Deep Learning Theory and Applications

Generative Adversarial Networks (GANs)

Yale

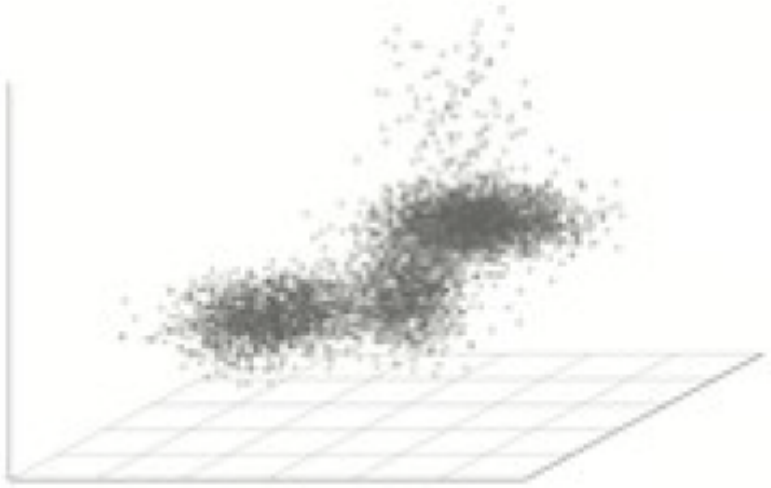
CPSC/AMTH 663

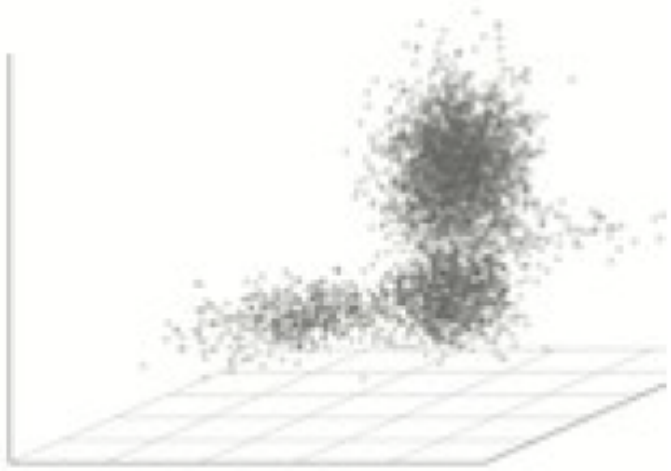
Outline

1. Probability Distributions
2. Distances and divergences between distributions
3. GANs
4. WGANs

Comparing two distributions

- Lets say you have to different datasets defining two different probability distributions
- How do you compare them?

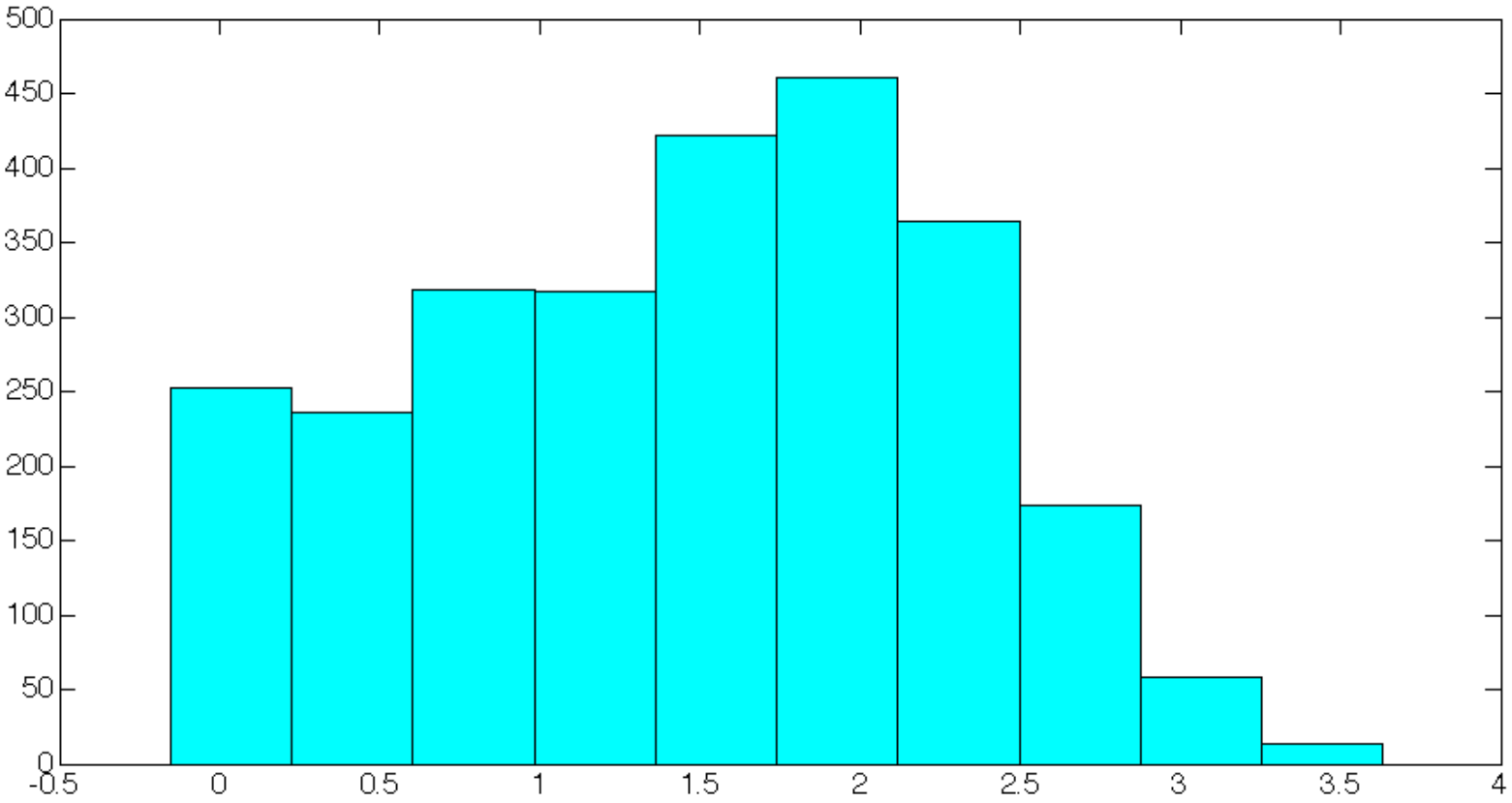




Comparing two clouds

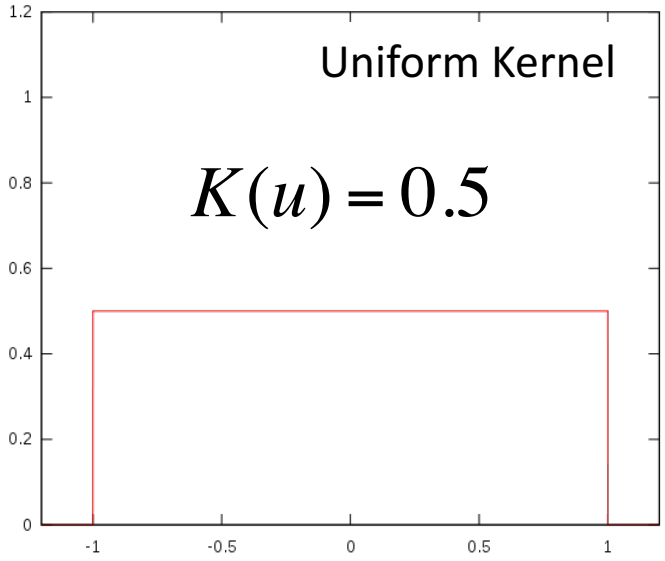
Estimating Probability Distributions

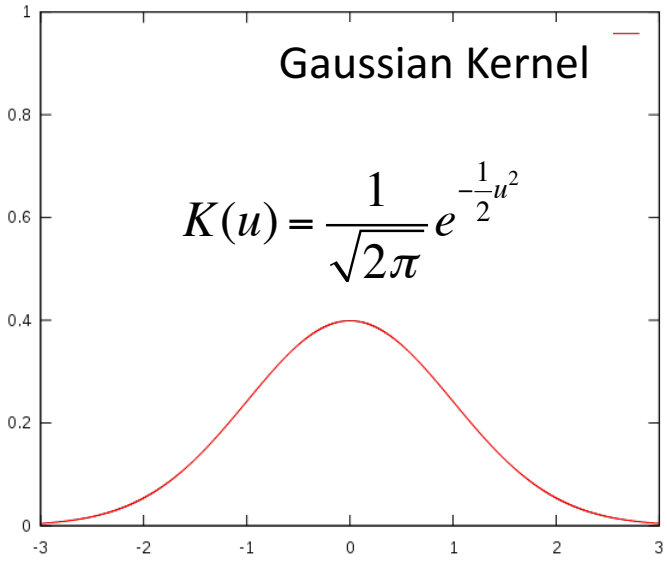
Recall you have points, and you can estimate a distribution



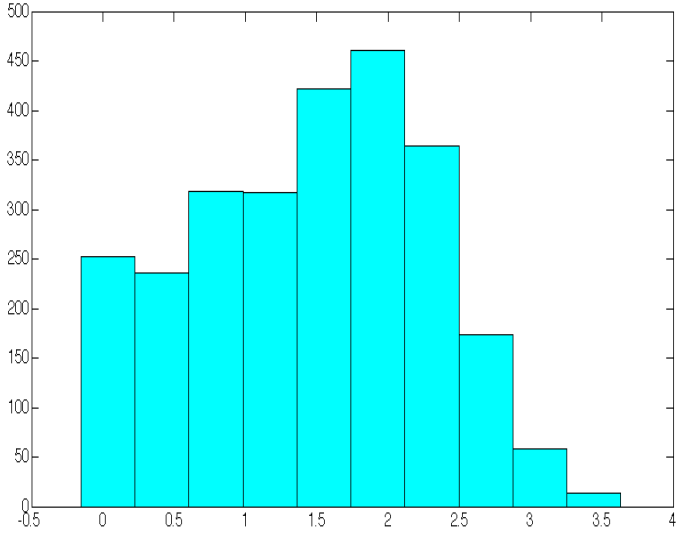
Kernel

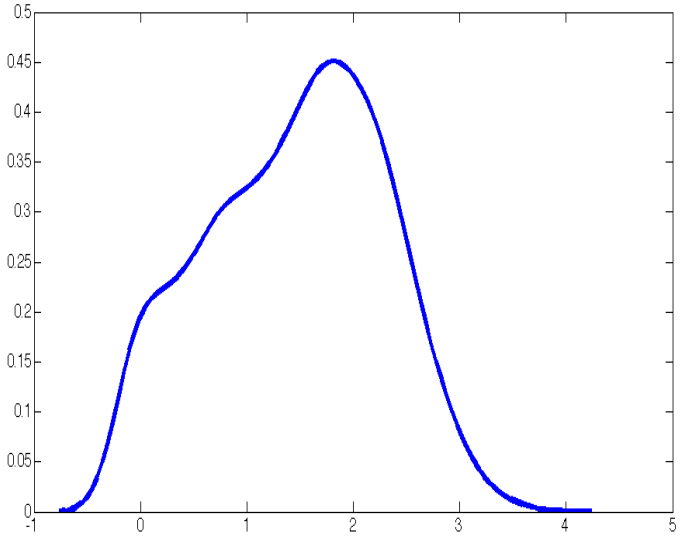
- A Kernel is a non-negative, real-valued, integrable function K that :
  - Integrates to 1
  - Symmetric K(-u)=K(u)





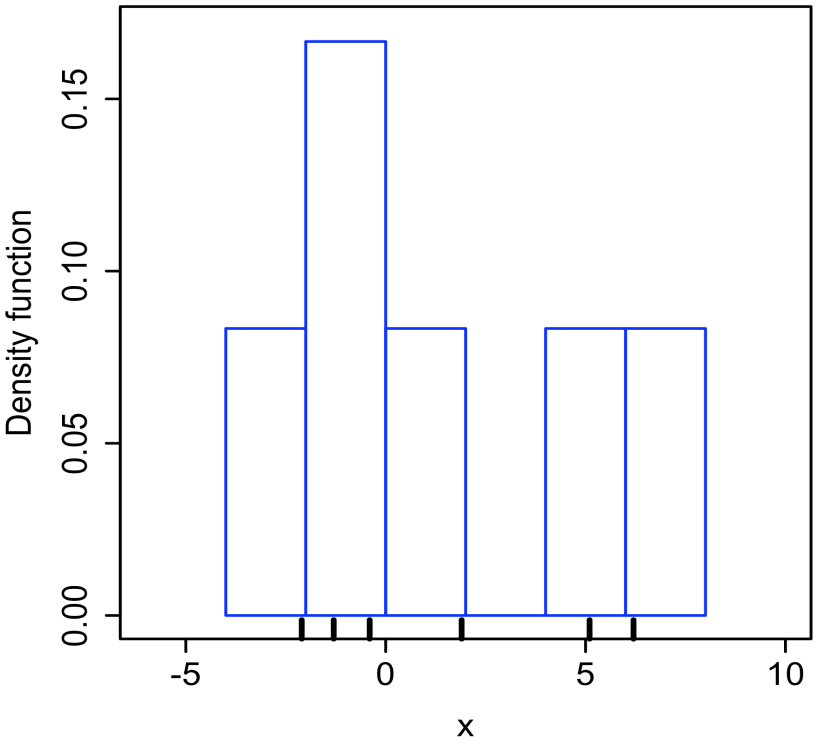
Density Estimate



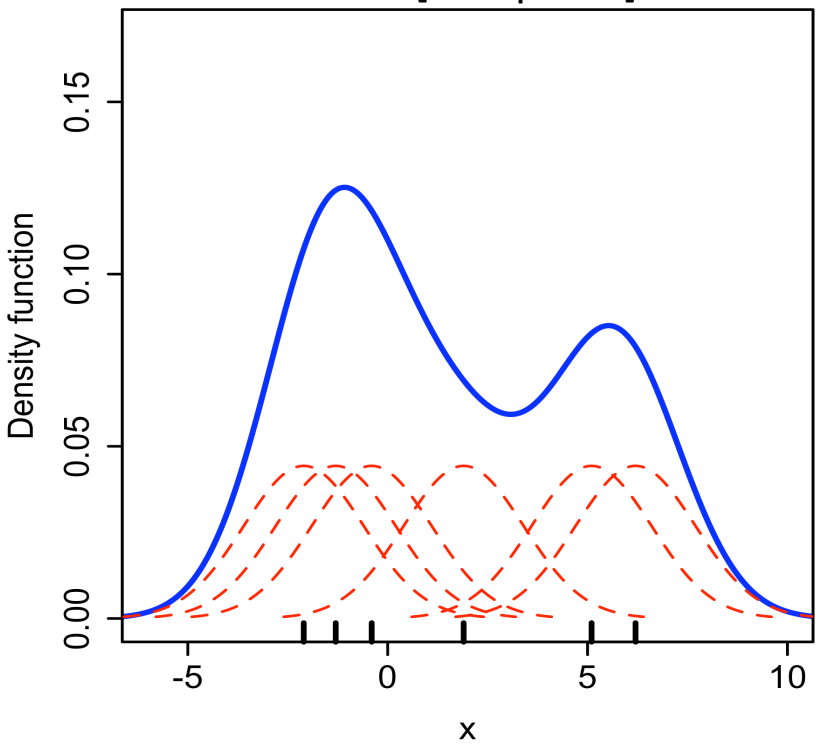


- Gives a smooth approximation to density
- Interpolates gaps

Kernel Density Estimate



[Wikipedia]



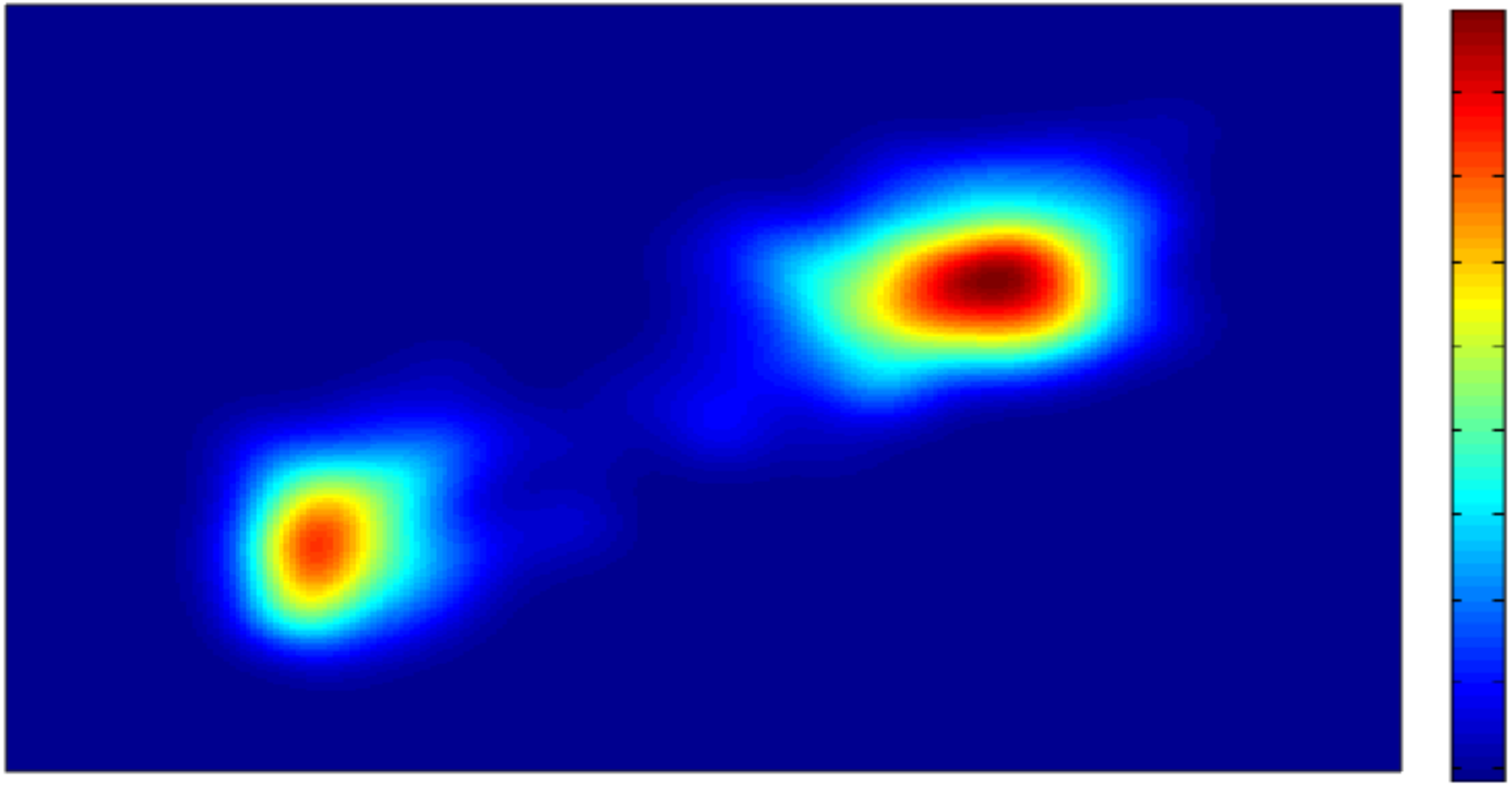
Scale Kernel By Bandwidth

$$K_h(x) = \frac{1}{h} K(\frac{x}{h})$$

Integrate over all Kernels

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i)$$

Kernel Density Estimation in 2D



KDE learns underlying probability distribution, smooths data

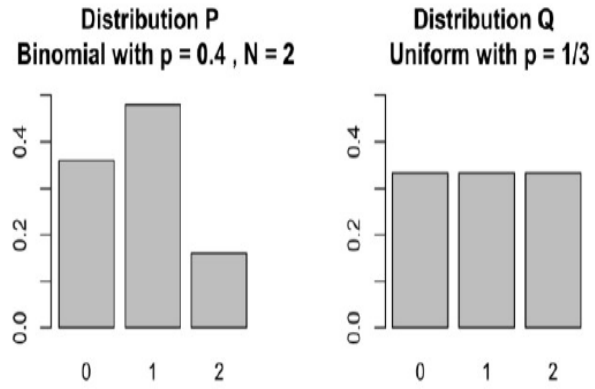
Ways of Comparing Probability Distributions

- Divergences
  - KL divergence
  - Jensen-Shannon Divergence

- Distances
  - Maximum Mean Discrepancy
  - Earth Mover's Distance (wasserstein distance)

KL-Divergence

$$D_{\mathrm{KL}}(P\|Q) = -\sum_i P(i)\,\log\left(\frac{Q(i)}{P(i)}\right)$$



	0	1	2
Distribution P	0.36	0.48	0.16
Distribution Q	0.333	0.333	0.333

$$\begin{aligned} D_{\mathrm{KL}}(Q\|P) &= \sum_i Q(i)\ln\left(\frac{Q(i)}{P(i)}\right) \\ &= 0.333\ln\left(\frac{0.333}{0.36}\right) + 0.333\ln\left(\frac{0.333}{0.48}\right) + 0.333\ln\left(\frac{0.333}{0.16}\right) \\ &= -0.02596 + (-0.12176) + 0.24408 \\ &= 0.09637 \end{aligned}$$

Interpretation

- Information gained by using Q instead of P
- Number of bits needed to encode P using the code of Q

$$D_{\mathrm{KL}}(P\|Q) = -\sum_{x} p(x)\log q(x) + \sum_{x} p(x)\log p(x)$$
$$= \mathrm{H}(P,Q) - \mathrm{H}(P)$$

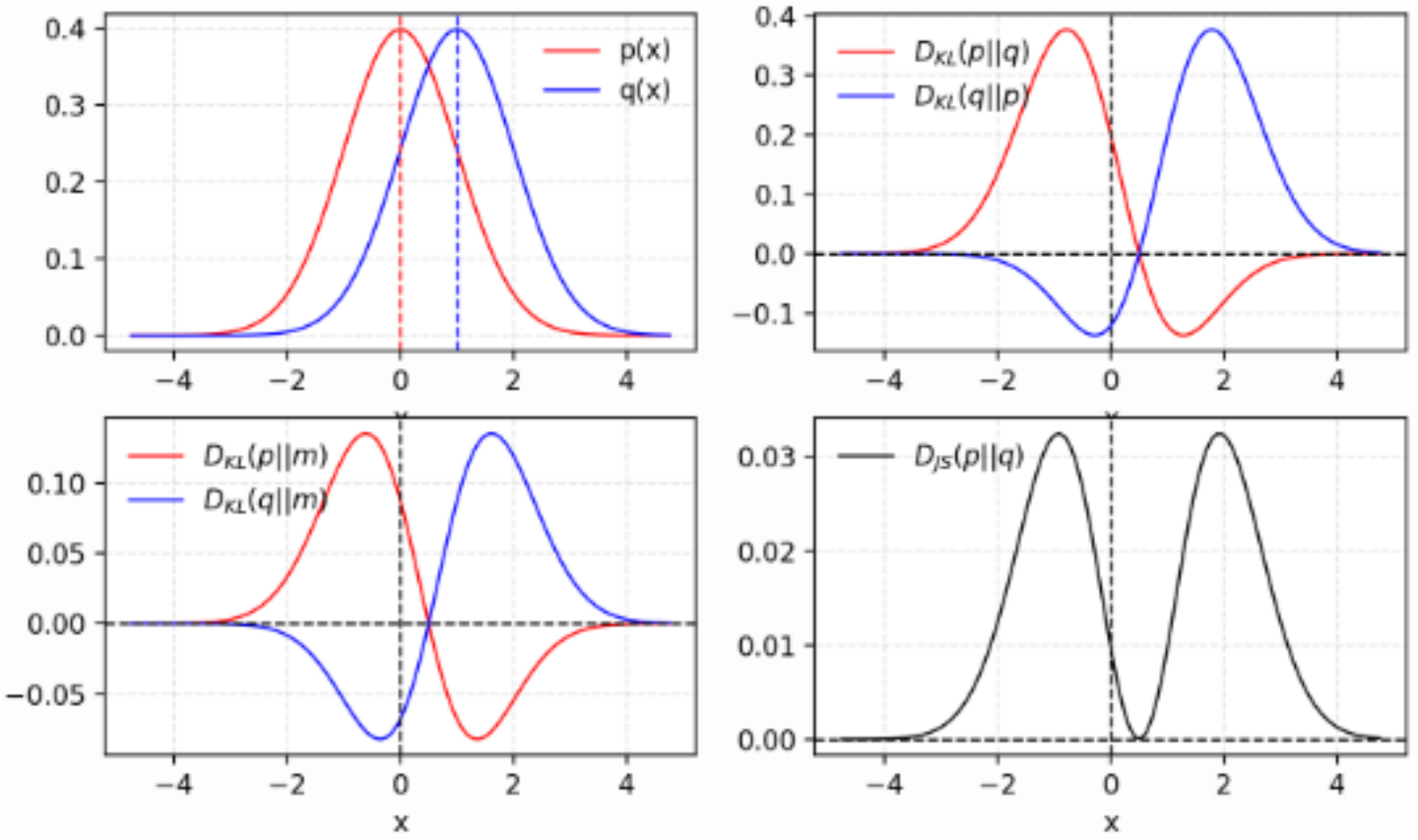
- H(P,Q) here is called the cross entropy

Continuous formula

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x)\log\left(\frac{p(x)}{q(x)}\right) dx \quad \textit{(Eq.2)}$$

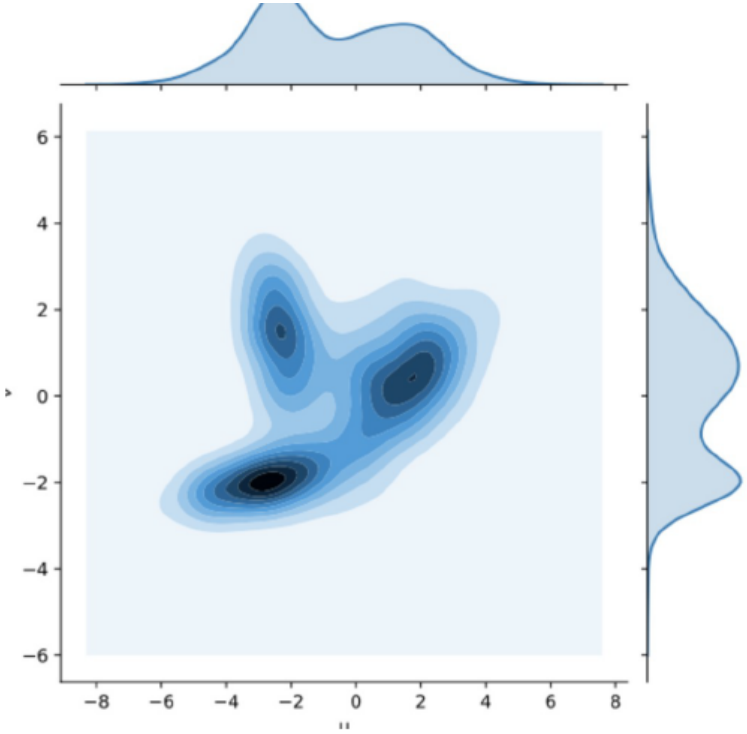
Jensen-Shannon Divergence

$$D_{JS}(p\|q) = \frac{1}{2}D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q\|\frac{p+q}{2})$$

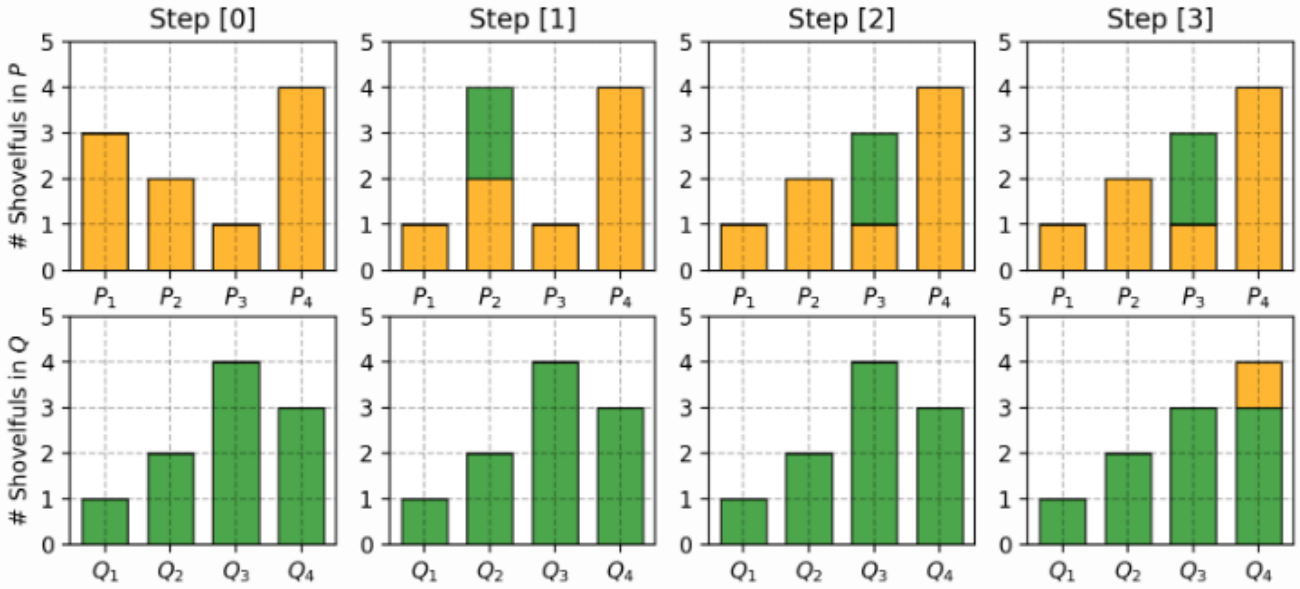


Earth Mover's Distance

- Each distribution is dirt piled on M
- The metric is the minimum cost of moving the dirt from one pile to another
- Roughly speaking its the amount of dirt times distance



Optimal Transport



$$\delta_0 = 0$$
$$\delta_1 = 0 + 3 - 1 = 2$$
$$\delta_2 = 2 + 2 - 2 = 2$$
$$\delta_3 = 2 + 1 - 4 = -1$$
$$\delta_4 = -1 + 4 - 3 = 0$$

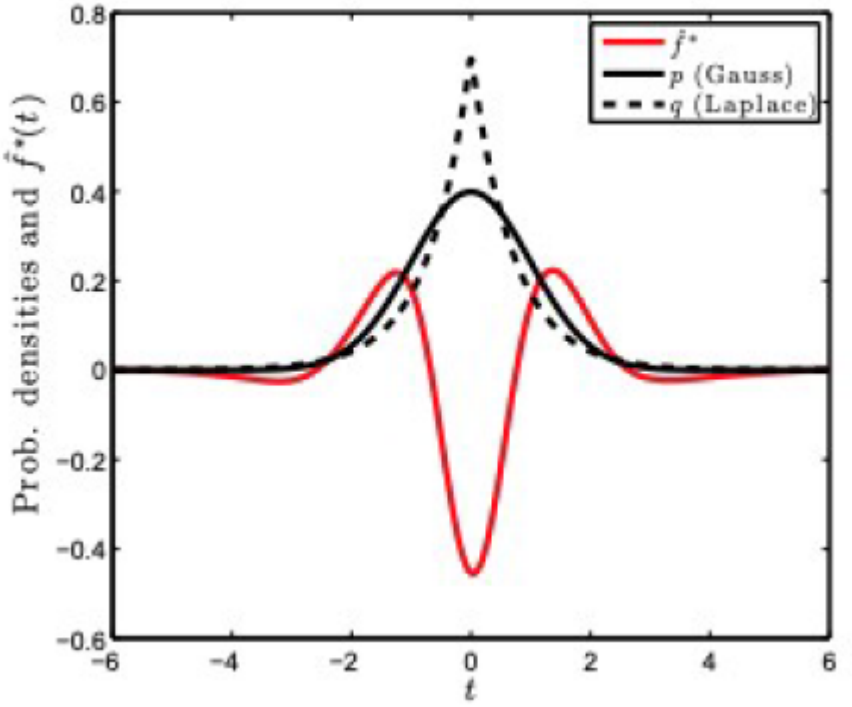
Finally the Earth Mover's distance is $W = \sum |\delta_i| = 5$.

Rubenstein-Kantorovich Inequality

Kantorovich-Rubenstein duality:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$$

Witness function f



Continuous Formula for EMD

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

$\Pi(p_r, p_g)$ is the set of all possible joint probability distributions

$\gamma \in \Pi(p_r, p_g)$ describes one dirt transport plan,

Solved using an optimizer such as the Hungarian algorithm or a newer wavelet based algorithm

GANs: Motivation

- We have a sample **x** from some distribution P(**X**)
- How do we sample from P(**X**)?

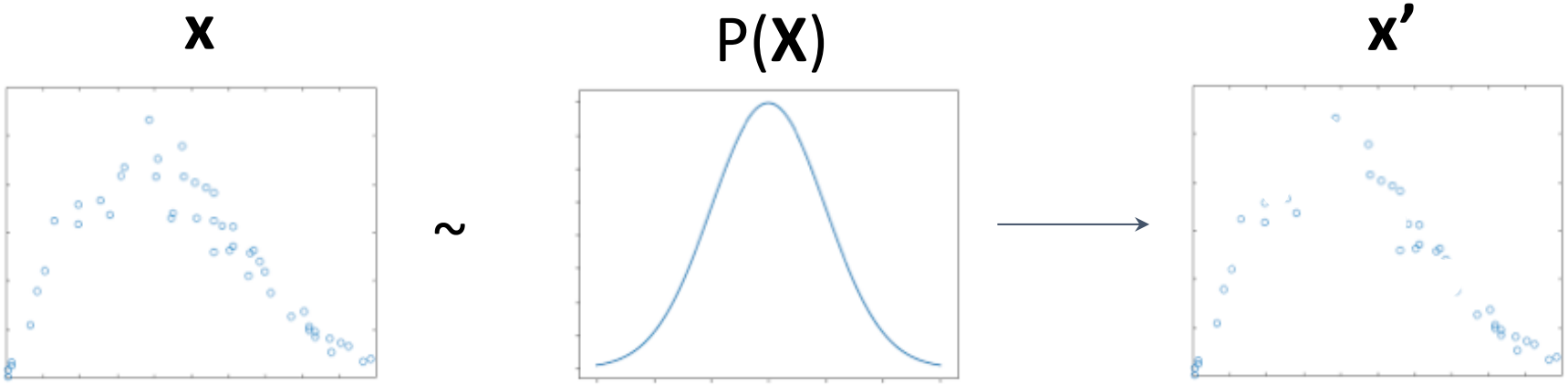
x

~

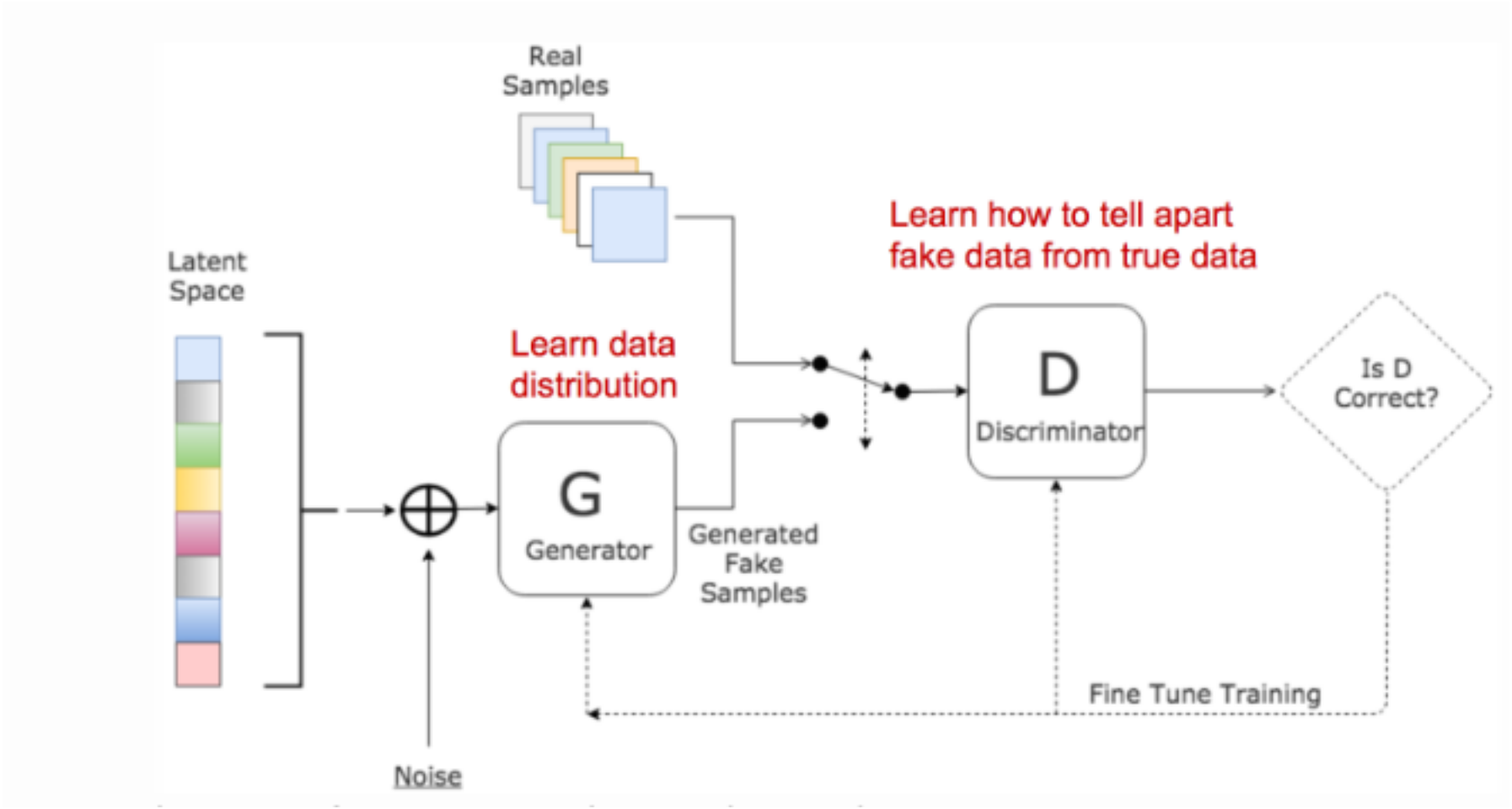
P(**X**)

GANs: Motivation

- Would be easier in low dimensions and a parametric model
- We're interested in very high dimensions
  - **Do not** want to limit ourselves with a parametric model
  - High dimensional real datasets rarely satisfy parametric assumptions



Generative Adversarial Network



Goodfellow et al. 2014

GAN Loss function

p_z	Data distribution over noise input z
p_g	The generator's distribution over data x
p_r	Data distribution over real sample x

Maximize Discriminator's answer. (1) on real samples

$$\mathbb{E}_{x \sim p_r(x)}[\log D(x)]$$

Maximize Discriminator's answer (0) for fake samples

$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))].$$

Minimax game: Generator wants discriminator to perform badly

$$\begin{aligned}\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]\end{aligned}$$

Optimal Discriminator Behavior

- Proof from paper:

$$L(G,D) = \int_x \left(p_r(x)\log(D(x)) + p_g(x)\log(1-D(x)) \right) dx$$

$$\tilde{x} = D(x), A = p_r(x), B = p_g(x)$$

$$f(\tilde{x}) = A log\tilde{x} + Blog(1-\tilde{x})$$

$$\begin{aligned}\frac{df(\tilde{x})}{d\tilde{x}} &= A\frac{1}{ln10}\frac{1}{\tilde{x}} - B\frac{1}{ln10}\frac{1}{1-\tilde{x}} \\ &= \frac{1}{ln10}\left(\frac{A}{\tilde{x}} - \frac{B}{1-\tilde{x}}\right) \\ &= \frac{1}{ln10}\frac{A-(A+B)\tilde{x}}{\tilde{x}(1-\tilde{x})}\end{aligned}$$

Thus, set $\frac{df(\tilde{x})}{d\tilde{x}} = 0$, we get the best value of the discriminator:

$$D^*(x) = \tilde{x}^* = \frac{A}{A+B} = \frac{p_r(x)}{p_r(x)+p_g(x)} \in [0,1].$$

Discriminator computes JS Divergence

- When generator is optimal $p_g = p_r, D^*(x)$
- When both G and D are optimal D loss is -2 log 2 (otherwise known as ½)
- According to formula for JS Divergence

$$
\begin{aligned}
D_{JS}(p_r \| p_g) &= \frac{1}{2} D_{KL}(p_r || \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g || \frac{p_r + p_g}{2}) \\
&= \frac{1}{2}\left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r + p_g(x)} dx \right) + \\
&\quad \frac{1}{2}\left(\log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r + p_g(x)} dx \right) \\
&= \frac{1}{2}\left(\log 4 + L(G, D^*) \right)
\end{aligned}
$$

- Thus the discriminator converges to a JS divergence between the samples

Potential Improvement of GAN over VAE

- A maximum likelihood model implicitly minimizes KL divergence between generated samples and real data

$$D_{KL}[P(x|\theta^*) \,\|\, P(x|\theta)] = \mathbb{E}_{x\sim P(x|\theta^*)}\left[\log \frac{P(x|\theta^*)}{P(x|\theta)}\right]$$

$$= \mathbb{E}_{x\sim P(x|\theta^*)}\left[\log P(x|\theta^*) - \log P(x|\theta)\right]$$

$$= \mathbb{E}_{x\sim P(x|\theta^*)}\left[\log P(x|\theta^*)\right] - \mathbb{E}_{x\sim P(x|\theta^*)}\left[\log P(x|\theta)\right]$$

As N goes to infinity

$$-\frac{1}{N}\sum_{i}^{N} \log P(x_i|\theta) = -\mathbb{E}_{x\sim P(x|\theta^*)}\left[\log P(x|\theta)\right]$$

Thus the right term becomes constant and so the left term maximizes log likelihood

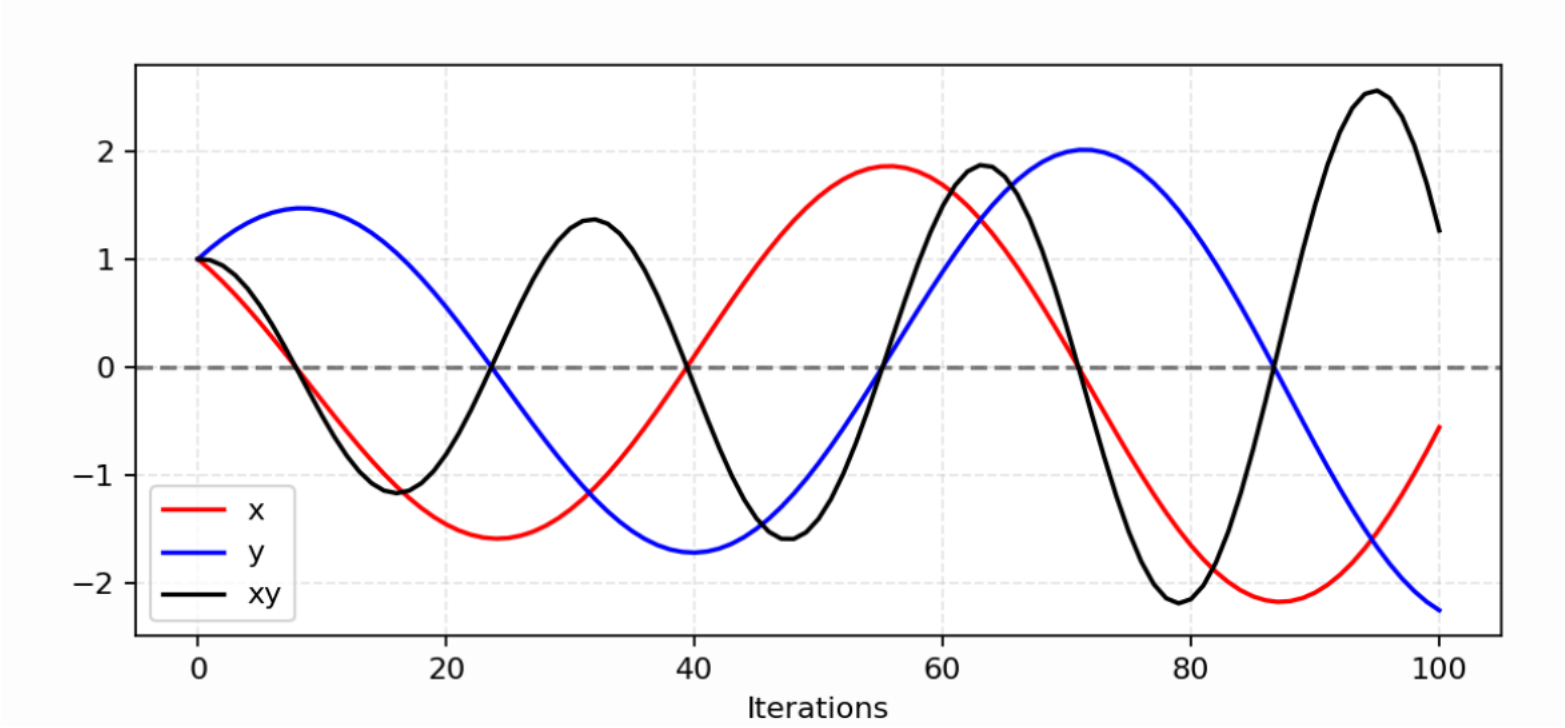
VAEs and many other parameter estimation algorithms do maximum likelihood

GANs

- GANs can generate sharper images generally
  - Do not need to explicitly define distribution distances
- Unfortunately, learning GANs with neural networks can be difficult in practice
  - Convergence to equilibrium is not guaranteed
  - Equilibria for minimax game are saddle points

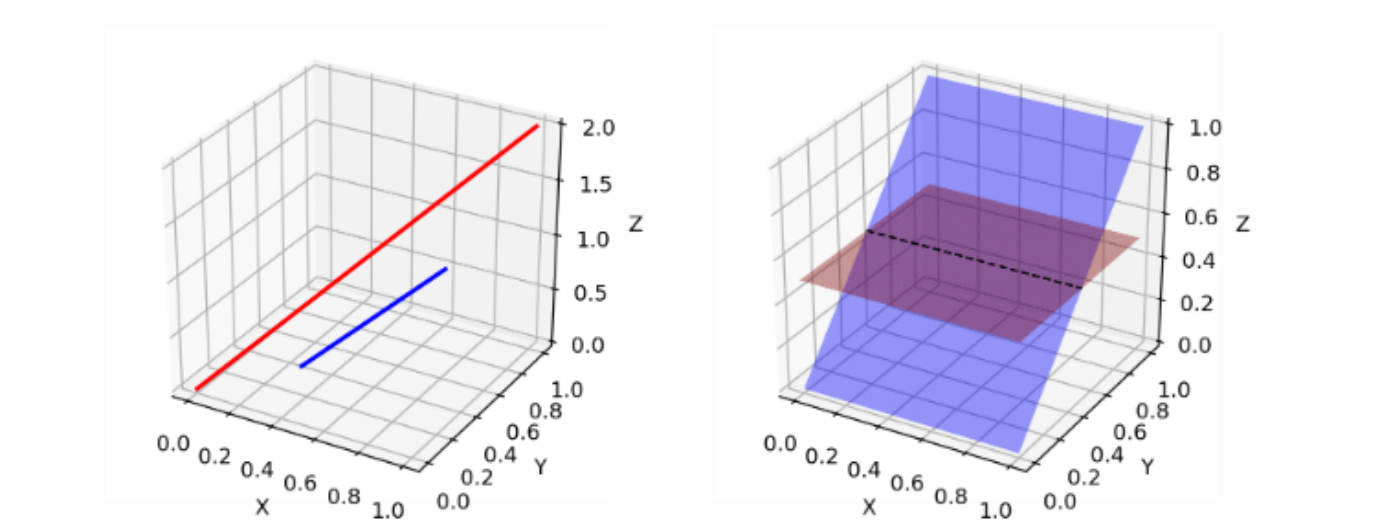
Instability with 2-player game

- Example: $v(a, b) = ab$
  - One player controls a and incurs cost ab; other player controls b and incurs cost $-ab$
  - Making small gradient steps can result in a stable circular orbit instead of arriving at the origin

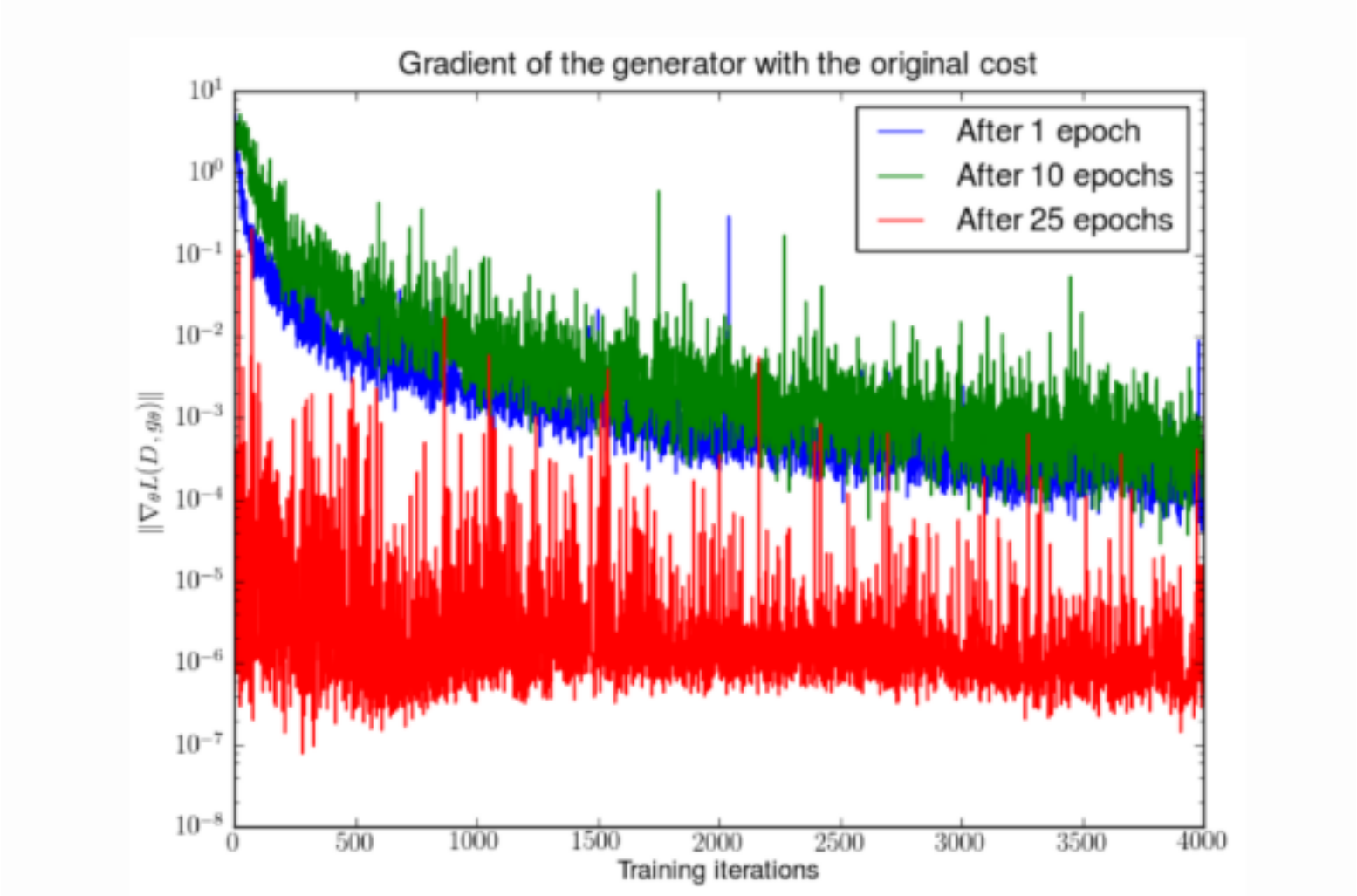


Problems with GANs 1: Small gradients when generator is bad

- When generated data is very far from training data the two distributions are far away discriminator can be very good
- Small gradients, generator does not know how to change itself to learn well

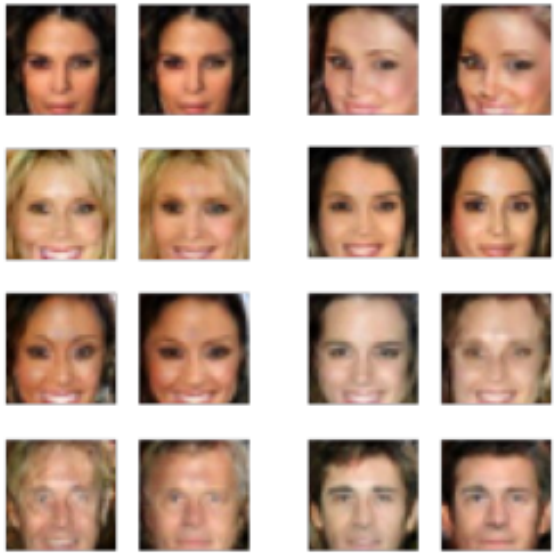


Problem 2: Vanishing gradient for generator when discriminator gets good

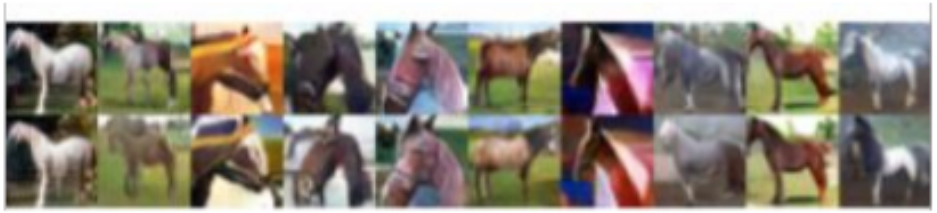


Problem 3: Mode Collapse

The generator can fool the discriminator by memorizing a small number of images.



evidence of lack of diversity



Improvements in Training GANs

- Feature mathcing
  - Optimize the discriminator to inspect whether the generator's output matches the statistics of real samples, i.e., does the generated mean equal the real mean?

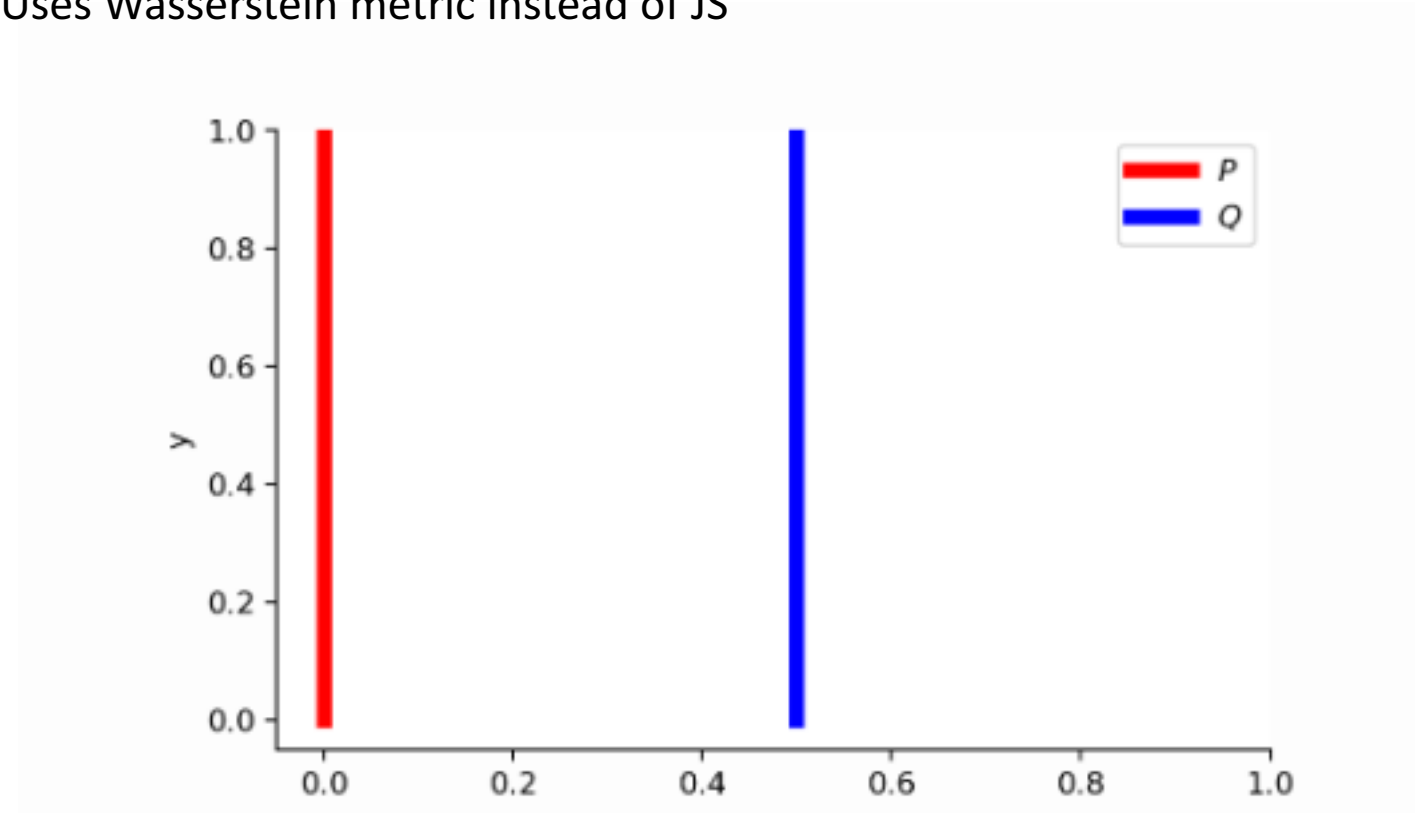
$$\|\mathbb{E}_{x\sim p_r} f(x) - \mathbb{E}_{z\sim p_z(z)} f(G(z))\|_2^2$$

- Minibatch discrimination
  - Take the batch as a whole instead of one input at a time and learn relationships between pairs of samples $c(x_i, x_j)$,

$$o(x_i) = \sum_j c(x_i, x_j).$$

Wasserstein GAN [Arjovsky et al. 2017]

Uses Wasserstein metric instead of JS



Distance between manifolds with disjoint support

Distances on parallel lines

KL Divergence:

$$D_{KL}(P\|Q) = \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q\|P) = \sum_{x=\theta, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

JS Divergence:

$$D_{JS}(P, Q) = \frac{1}{2}\left(\sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2} \right) = \log 2$$

Same value independent of distance between lines!

Wasserstein on Parallel Lines

$$W(P, Q) = |\theta|$$

Matches with intuition of moving dirt.

Moral of the story: Wasserstein distance can be better when the supports of the two probability distributions are far away.

This can occur when generated data is far from training data

Wasserstein Distance by Discrimnator

- Instead of the Discriminator giving a decision “0” for fake data but “1” for real data gives a value of a function f
- The witness function f can be estimated by the Discriminator
- The Discriminator would be outputing a low value for fake samples and high value for real data

Kantorovich-Rubenstein duality:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$$

- This function has to be lipschitz continous (can’t be jagged) has to vary smoothly

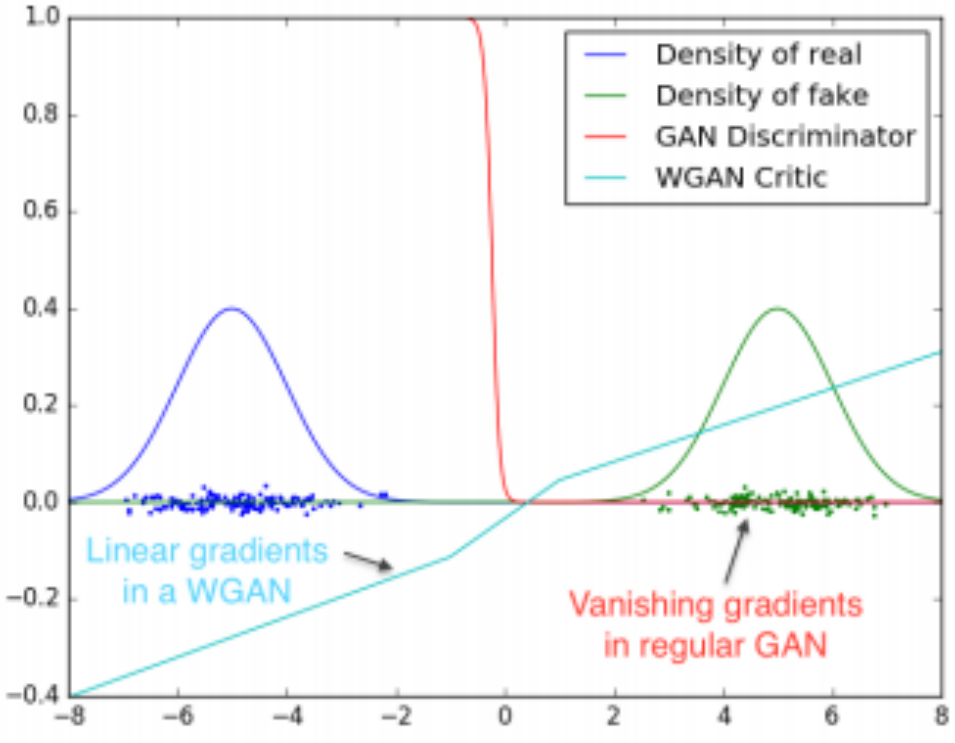
$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$$

WGAN In practice

- Matt: “WGANs just don’t have a sigmoid output layer” :)
- After every gradient update to the witness function f, clamp weights to a small fixed range
- New loss function does not have logarithm, just difference of expected values

$$L(p_r, p_g) = W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$$

Improves vanishing Gradient



GANs

- Can learn well with carefully selected model architectures
  - Radford et al. (2015): deep convolutional GAN (DCGAN)
  - Generated images of bedrooms
- Can break the generation process into many levels
  - Can learn to sample from conditional distributions
  - Train a series of conditional GANs to first generate low-res versions and then incrementally add details
  - Denton et al. (2015): LAPGAN
  - Generated images of churches

Further reading

- Goodfellow et al., Section 20.10.4
- Lectures on GANs CS 11-785 at CMU
- https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html
- https://wiseodd.github.io/techblog/2017/01/26/kl-mle/
- Goodfellow et al 2014 paper
- Arjovsky et al 2017 paper