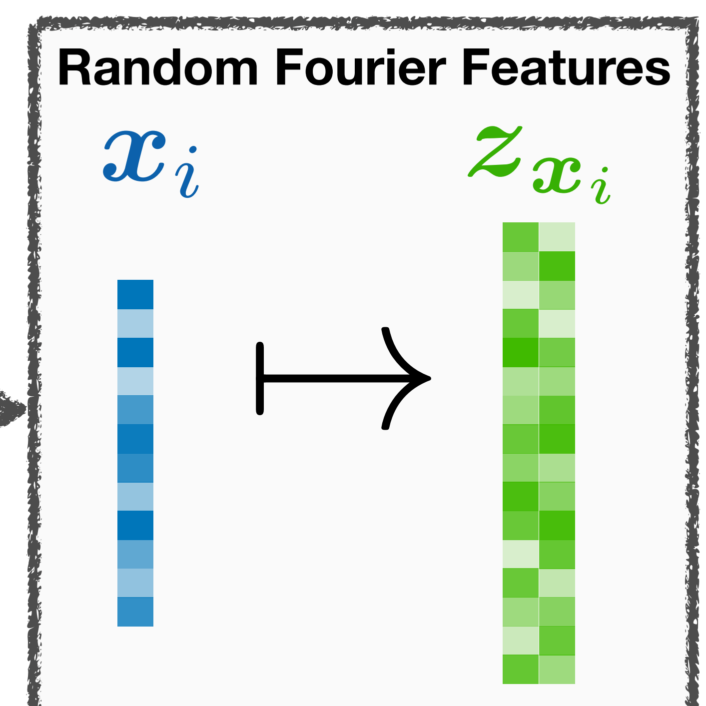
[A. Rahimi, B. Recht, "*Random Features for Large-scale Kernel Machines*",NIPS, 2007]

Sketching a dataset



1. **Project on m (random) vectors**
2. **Nonlinear periodic signature function**
3. **Pooling (average)**

$$z_X = \left[\frac{1}{N} \sum_{x_i \in X} e^{\mathrm{i}\omega_j^T x_i} \right]_{j=1}^{m} \in \mathbb{C}^m$$



[A. Rahimi, B. Recht, “*Random Features for Large-scale Kernel Machines*”,NIPS, 2007]

Compressively learning generative networks

Proposed approach: match the sketches of real and generated data

$$\min_{\theta} \left\| \boldsymbol{z}_X - \frac{1}{N'} \sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\Omega^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \right\|_2^2$$

Compressively learning generative networks

Proposed approach: match the sketches of real and generated data

$$\min_{\theta} \left\| \boldsymbol{z}_X - \frac{1}{N'} \sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\Omega^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \right\|_2^2$$

$\simeq$

Sampled (Monte Carlo)
estimation to the MMD!

$\boldsymbol{\omega}_j \sim \Lambda$

$$\min_{\theta} \mathbb{E}_{\boldsymbol{\omega} \sim \Lambda} \left| \frac{1}{N} \sum_{\boldsymbol{x}_i \in X} e^{\mathrm{i}\boldsymbol{\omega}^T \boldsymbol{x}_i} - \frac{1}{N'} \sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\boldsymbol{\omega}^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \right|^2$$

...but where dataset is accessed only once 👍

Compressively learning generative networks

Proposed approach: match the sketches of real and generated data

$$\min_{\theta} \left\| \boldsymbol{z}_X - \frac{1}{N'} \sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\Omega^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \right\|_2^2$$

$\simeq$

Sampled (Monte Carlo) estimation to the MMD!

$\boldsymbol{\omega}_j \sim \Lambda$

$$\min_{\theta} \mathbb{E}_{\boldsymbol{\omega} \sim \Lambda} \left| \frac{1}{N} \sum_{\boldsymbol{x}_i \in X} e^{\mathrm{i}\boldsymbol{\omega}^T \boldsymbol{x}_i} - \frac{1}{N'} \sum_{\boldsymbol{z}_i \in Z} e^{\mathrm{i}\boldsymbol{\omega}^T \mathcal{G}_\theta(\boldsymbol{z}_i)} \right|^2$$

...but where dataset is accessed only once

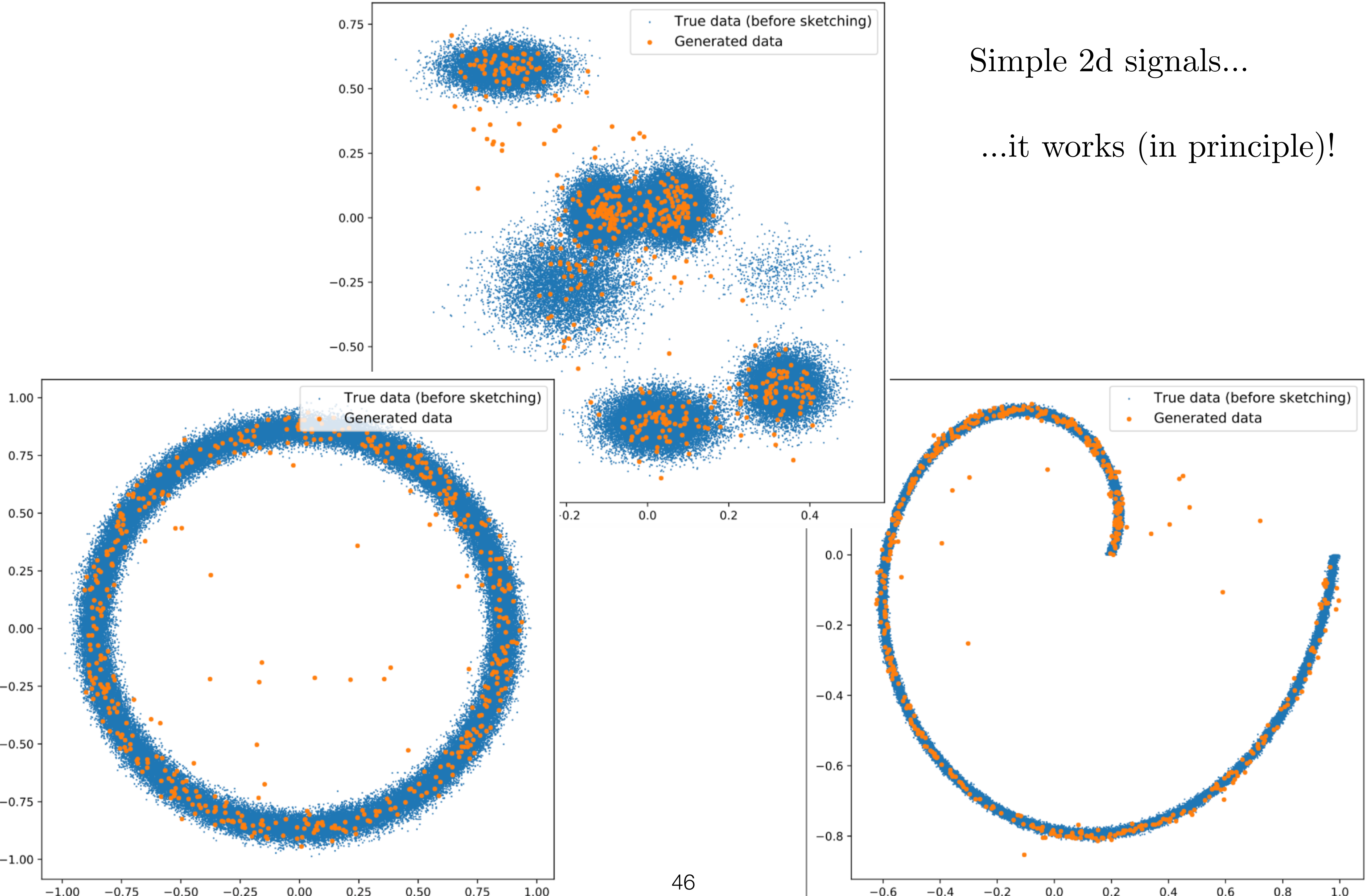
Practical learning algorithm?

Differentiable by chain rule (feat. backprop)

$$\nabla_\theta \widehat{\mathcal{L}}(\theta; \boldsymbol{z}_X) = -2 \cdot \frac{1}{n'} \sum_{i=1}^{n'} \Re \left[\boldsymbol{r}^H \left(\left. \frac{\partial \Phi(\boldsymbol{u})}{\partial \boldsymbol{u}} \right|_{\boldsymbol{u} = \mathcal{G}_\theta(\boldsymbol{z}_i)} \cdot \frac{\partial \mathcal{G}_\theta(\boldsymbol{z}_i)}{\partial \theta} \right) \right]$$

$\mathcal{O}(N')$

Results: preliminary toy example



Simple 2d signals...

...it works (in principle)!

Results (from other authors)

Differentially Private Mean Embeddings with Random Features for Synthetic Data Generation

Frederik Harder[*,1,2], **Kamil Adamczewski**[*,1,3], **Mijung Park**[1,2]
[*]Equal Contribution
[1]Max Planck Institute for Intelligent Systems
[2]University of Tübingen
[3]ETH Zürich
{fharder|kadamczewski|mpark}@tue.mpg.de

Submission history
From: Frederik Harder [view email]
[v1] Wed, 26 Feb 2020 16:41:41 UTC (1,629 KB)

Abstract

We present a differentially private data generation paradigm using random feature representations of kernel mean embeddings when comparing the distribution of true data with that of synthetic data. We exploit the random feature representations for two important benefits. First, we require a very low privacy cost for training deep generative models. This is because unlike kernel-based distance metrics that require computing the kernel matrix on all pairs of true and synthetic data points, we can detach the data-dependent term from the term solely dependent on synthetic data. Hence, we ne
and then use it until the end
analytic sensitivity of the ker
bounded by construction. Th
for a clipping norm to handle
provide several variants of ou
with random features (DP-M
for datasets such as heteroge
achieves better privacy-utilit
several datasets.

- Use the same cost function
- Rely on privacy-friendly benefits of sketching
- Rely on pre-trained autoencoders to handle images

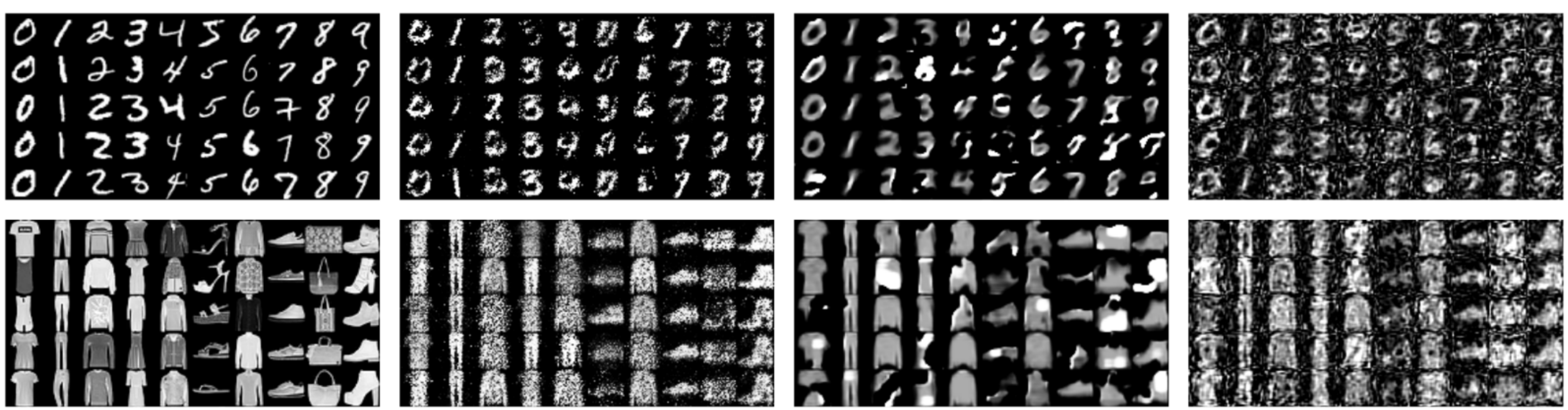


Figure 1: Generated samples with different levels of privacy

To conclude: open challenges