“Creative” Neural Networks

The Hitchhiker’s Guide to ~~Deepfakes~~ Generative Modelling

Artem Maevskiy
Research Fellow – Institute for Functional Intelligent Materials @ National University of Singapore

MLHEP-2023
Erice, Apr 11 – 18, 2023

Practicum

Outline

- Intro:
  - Generative modelling: what it is and how it’s useful
  - Generative modelling: how it’s done
- Generative Adversarial Networks (GAN)
- Variational Autoencoders (VAE)
- Normalizing Flows (NF)
- Evaluating Generative Models

Intro to Generative modelling

1. What it is and how it’s useful

Progress (as of 2018…)

2014

2015

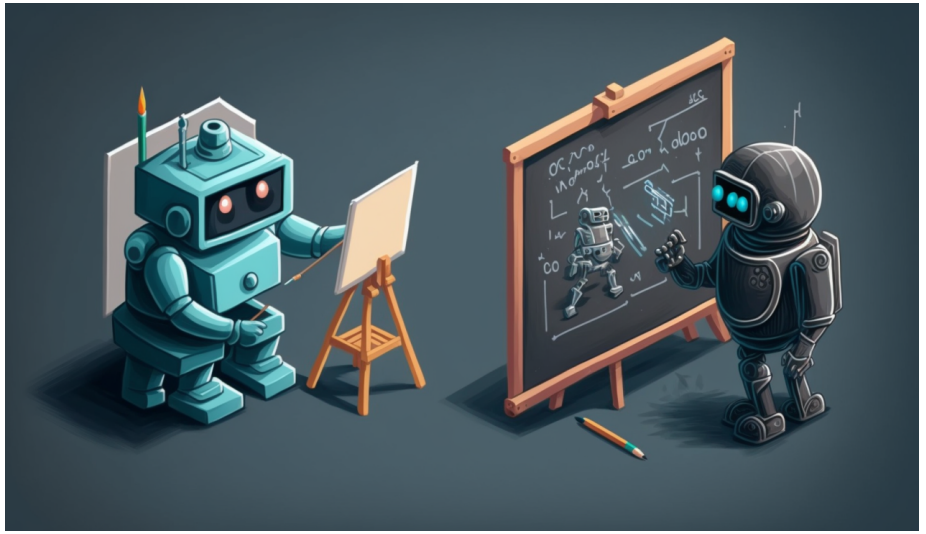
2016

2017

2018

https://twitter.com/goodfellow_ian/status/1084973596236144640

Progress (text-to-image, 2023)

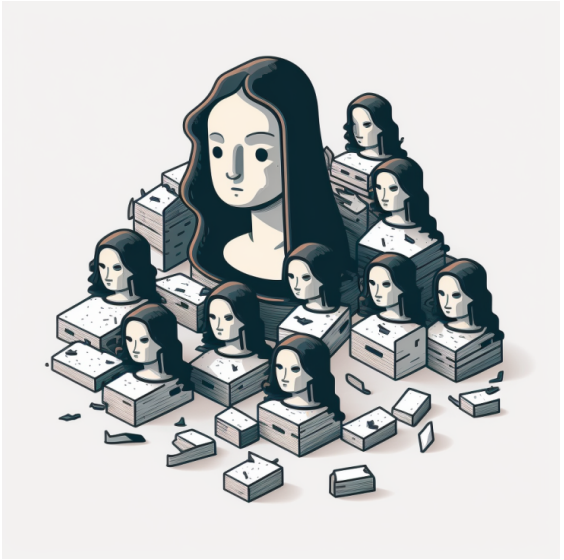


“Robot artist painting a picture and another robot mathematician writing formulas on a blackboard, isometric cartoon” – as interpreted by Midjourney

“Isometric clumsy black-box robot painter at work, minimalistic diagram with white background” – as interpreted by Midjourney

What a generative model does

Black-box model

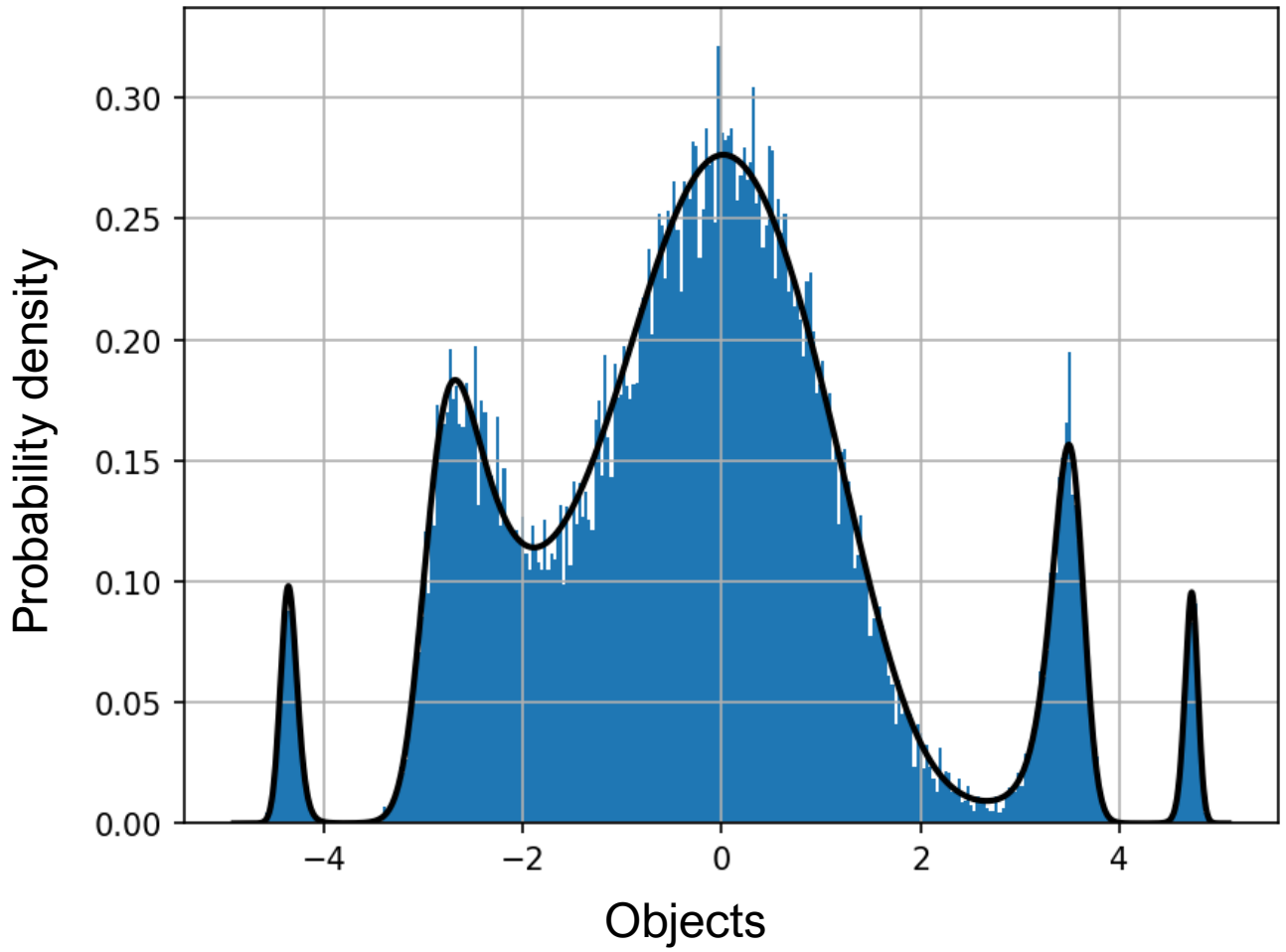


Objects

“Black box with gears and switches, isometric drawing, minimalistic, cartoon, white background” – as interpreted by Midjourney

“A pile of Mona Lisa drawings, minimalistic cartoon, white background, isometric” – as interpreted by Midjourney

What a generative model does



Parameters of the model control **the shape of the distribution**

May also depend on conditionals (model **maps conditionals to distributions**)

Learns distribution from data

Generative modelling in HEP

- Screenshot from https://github.com/iml-wg/HEPML-LivingReview
- Each entry = somebody's work

- Generative models / density estimation
  - GANs:
    - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis [DOI]
    - Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters [DOI]
    - CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks [DOI]
    - Image-based model parameter optimization using Model-Assisted Generative Adversarial Networks [DOI]
    - How to GAN Event Subtraction [DOI]
    - Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description [DOI]
    - How to GAN away Detector Effects [DOI]
    - 3D convolutional GAN for fast simulation
    - Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks [DOI]
    - Lund jet images from generative and cycle-consistent adversarial networks [DOI]
    - How to GAN LHC Events [DOI]
    - Machine Learning Templates for QCD Factorization in the Search for Physics Beyond the Standard Model [DOI]
    - DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC [DOI]
    - LHC analysis-specific datasets with Generative Adversarial Networks
    - Generative Models for Fast Calorimeter Simulation.LHCb case [DOI]
    - Deep generative models for fast shower simulation in ATLAS
    - Regressive and generative neural networks for scalar field theory [DOI]
    - Three dimensional Generative Adversarial Networks for fast simulation
    - Generative models for fast simulation
    - Unfolding with Generative Adversarial Networks
    - Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks [DOI]
    - Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks [DOI]
    - Generative models for fast cluster simulations in the TPC for the ALICE experiment
    - RICH 2018 [DOI]
    - GANs for generating EFT models [DOI]
    - Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network [DOI]
    - Reducing Autocorrelation Times in Lattice Simulations with Generative Adversarial Networks [DOI]
    - Tips and Tricks for Training GANs with Physics Constraints
    - Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters [DOI]
    - Next Generation Generative Neural Networks for HEP
    - Calorimetry with Deep Learning: Particle Classification, Energy Regression, and Simulation for High-Energy Physics
    - Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics [DOI]
    - Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed
    - AI-based Monte Carlo event generator for electron-proton scattering
    - DCTRGAN: Improving the Precision of Generative Models with Reweighting [DOI]
    - GANplifying Event Samples
    - Graph Generative Adversarial Networks for Sparse Data Generation in High Energy Physics
    - Simulating the Time Projection Chamber responses at the MPD detector using Generative Adversarial Networks
    - Explainable machine learning of the underlying physics of high-energy particle collisions
    - A Data-driven Event Generator for Hadron Colliders using Wasserstein Generative Adversarial Network [DOI]
    - Reduced Precision Strategies for Deep Learning: A High Energy Physics Generative Adversarial Network Use Case [DOI]
    - Validation of Deep Convolutional Generative Adversarial Networks for High Energy Physics Calorimeter Simulations
    - Compressing PDF sets using generative adversarial networks
    - Physics Validation of Novel Convolutional 2D Architectures for Speeding Up High Energy Physics Simulations
  - Autoencoders
    - Deep Learning as a Parton Shower
    - Deep generative models for fast shower simulation in ATLAS
    - Variational Autoencoders for Anomalous Jet Tagging
    - Variational Autoencoders for Jet Simulation
    - Foundations of a Fast, Data-Driven, Machine-Learned Simulator
    - Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network
    - Bump Hunting in Latent Space
    - {End-to-end Sinkhorn Autoencoder with Noise Generator
    - Graph Generative Models for Fast Detector Simulations in High Energy Physics
    - DeepRICH: Learning Deeply Cherenkov Detectors [DOI]
  - Normalizing flows
    - Flow-based generative models for Markov chain Monte Carlo in lattice field theory [DOI]
    - Equivariant flow-based sampling for lattice gauge theory [DOI]
    - Flows for simultaneous manifold learning and density estimation
    - Exploring phase space with Neural Importance Sampling [DOI]
    - Event Generation with Normalizing Flows [DOI]
    - i-flow: High-Dimensional Integration and Sampling with Normalizing Flows [DOI]
    - Anomaly Detection with Density Estimation [DOI]
    - Data-driven Estimation of Background Distribution through Neural Autoregressive Flows
    - SARM: Sparse Autoregressive Model for Scalable Generation of Sparse Images in Particle Physics [DOI]
    - Measuring QCD Splittings with Invertible Networks
    - Efficient sampling of constrained high-dimensional theoretical spaces with machine learning
  - Physics-inspired
    - JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics
    - Binary JUNIPR: an interpretable probabilistic model for discrimination [DOI]
    - Exploring the Possibility of a Recovery of Physics Process Properties from a Neural Network Model [DOI]
    - Explainable machine learning of the underlying physics of high-energy particle collisions
    - Symmetry meets AI
  - Mixture Models
    - Data Augmentation at the LHC through Analysis-specific Fast Simulation with Deep Learning
    - Mixture Density Network Estimation of Continuous Variable Maximum Likelihood Using Discrete Training Samples
  - Phase space generation
    - Efficient Monte Carlo Integration Using Boosted Decision
    - Exploring phase space with Neural Importance Sampling [DOI]
    - Event Generation with Normalizing Flows [DOI]
    - i-flow: High-Dimensional Integration and Sampling with Normalizing Flows [DOI]
    - Neural Network-Based Approach to Phase Space Integration [DOI]
    - VegasFlow: accelerating Monte Carlo simulation across multiple hardware platforms [DOI]
    - A Neural Resampler for Monte Carlo Reweighting with Preserved Uncertainties [DOI]
    - Improved Neural Network Monte Carlo Simulation [DOI]
    - Phase Space Sampling and Inference from Weighted Events with Autoregressive Flows [DOI]
    - How to GAN Event Unweighting
  - Gaussian processes
    - Modeling Smooth Backgrounds and Generic Localized Signals with Gaussian Processes
    - Accelerating the BSM interpretation of LHC data with machine learning [DOI]
    - $\textsf{Xsec}$: the cross-section evaluation code [DOI]
    - AI-optimized detector design for the future Electron-Ion Collider: the dual-radiator RICH case [DOI]

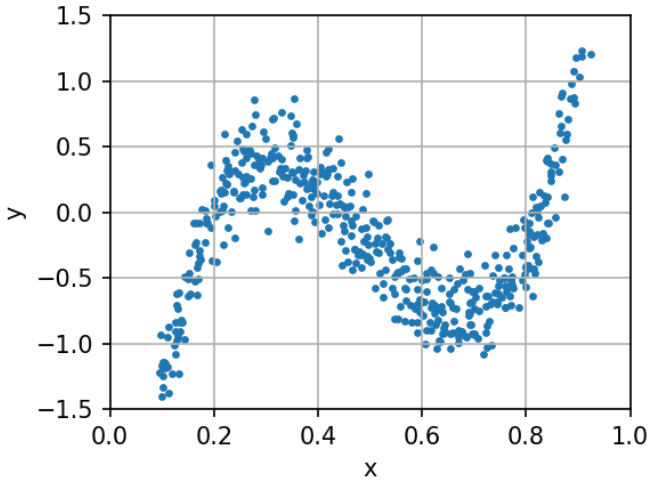
Intro to Generative modelling

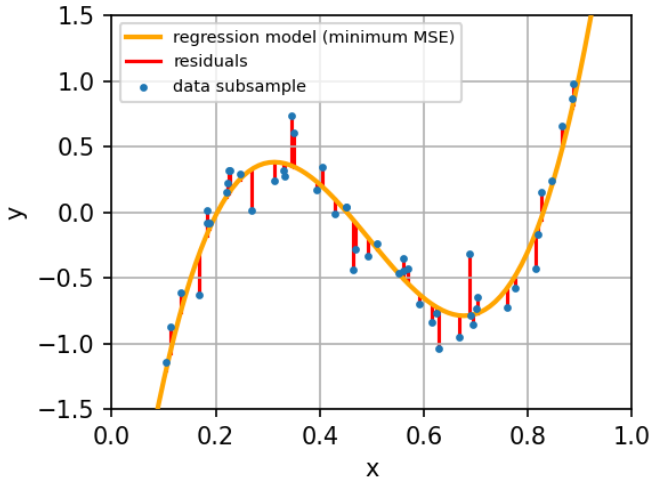
2. How it’s done (in a nutshell)

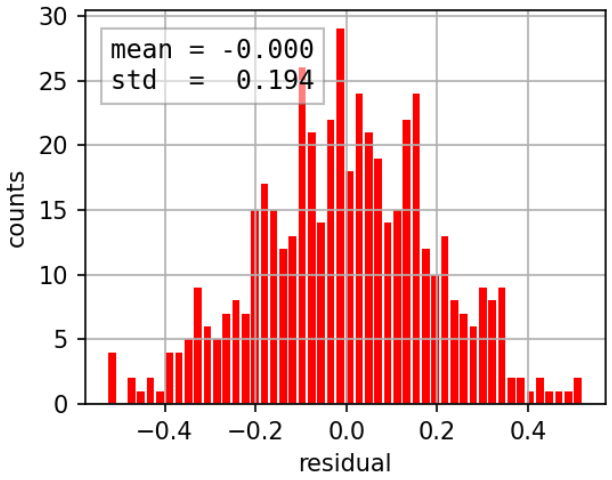
From supervised to generative

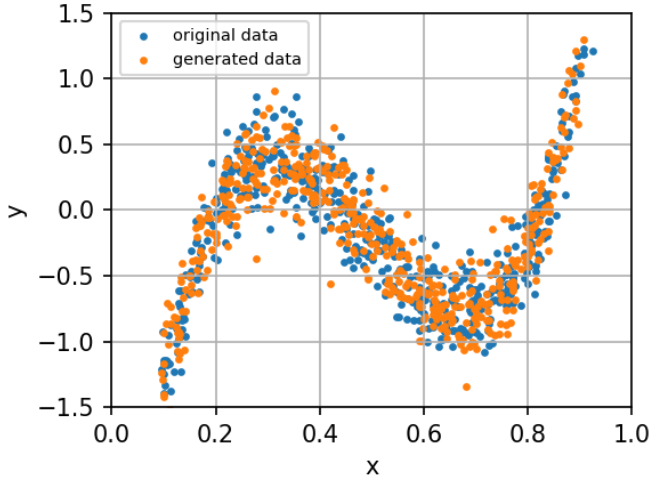
- A regression model $\hat{y} = \hat{f}_\theta(x)$ may be interpreted as a generative one, e.g.:

$$p^{\text{model}}(y|x) = \mathcal{N}\left(y\middle|\mu = \hat{f}_\theta(x), \sigma^2\right)$$









Defining PDF with a NN

- Straightforward way: can we just take a neural net $f_\theta : \mathbb{R}^d \to \mathbb{R}$ and treat is as some PDF $p_\theta(x) \equiv f_\theta(x)$?
  - Needs to be non-negative
    - can easily achieve this with a **proper activation at the last layer**
  - Needs to be normalized $\int f_\theta(x)dx = 1$
    - **how on earth do we satisfy this???**

Defining PDF with a NN

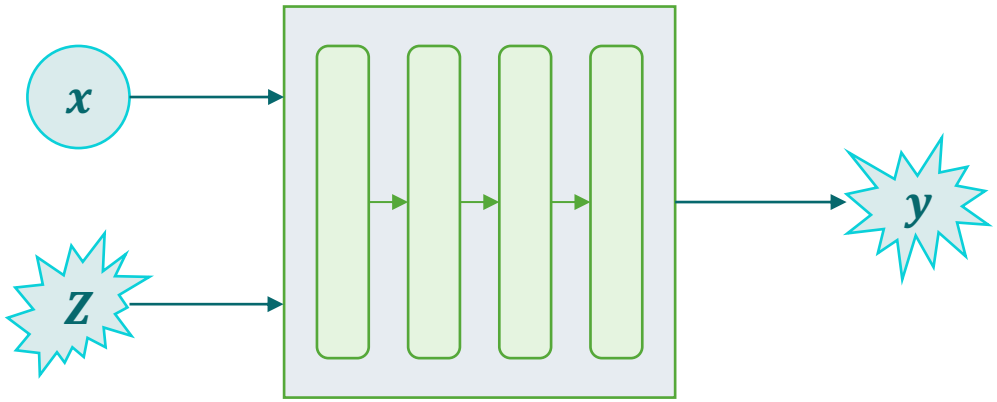
- A bit less straightforward way
  - $\tilde{p}_\theta(x) \equiv f_\theta(x)$ – the NN models the unnormalized PDF value
  - $p_\theta(x) = \frac{\tilde{p}_\theta(x)}{Z_\theta}$ – the true PDF
  - $Z_\theta = \int \tilde{p}_\theta(x')dx'$ – normalization constant (intractable)
- There exist methods to sample from and train such models
  - Computationally demanding
    - See, e.g., Metropolis-Hastings algorithm and Contrastive Divergence
    - Check out https://www.deeplearningbook.org/, chapters 16-18 for more info

Defining PDF with a NN

- Alternative:
  - make $\mathrm{y} = f_\theta(x)$ a stochastic function
  - treat output as samples from $p_\theta(y)$
    - $p_\theta(y)$ likely becomes intractable, but we may be ok with that

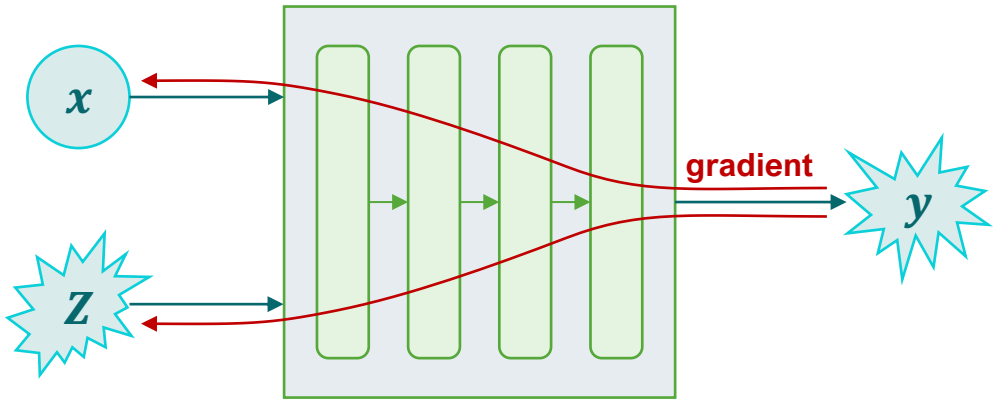
Introducing stochasticity into a NN

- $f_\theta \equiv f_\theta(x, Z)$
  - x – deterministic input (if needed)
  - Z – random variable of some known fixed distribution



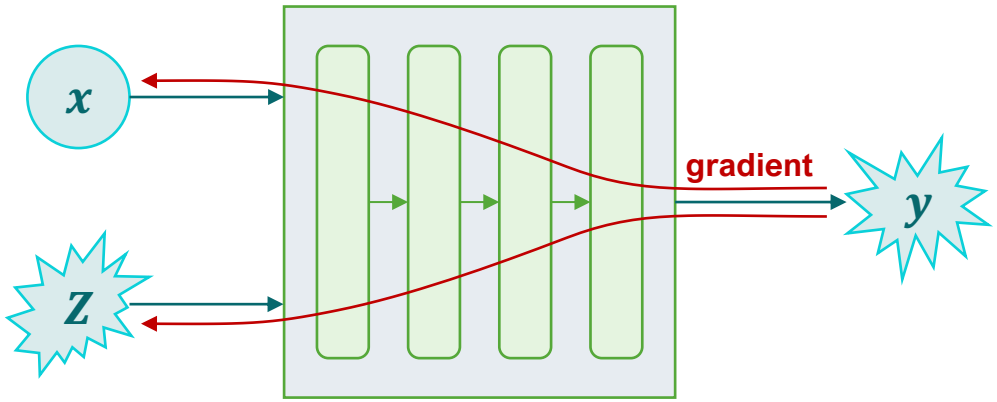
Introducing stochasticity into a NN

- $f_\theta \equiv f_\theta(x, Z)$
  - x – deterministic input (if needed)
  - Z – random variable of some known fixed distribution



Introducing stochasticity into a NN

- $f_\theta \equiv f_\theta(x, Z)$
  - x – deterministic input (if needed)
  - Z – random variable of some known fixed distribution



- How does one train this?

Comparing objects?

- Take some loss function $l(y, y')$ – "how similar y and y' are"
- Then, can't we just minimize this:

$$\mathcal{L} = \mathbb{E}_{y \sim p_{\text{data}},\, y' \sim p_\theta}[l(y, y')]$$

(by sampling y and y' and calculating averages)?

Comparing objects?

$$\min_{\theta} \mathcal{L} = \min_{\theta} \int p_{\theta}(y') \int p_{\text{data}}(y) l(y, y') dy \, dy'$$

Comparing objects?

$$\min_\theta \mathcal{L} = \min_\theta \int p_\theta(y') \int p_{\text{data}}(y) l(y, y') dy \, dy'$$

some function $F(y')$

$$= \min_\theta \int p_\theta(y') \, F(y') dy'$$

Comparing objects?

$$\min_\theta \mathcal{L} = \min_\theta \int p_\theta(y') \int p_{\text{data}}(y) l(y, y') dy \, dy'$$

some function $F(y')$

$$= \min_\theta \int p_\theta(y') \, F(y') dy'$$

- Minimized when $p_\theta(y') \equiv \delta(y' - \operatorname{argmin} F)$
- Such model will produce the same sample all the time

Comparing objects?

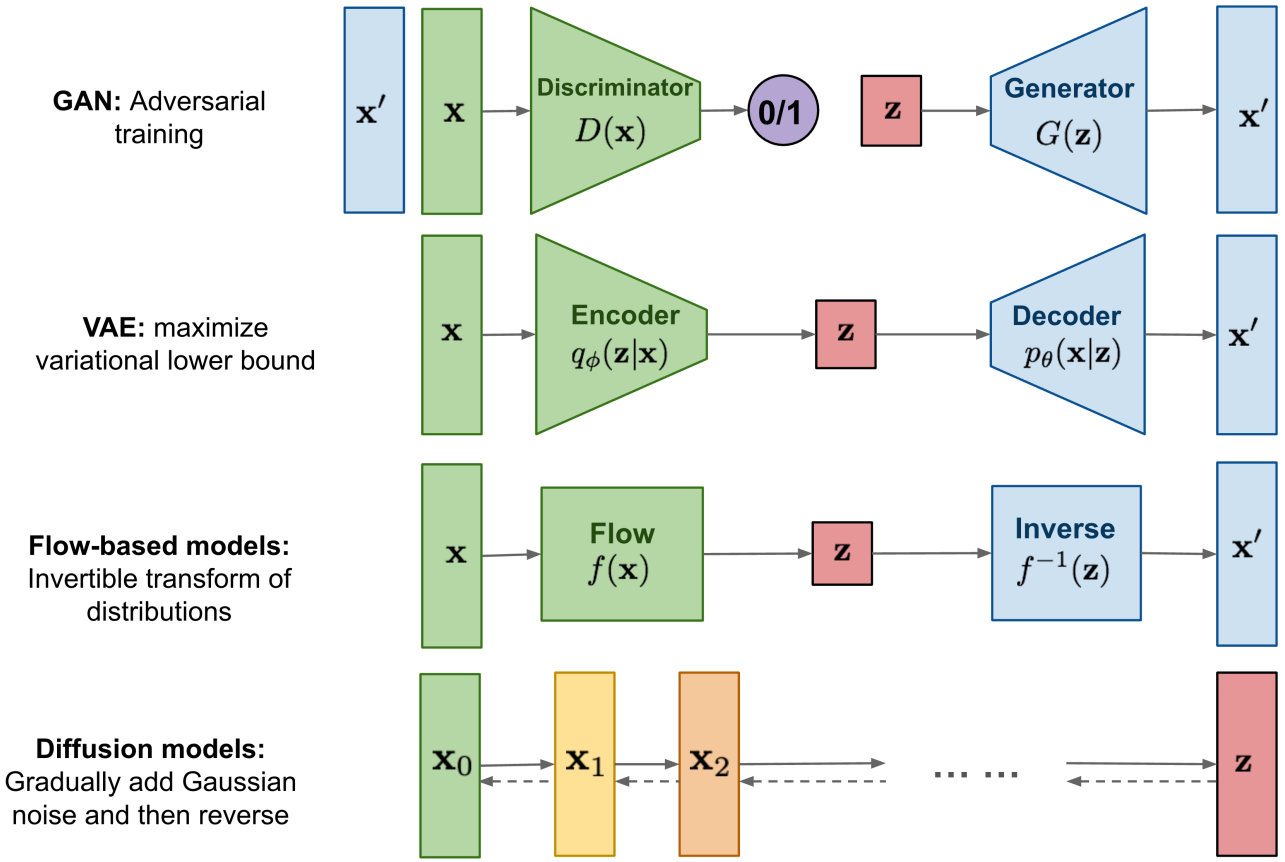
$$\min_\theta \mathcal{L} = \min_\theta \int p_\theta(y') \int p_{\text{data}}(y) l(y, y') dy \, dy'$$

some function $F(y')$

$$= \min_\theta \int p_\theta(y') \, F(y') dy'$$

- Minimized when $p_\theta(y') \equiv \delta(y' - \operatorname{argmin} F)$
- Such model will produce the same sample all the time
- Comparing objects is not enough; **we need to compare distributions**

Approaches to training generative models



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

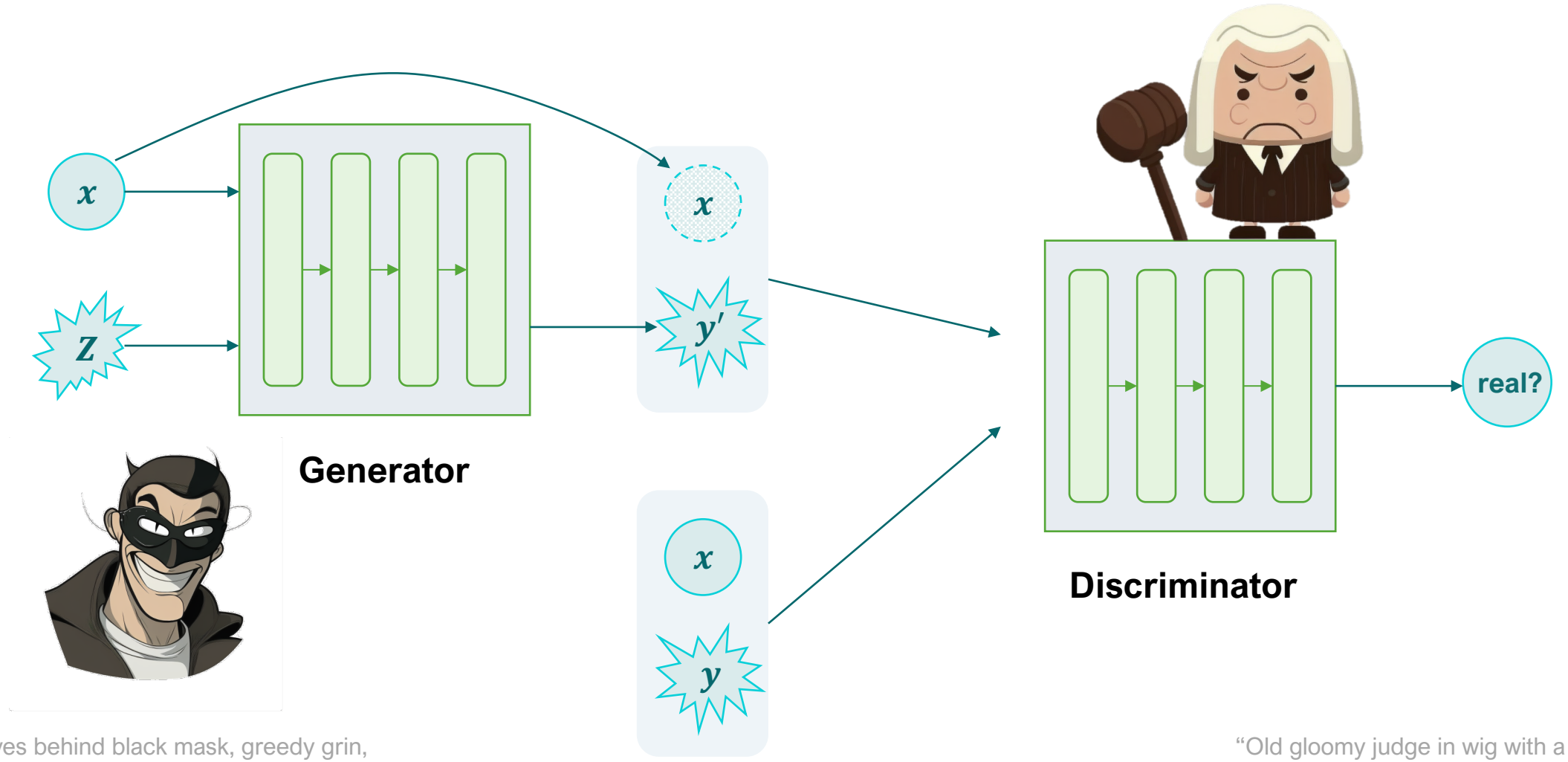
Generative Adversarial Networks

Our setting so far

- Training sample $\{y_1, y_2, \ldots, y_N\}$ with i.i.d. $y_i \sim p_{\text{data}}$
- A model that can sample arbitrary many $y' \sim p_\theta$
- Ability to calculate gradients $\frac{\partial y'_j}{\partial \theta}$ for any given generated sample j

- We want to find θ such that $p_\theta \approx p_{\text{data}}$

Adversarial approach



"Sly burglar, eyes behind black mask, greedy grin, white teeth, silly 2d cartoon, white background" – as interpreted by Midjourney

"Old gloomy judge in wig with a wooden judge hammer, front portrait view, minimalistic 2d cartoon, white background" – as interpreted by Midjourney

Mathematical formulation

- Objective function:

$$\mathcal{L} = \mathbb{E}_{y \sim p_{\text{data}}}\left[\log D_\psi(y)\right] + \mathbb{E}_{y' \sim p_\theta}\left[\log\left(1 - D_\psi(y')\right)\right]$$

  - ψ – parameters of the discriminator
  - θ – parameters of the generator
- Min-max problem:
  - $\mathcal{L} \rightarrow \max_{\psi}, \min_{\theta}$

Mathematical formulation

Note: the paper uses "x" for the target objects

We used "y" for the objects to allow for features "x" in case of conditional generation

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

https://arxiv.org/abs/1406.2661 (original GAN paper)

Problems

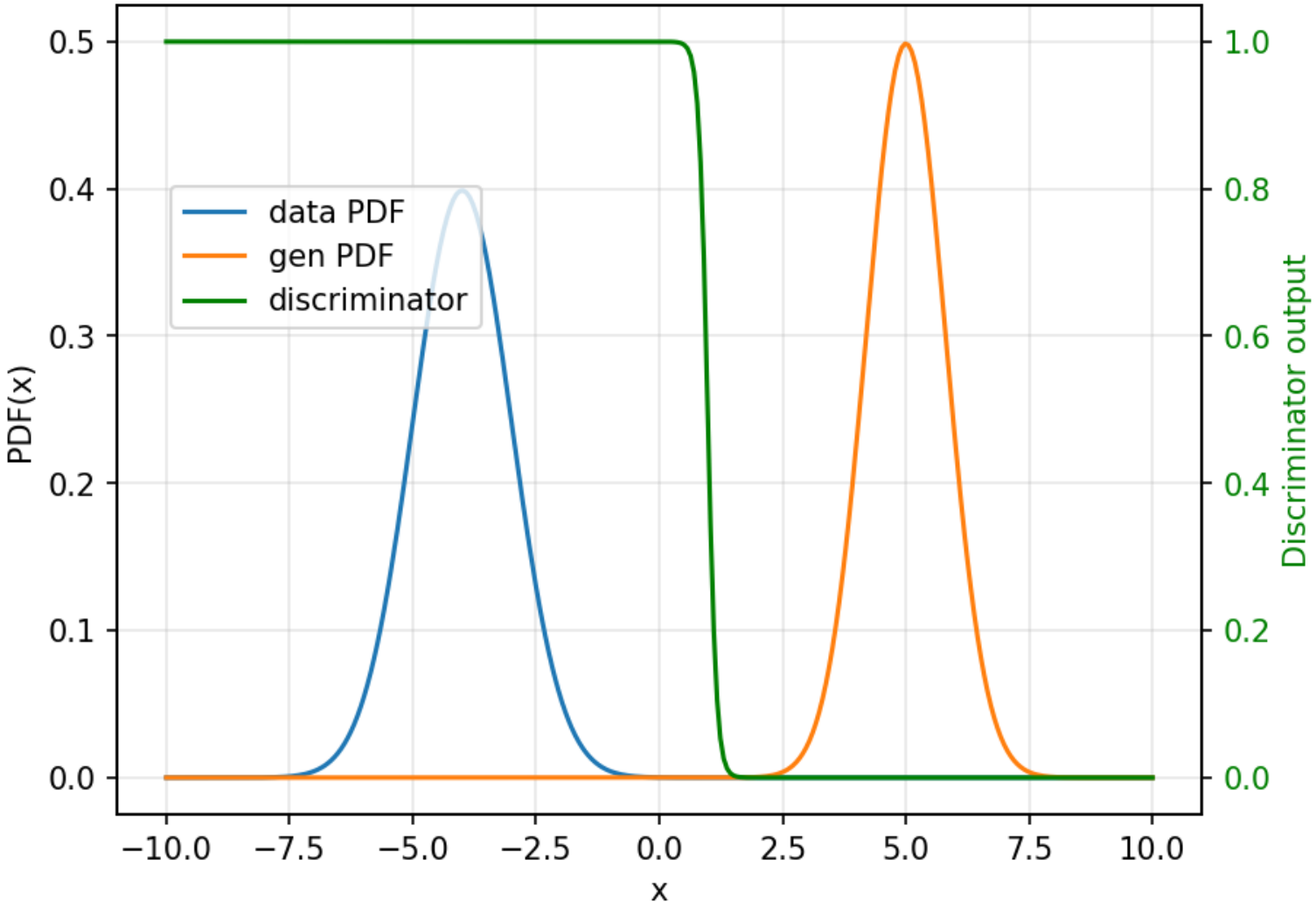
- Optimal discriminator:

$$\underset{D_\psi}{\operatorname{argmax}}\,\mathcal{L} = \underset{D_\psi}{\operatorname{argmax}} \int \left[p_{\text{data}}(y) \log D_\psi(y) + p_\theta(y) \log\left(1 - D_\psi(y)\right)\right] dy$$

$$= \underset{D_\psi}{\operatorname{argmax}} \underbrace{\left[p_{\text{data}}(y) \log D_\psi(y) + p_\theta(y) \log\left(1 - D_\psi(y)\right)\right]}_{\ell\left(D_\psi(y)\right)}$$

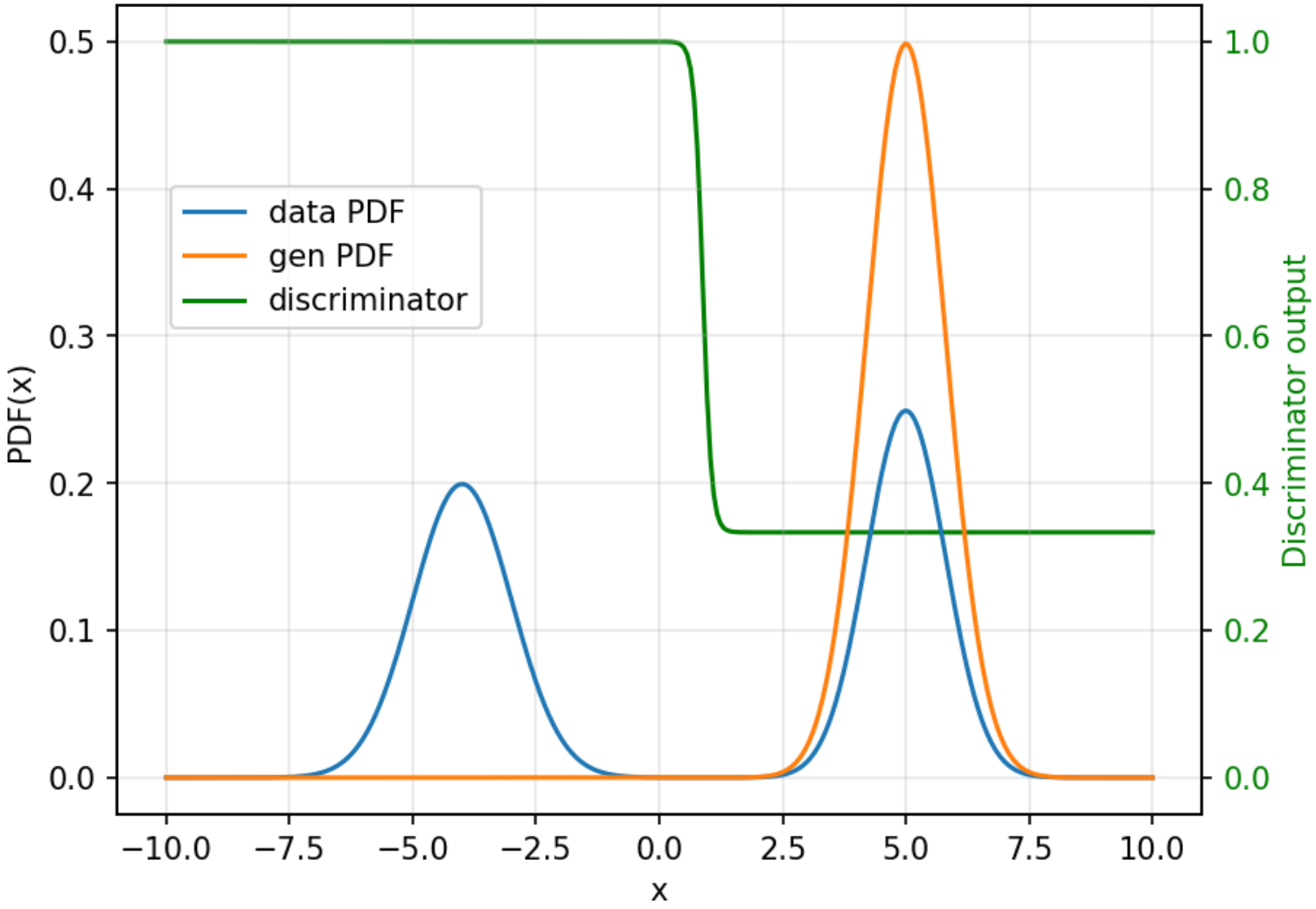
$$\frac{\partial}{\partial D_\psi(y)}\left[\ell\left(D_\psi(y)\right)\right] = \frac{p_{\text{data}}(y)}{D_\psi(y)} - \frac{p_\theta(y)}{1 - D_\psi(y)} = 0 \;\Rightarrow\; \boxed{D_\psi(y) = \frac{p_{\text{data}}(y)}{p_{\text{data}}(y) + p_\theta(y)}}$$

Non-overlapping support



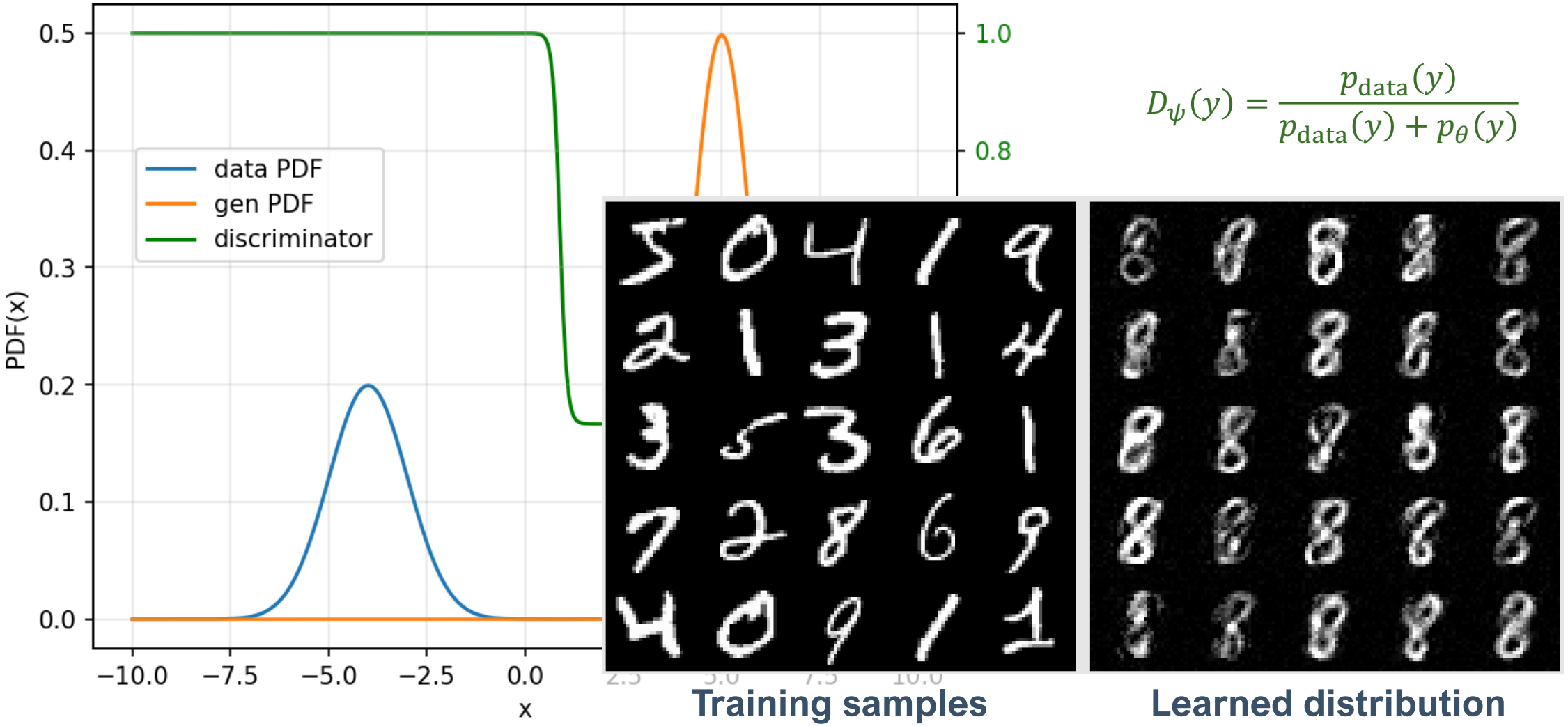
$$D_\psi(y) = \frac{p_{\text{data}}(y)}{p_{\text{data}}(y) + p_\theta(y)}$$

Partially overlapping support



$$D_\psi(y) = \frac{p_{\text{data}}(y)}{p_{\text{data}}(y) + p_\theta(y)}$$

Partially overlapping support



$$D_\psi(y) = \frac{p_{\text{data}}(y)}{p_{\text{data}}(y) + p_\theta(y)}$$

Training samples **Learned distribution**

KL and JSD

- With the “optimal” discriminator, we have our objective as:

$$\mathcal{L} = \mathbb{E}_{y \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(y)}{p_{\text{data}}(y) + p_\theta(y)} \right] + \mathbb{E}_{y' \sim p_\theta} \left[\log \frac{p_\theta(y)}{p_{\text{data}}(y) + p_\theta(y)} \right]$$

KL and JSD

- With the “optimal” discriminator, we have our objective as:

$$\mathcal{L} = \mathbb{E}_{y\sim p_{\text{data}}}\left[\log\frac{p_{\text{data}}(y)}{p_{\text{data}}(y)+p_\theta(y)}\right] + \mathbb{E}_{y'\sim p_\theta}\left[\log\frac{p_\theta(y)}{p_{\text{data}}(y)+p_\theta(y)}\right]$$

$$= -\log 4 + 2\cdot \text{JSD}(p_{\text{data}}\|p_\theta) \rightarrow \min_\theta$$

Jensen–Shannon divergence

$$\text{JSD}(p\|q) \equiv \frac{1}{2}\left[D_{\text{KL}}\left(p\middle\|\frac{p+q}{2}\right) + D_{\text{KL}}\left(q\middle\|\frac{p+q}{2}\right)\right]$$

KL and JSD

- With the “optimal” discriminator, we have our objective as:

$$\mathcal{L} = \mathbb{E}_{y\sim p_{\text{data}}}\left[\log\frac{p_{\text{data}}(y)}{p_{\text{data}}(y)+p_\theta(y)}\right] + \mathbb{E}_{y'\sim p_\theta}\left[\log\frac{p_\theta(y)}{p_{\text{data}}(y)+p_\theta(y)}\right]$$

$$= -\log 4 + 2\cdot \text{JSD}(p_{\text{data}}\|p_\theta) \rightarrow \min_\theta$$

Jensen–Shannon divergence

$$\text{JSD}(p\|q) \equiv \frac{1}{2}\left[D_{\text{KL}}\left(p\middle\|\frac{p+q}{2}\right) + D_{\text{KL}}\left(q\middle\|\frac{p+q}{2}\right)\right]$$

Kullback–Leibler divergence

$$D_{\text{KL}}(p\|q) = \mathbb{E}_{X\sim p}\left[\log\frac{p(X)}{q(X)}\right]$$

KL and JSD

- With the "optimal" discriminator, we have

$$\frac{\partial}{\partial\theta}\mathrm{JSD}(p_{\mathrm{data}}\|p_\theta) = 0$$

if p_{data} and p_θ do not overlap

$$\mathcal{L} = \mathbb{E}_{y\sim p_{\mathrm{data}}}\left[\log\frac{p_{\mathrm{data}}(y)}{p_{\mathrm{data}}(y)+p_\theta(y)}\right] + \mathbb{E}_{y'\sim p_\theta}\left[\log\frac{p_\theta(y)}{p_{\mathrm{data}}(y)+p_\theta(y)}\right]$$

$$= -\log 4 + 2\cdot\mathrm{JSD}(p_{\mathrm{data}}\|p_\theta) \to \min_\theta$$

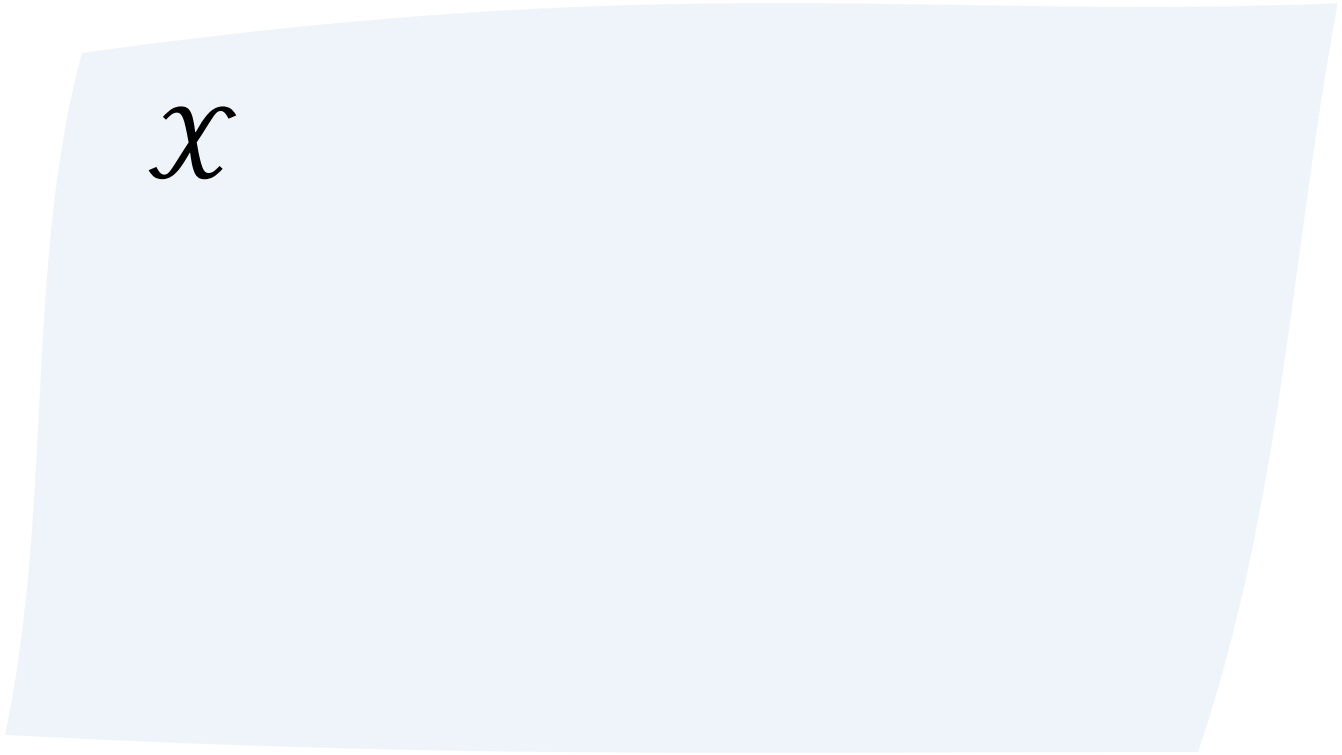
Jensen–Shannon divergence

$$\mathrm{JSD}(p\|q) \equiv \frac{1}{2}\left[D_{\mathrm{KL}}\left(p\middle\|\frac{p+q}{2}\right) + D_{\mathrm{KL}}\left(q\middle\|\frac{p+q}{2}\right)\right]$$

Kullback–Leibler divergence

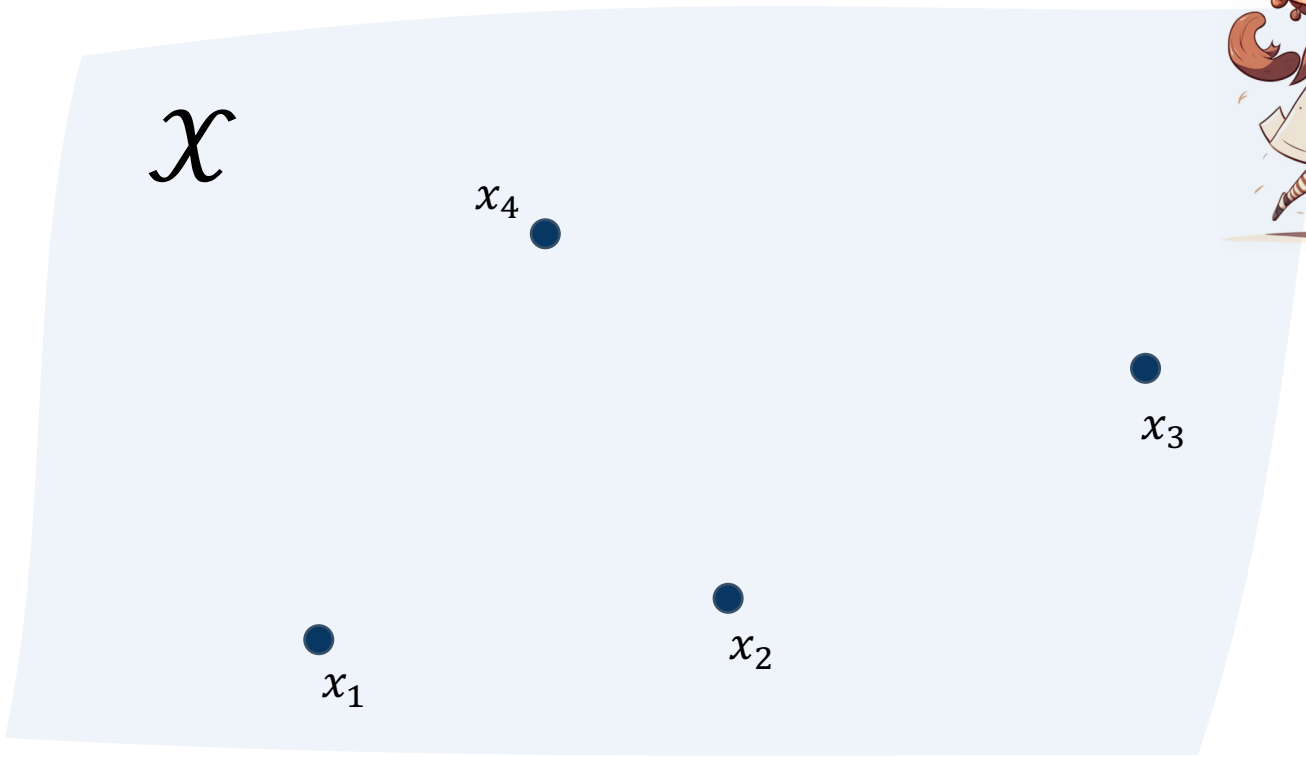
$$D_{\mathrm{KL}}(p\|q) = \mathbb{E}_{X\sim p}\left[\log\frac{p(X)}{q(X)}\right]$$

Optimal transport



Optimal transport

$p(x)$ – amount of pastry produced at x



● **bakery**

"Cozy bakery, silly 2d cartoon, white background" – as interpreted by Midjourney

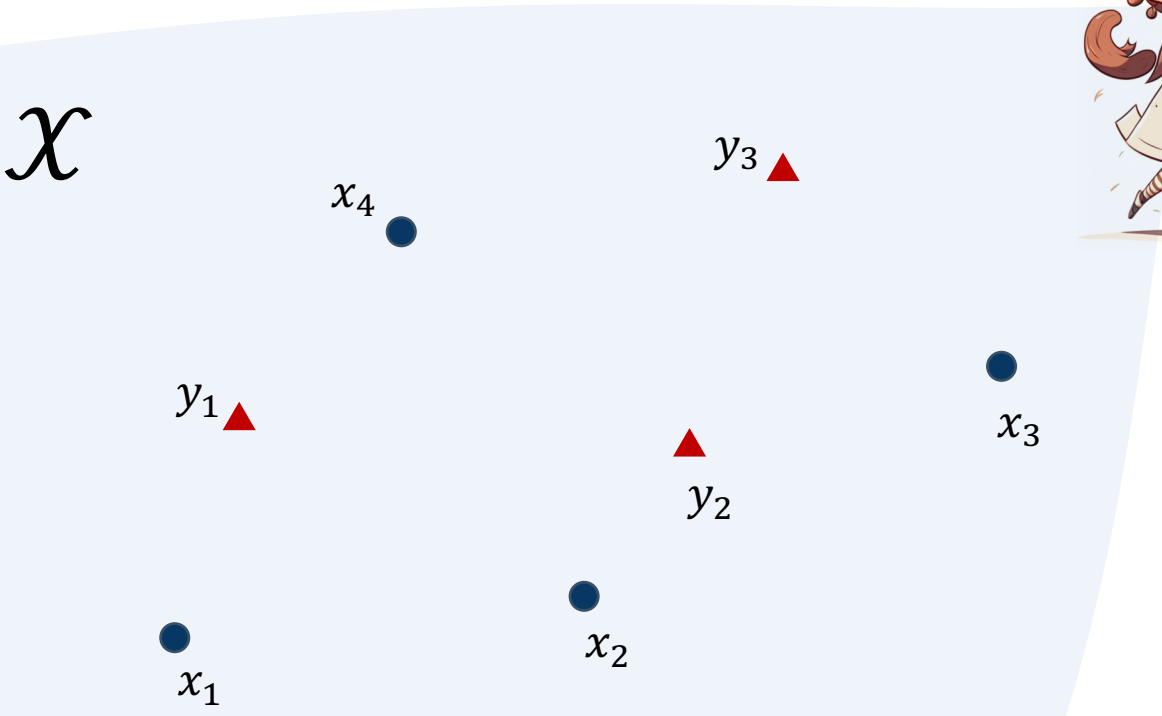
Optimal transport

$p(x)$ – amount of pastry produced at x

$q(y)$ – amount of pastry consumed at y

● bakery

▲ café



"Cozy patisserie cafe, happy customers, silly 2d cartoon, white background" – as interpreted by Midjourney

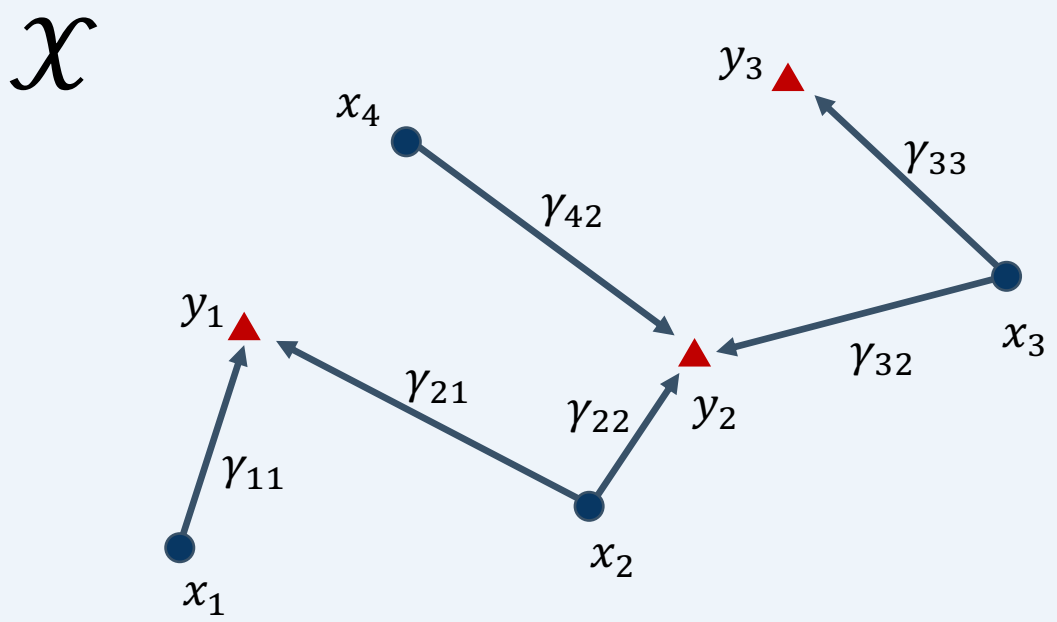
"Cozy bakery, silly 2d cartoon, white background" – as interpreted by Midjourney

Optimal transport

$p(x)$ – amount of pastry produced at x

$q(y)$ – amount of pastry consumed at y

▲ **café**



● **bakery**

$\gamma(x, y)$ – amount of pastry transported from x to y

"Cozy patisserie cafe, happy customers, silly 2d cartoon, white background" – as interpreted by Midjourney

"Cozy bakery, silly 2d cartoon, white background" – as interpreted by Midjourney

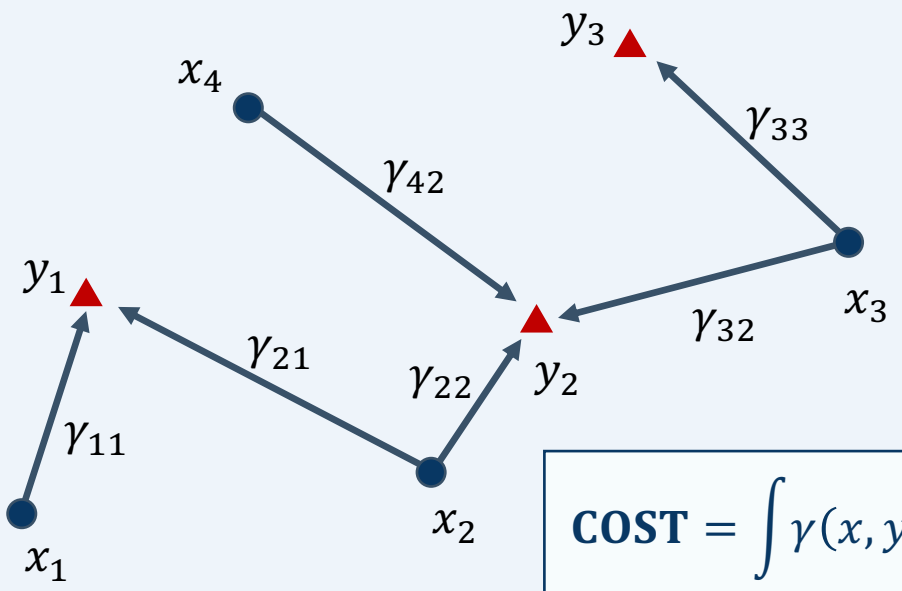
Optimal transport

$p(x)$ – amount of pastry produced at x

$q(y)$ – amount of pastry consumed at y

▲ **café**

X



● **bakery**

$\gamma(x, y)$ – amount of pastry transported from x to y

$$\textbf{COST} = \int \gamma(x,y)\|x-y\|dxdy = \mathbb{E}_\gamma\|x-y\| \to \min_\gamma$$

"Cozy patisserie cafe, happy customers, silly 2d cartoon, white background" – as interpreted by Midjourney

"Cozy bakery, silly 2d cartoon, white background" – as interpreted by Midjourney

Optimal transport

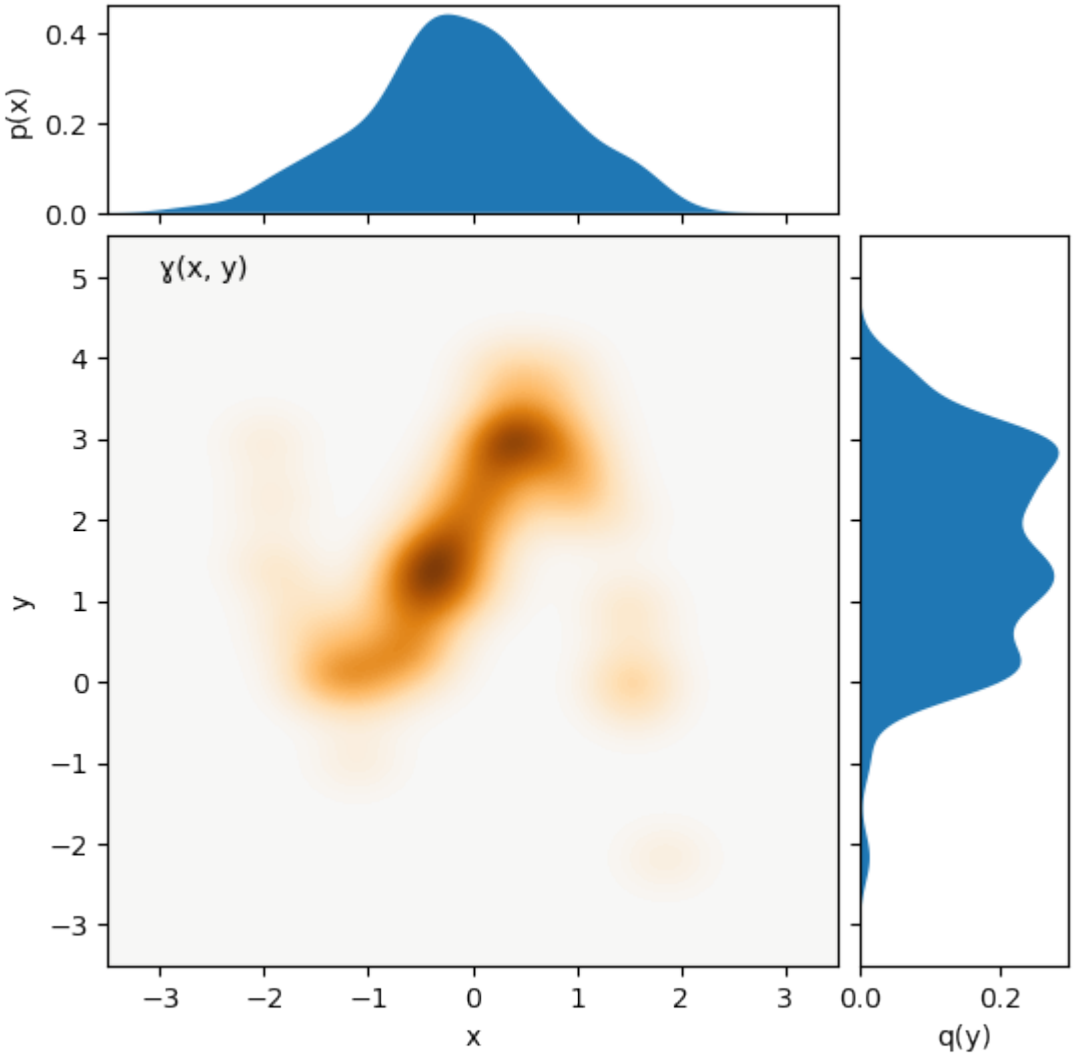
- Set of all valid plans:

$$\Pi(p,q) \equiv \left\{ \gamma(x,y) \quad : \quad \begin{array}{c} \int_X \gamma(x,y)dx = q(y) \\ \int_X \gamma(x,y)dy = p(x) \\ \int_{X\times X} \gamma(x,y)dxdy = 1 \\ \gamma(x,y) \geq 0 \end{array} \right\}$$

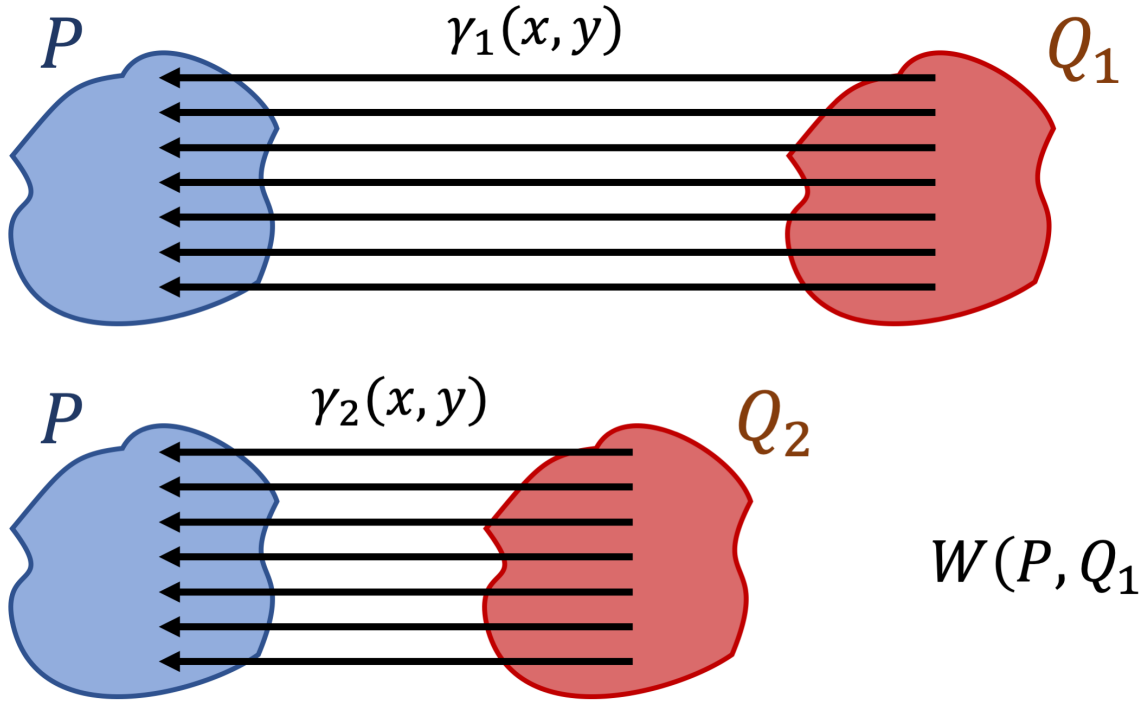
Optimal transport

- Earth Mover's distance, a.k.a. Wasserstein-1 distance:

$$W(p,q) \equiv \inf_{\gamma \in \Pi(p,q)} \left[\mathbb{E}_{\gamma} \|x - y\| \right]$$



Non-overlapping supports



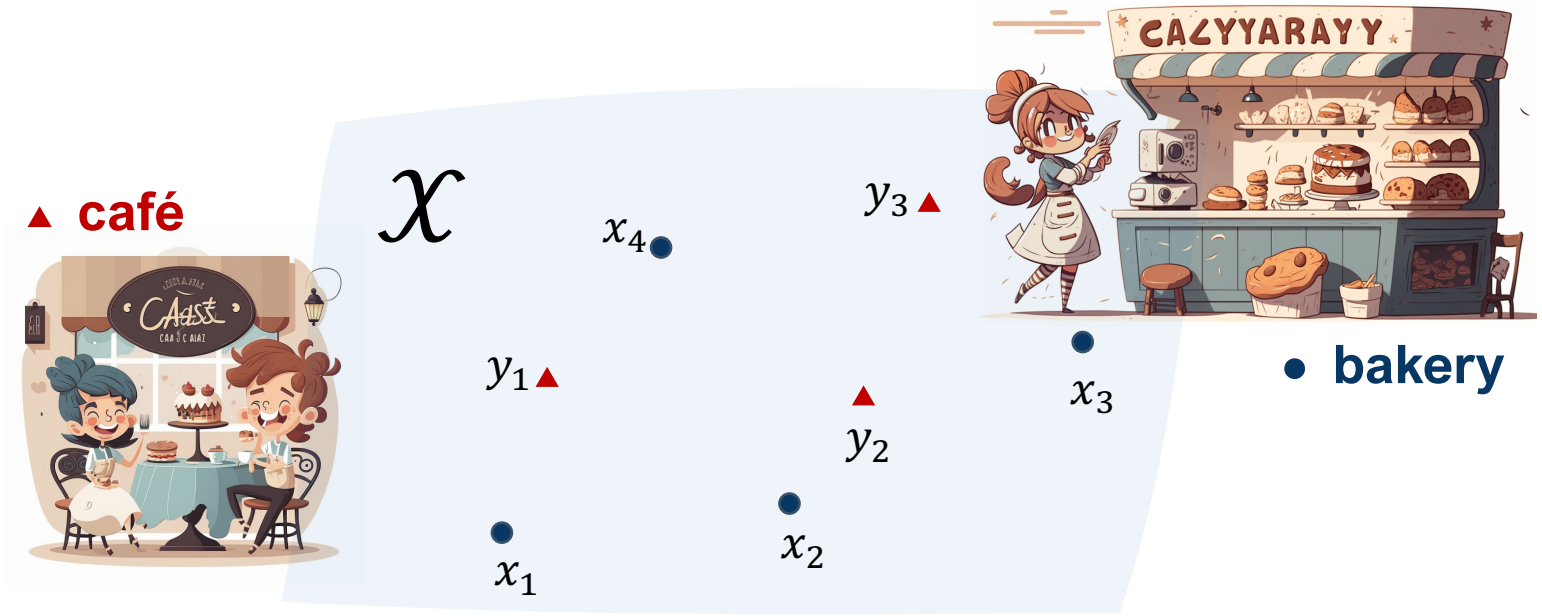
$$W(P, Q_1) > W(P, Q_2)$$

Dual formulation

- Let’s outsource the delivery

Dual formulation

- Let’s outsource the delivery



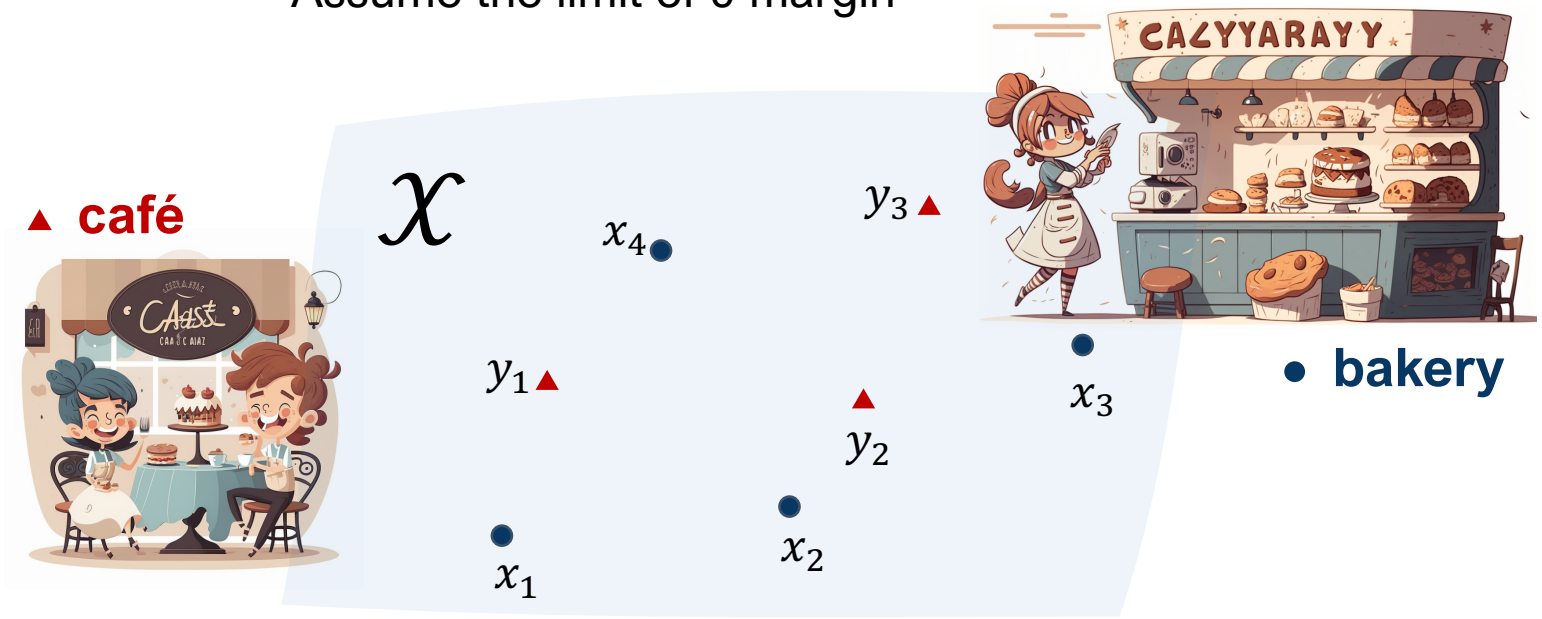
“Cozy patisserie cafe, happy customers, silly 2d cartoon, white background” – as interpreted by Midjourney

“Cozy bakery, silly 2d cartoon, white background” – as interpreted by Midjourney

Dual formulation

- Let's outsource the delivery
  - 3rd party company will buy the bread at bakeries for price $f(x)$ and sell to cafés for price $g(y)$
  - Assume the limit of 0 margin

"Shady trader in a van, silly 2d cartoon, white background" – as interpreted by Midjourney



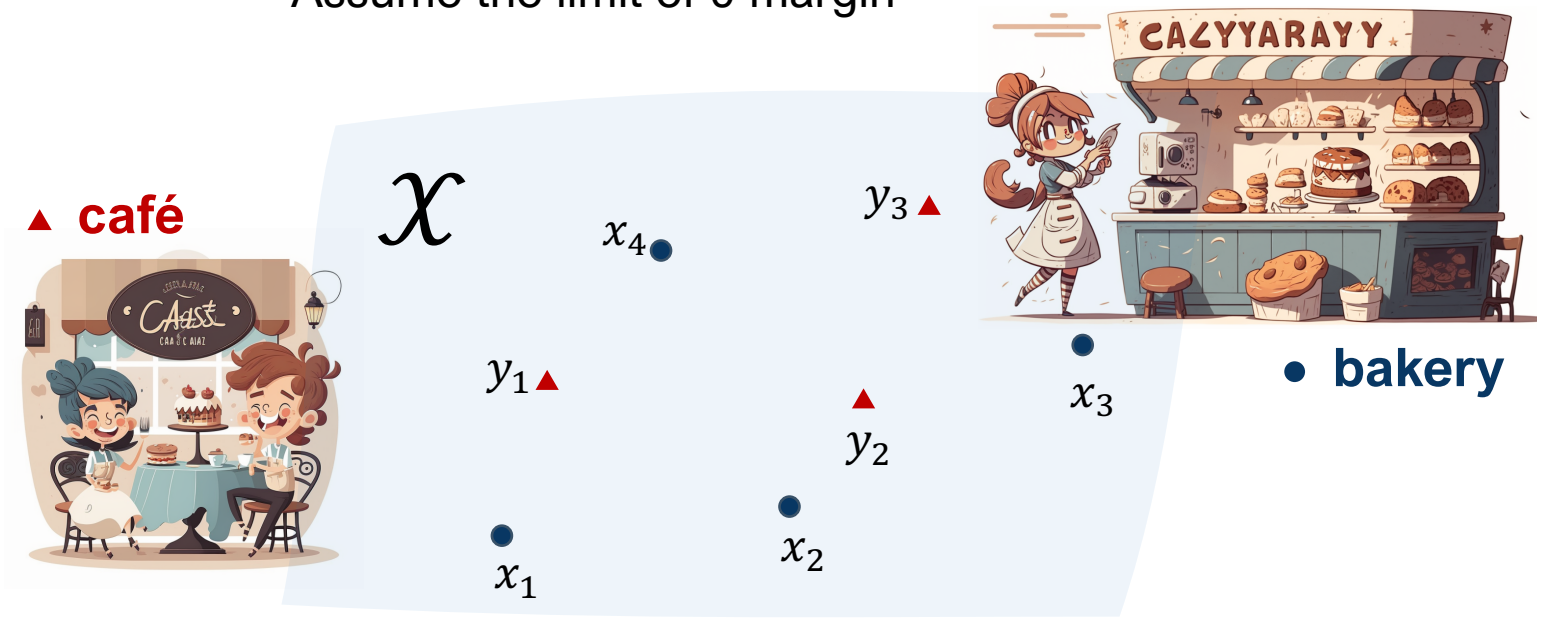
"Cozy patisserie cafe, happy customers, silly 2d cartoon, white background" – as interpreted by Midjourney

"Cozy bakery, silly 2d cartoon, white background" – as interpreted by Midjourney

Dual formulation

- Let's outsource the delivery
  - 3rd party company will buy the bread at bakeries for price $f(x)$ and sell to cafés for price $g(y)$
  - Assume the limit of 0 margin

"Shady trader in a van, silly 2d cartoon, white background" – as interpreted by Midjourney



- The company's revenue is:

$$\mathop{\mathbb{E}}_{y\sim q} g(y) - \mathop{\mathbb{E}}_{x\sim p} f(x)$$

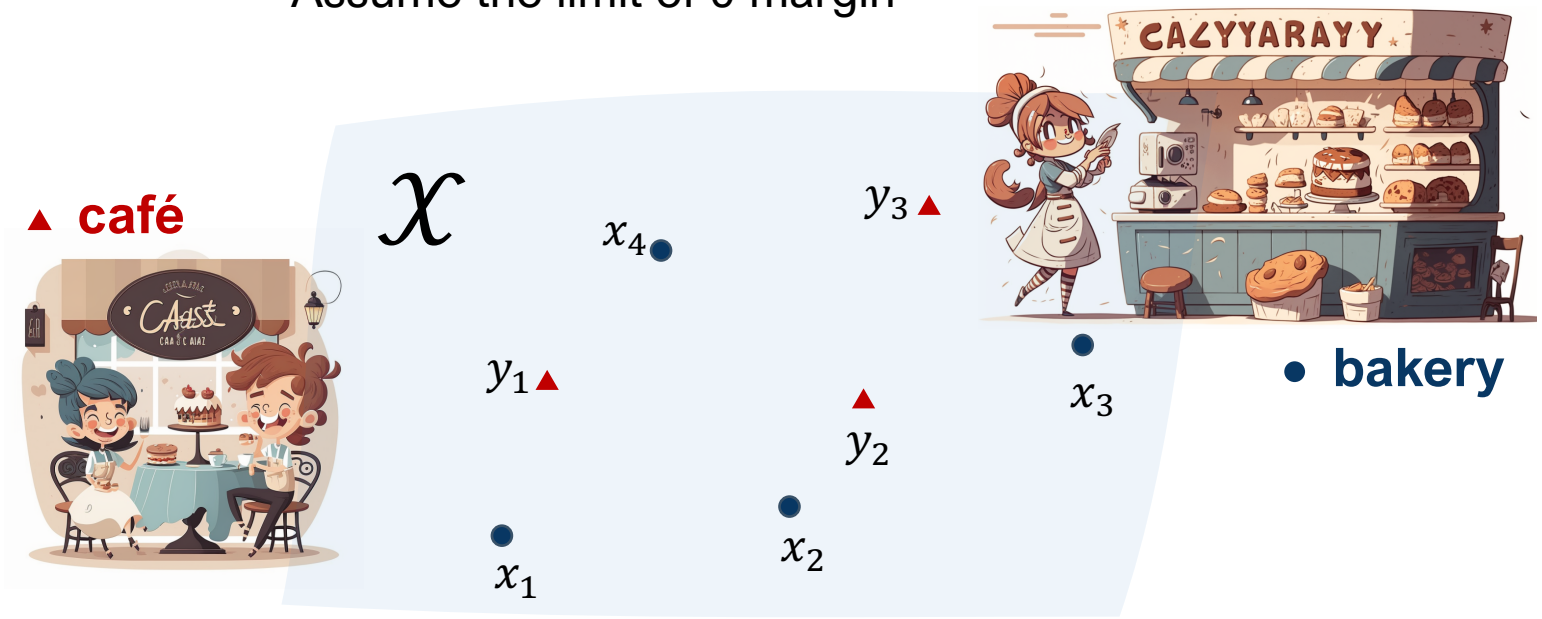
"Cozy patisserie cafe, happy customers, silly 2d cartoon, white background" – as interpreted by Midjourney

"Cozy bakery, silly 2d cartoon, white background" – as interpreted by Midjourney

Dual formulation

- Let's outsource the delivery
  - 3rd party company will buy the bread at bakeries for price $f(x)$ and sell to cafés for price $g(y)$
  - Assume the limit of 0 margin

"Shady trader in a van, silly 2d cartoon, white background" – as interpreted by Midjourney



"Cozy patisserie cafe, happy customers, silly 2d cartoon, white background" – as interpreted by Midjourney

"Cozy bakery, silly 2d cartoon, white background" – as interpreted by Midjourney

- The company's revenue is:

$$\mathop{\mathbb{E}}_{y\sim q} g(y) - \mathop{\mathbb{E}}_{x\sim p} f(x)$$

- To be competitive, the company's prices should satisfy:

$$g(y) - f(x) \le \|x - y\|$$

$$\Rightarrow g(x) \equiv f(x)$$

Dual formulation

- Functions satisfying $f(x) - f(y) \leq C\|x - y\|$ are called Lipschitz-continuous with Lipschitz constant $C = 1$
  - We'll denote them as $\|f\|_L \leq 1$

- The company seeks to maximize the revenue:

$$\mathop{\mathbb{E}}_{y \sim q} f(y) - \mathop{\mathbb{E}}_{x \sim p} f(x) \to \max_{\|f\|_L \leq 1}$$

https://en.wikipedia.org/wiki/Lipschitz_continuity

Dual formulation

- Let's look at the expectation from the original definition:

$$\mathop{\mathbb{E}}_{(x,y)\sim\gamma}\|x-y\| \geq \mathop{\mathbb{E}}_{(x,y)\sim\gamma}[f(y)-f(x)]$$

$$= \mathop{\mathbb{E}}_{(x,y)\sim\gamma} f(y) - \mathop{\mathbb{E}}_{(x,y)\sim\gamma} f(x)$$

$$= \mathop{\mathbb{E}}_{y\sim q} f(y) - \mathop{\mathbb{E}}_{x\sim p} f(x)$$

Dual formulation

- Let's look at the expectation from the original definition:

$$\min_{\gamma \in \Pi(p,q)} \mathop{\mathbb{E}}_{(x,y)\sim\gamma} \|x - y\| \geq \mathop{\mathbb{E}}_{(x,y)\sim\gamma} [f(y) - f(x)]$$

$$= \mathop{\mathbb{E}}_{(x,y)\sim\gamma} f(y) - \mathop{\mathbb{E}}_{(x,y)\sim\gamma} f(x)$$

$$= \mathop{\mathbb{E}}_{y\sim q} f(y) - \mathop{\mathbb{E}}_{x\sim p} f(x) \rightarrow \max_{\|f\|_L \leq 1}$$

Dual formulation

- Let's look at the expectation from the original definition:

$$\mathop{\mathbb{E}}_{(x,y)\sim\gamma}\|x-y\| \geq \mathop{\mathbb{E}}_{(x,y)\sim\gamma}[f(y)-f(x)]$$

$$= \mathop{\mathbb{E}}_{(x,y)\sim\gamma} f(y) - \mathop{\mathbb{E}}_{(x,y)\sim\gamma} f(x)$$

$$= \mathop{\mathbb{E}}_{y\sim q} f(y) - \mathop{\mathbb{E}}_{x\sim p} f(x) \rightarrow \max_{\|f\|_L \leq 1}$$

$\min_{\gamma\in\Pi(p,q)}$

$$W(p,q) = \inf_{\gamma\in\Pi(p,q)} \left[\mathbb{E}_\gamma\|x-y\|\right] = \sup_{\|f\|_L\leq 1} \left[\mathop{\mathbb{E}}_{y\sim q} f(y) - \mathop{\mathbb{E}}_{x\sim p} f(x)\right]$$

Algorithm (WGAN)

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

https://arxiv.org/abs/1701.07875

Algorithm (WGAN-GP)

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, \ldots, n_{\text{critic}}$ **do**
    for $i = 1, \ldots, m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

https://arxiv.org/abs/1704.00028

WGAN-GP

- Allowed for training of very deep GANs

Figure 2: Different GAN architectures trained with different methods. We only succeeded in training every architecture with a shared set of hyperparameters using WGAN-GP.