Generative Adversarial Networks

Egor Zakharov

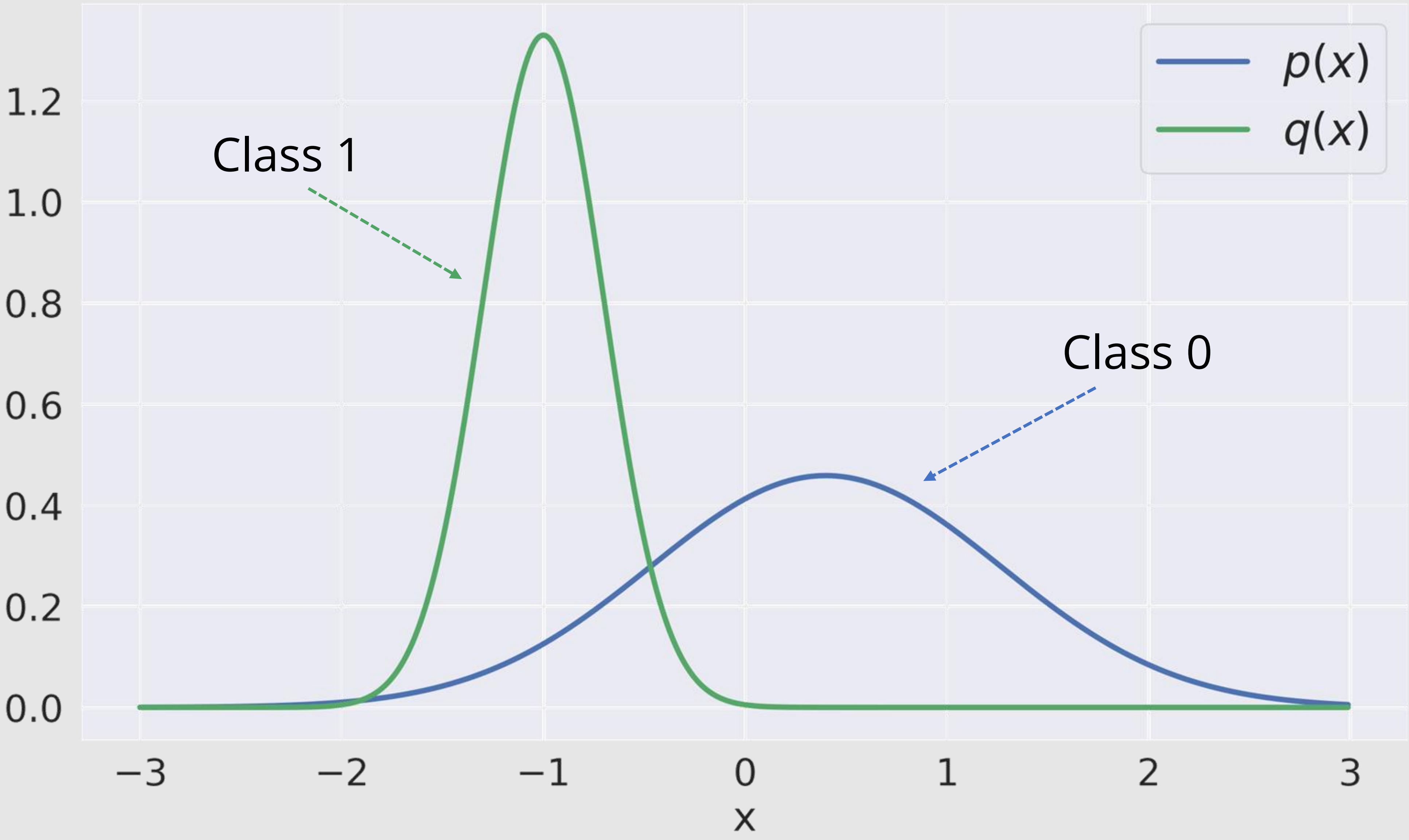
Engineer at Samsung AI Center

PhD student at Skolkovo Institute of Science and Technology

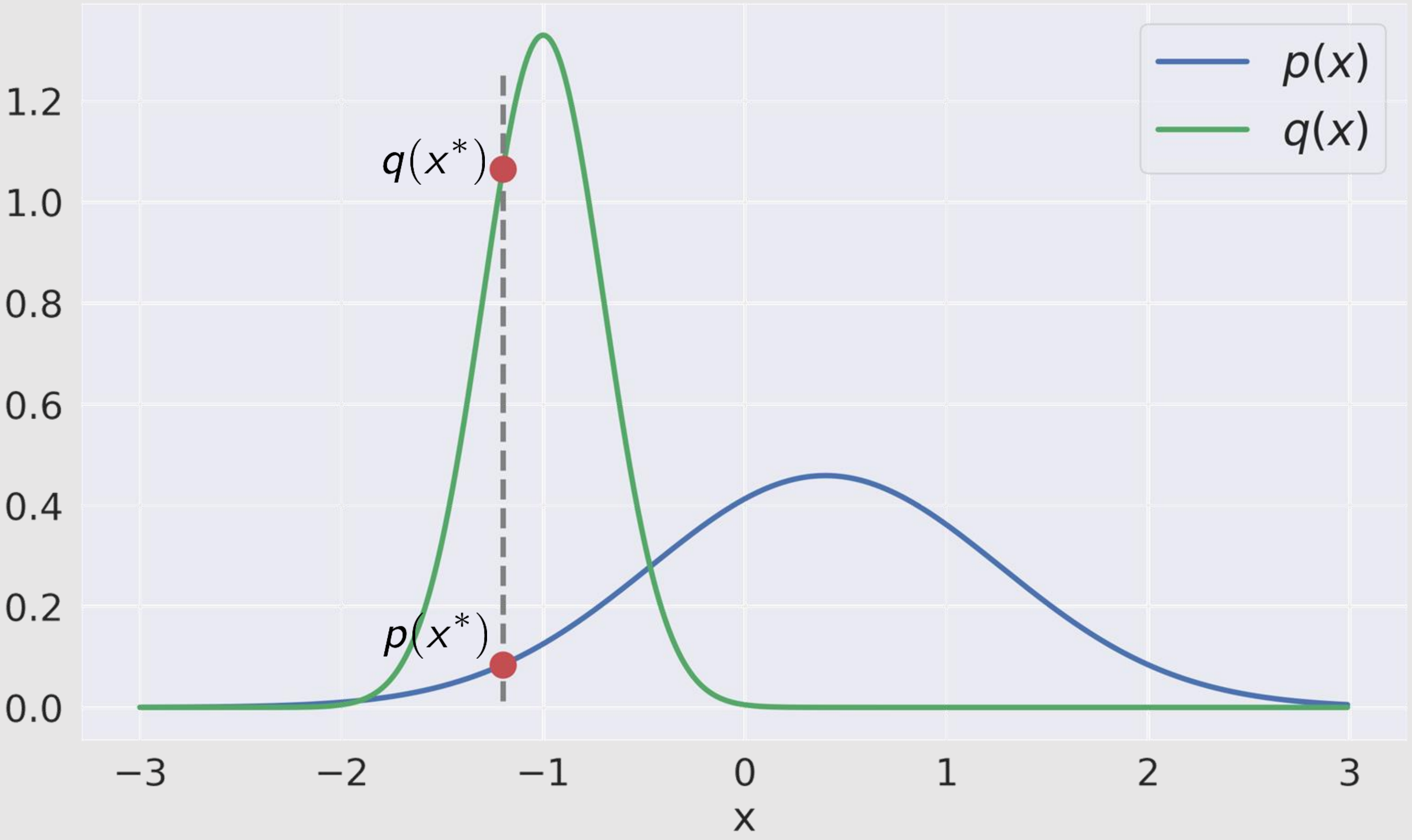
Outline

- Part I
  - Intuition behind GANs
  - Implicit vs explicit generative models
  - JS and f-divergences
  - Wasserstein GAN
- Part II
  - Spectral normalization
  - Moving averaging, weight averaging
  - Quality measurements
  - Applications

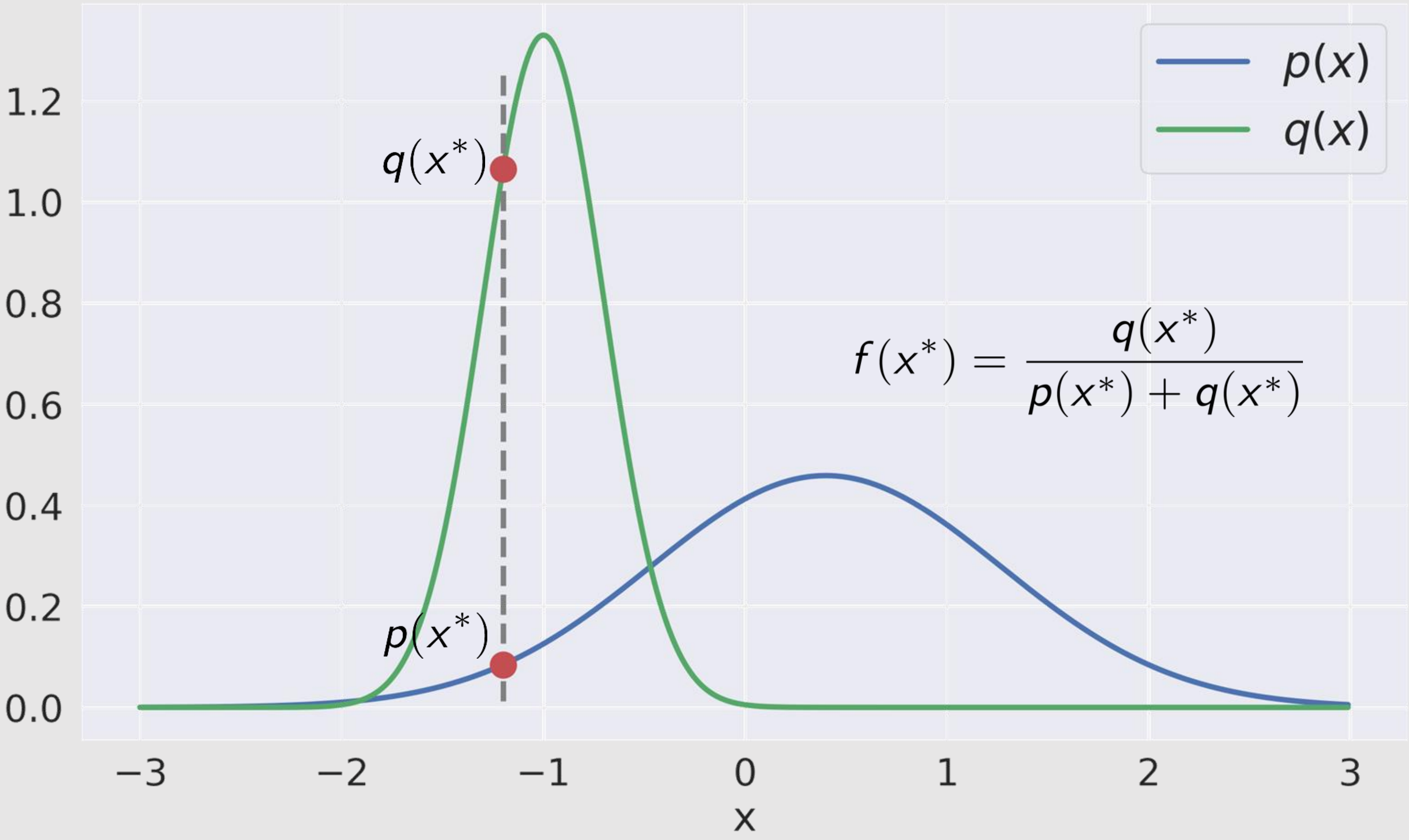
Classification



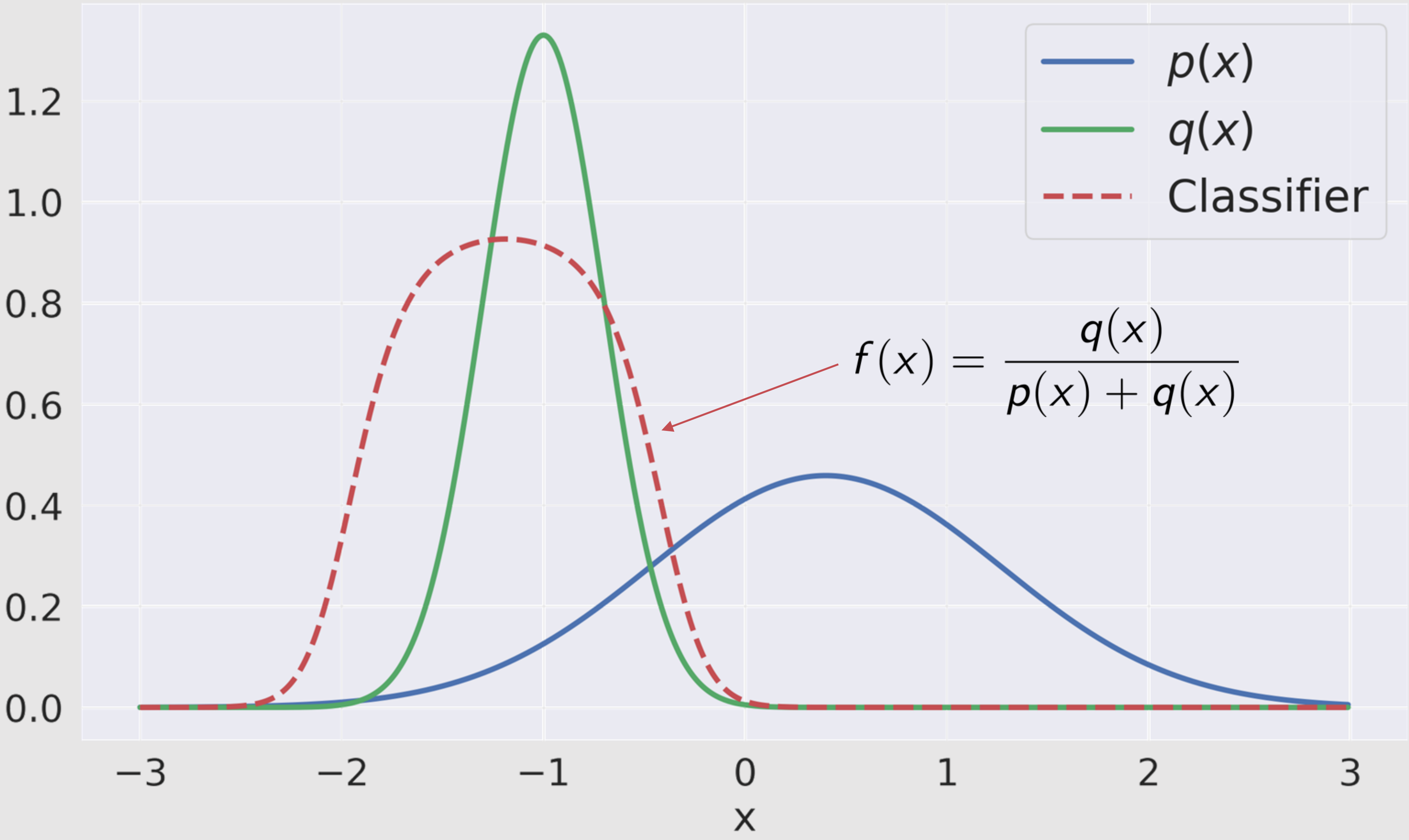
Classification



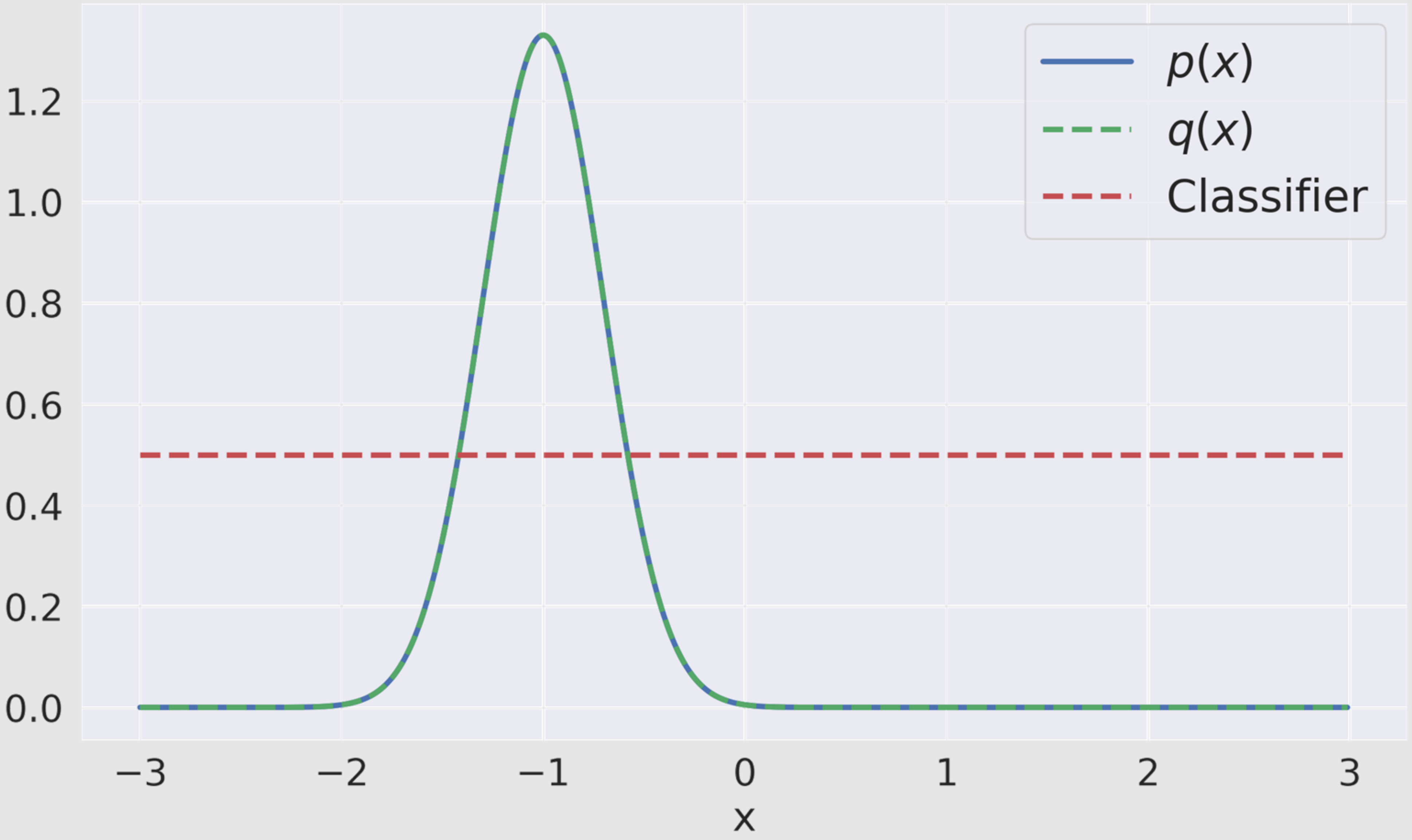
Classification



Classification



Classification

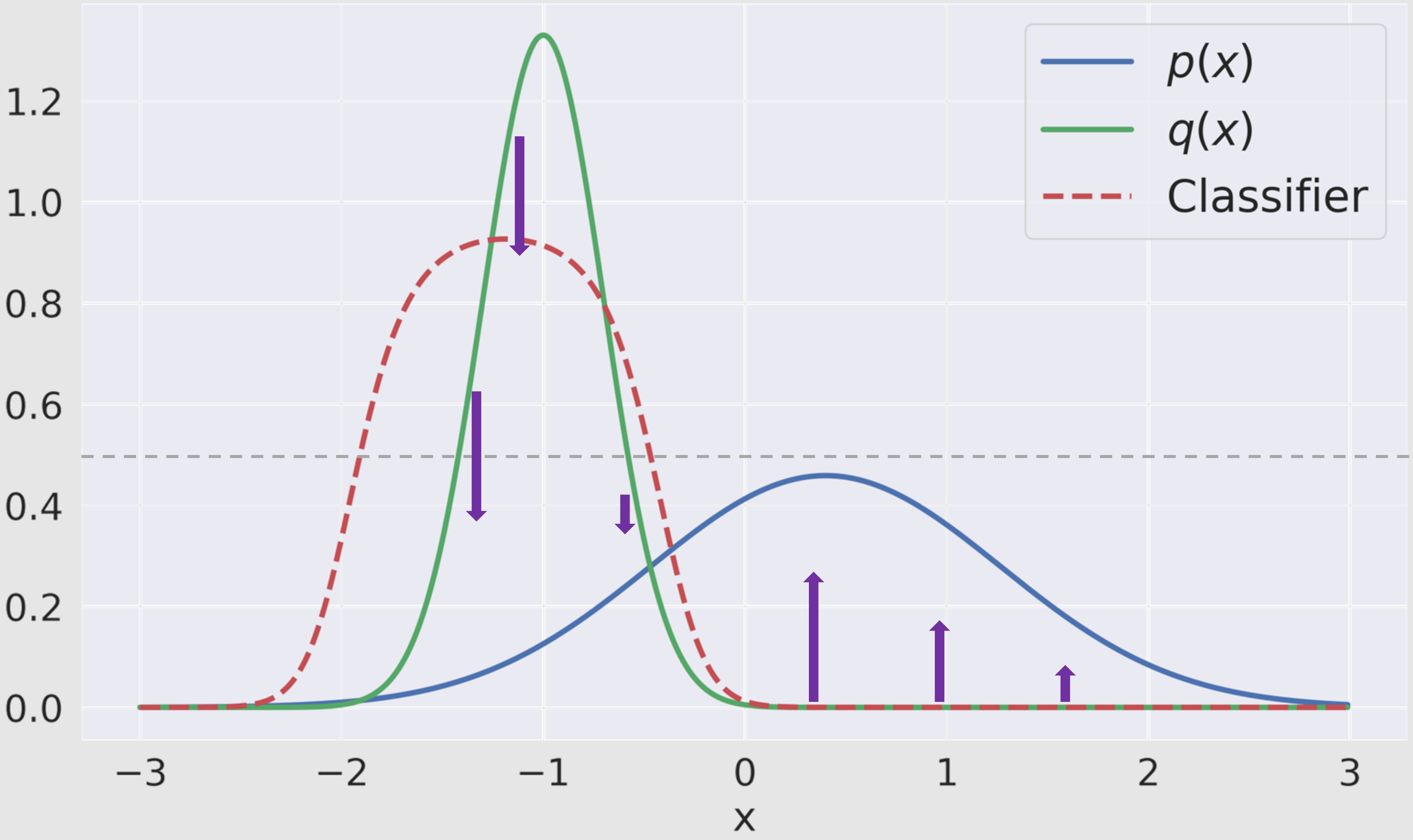


Some intuition

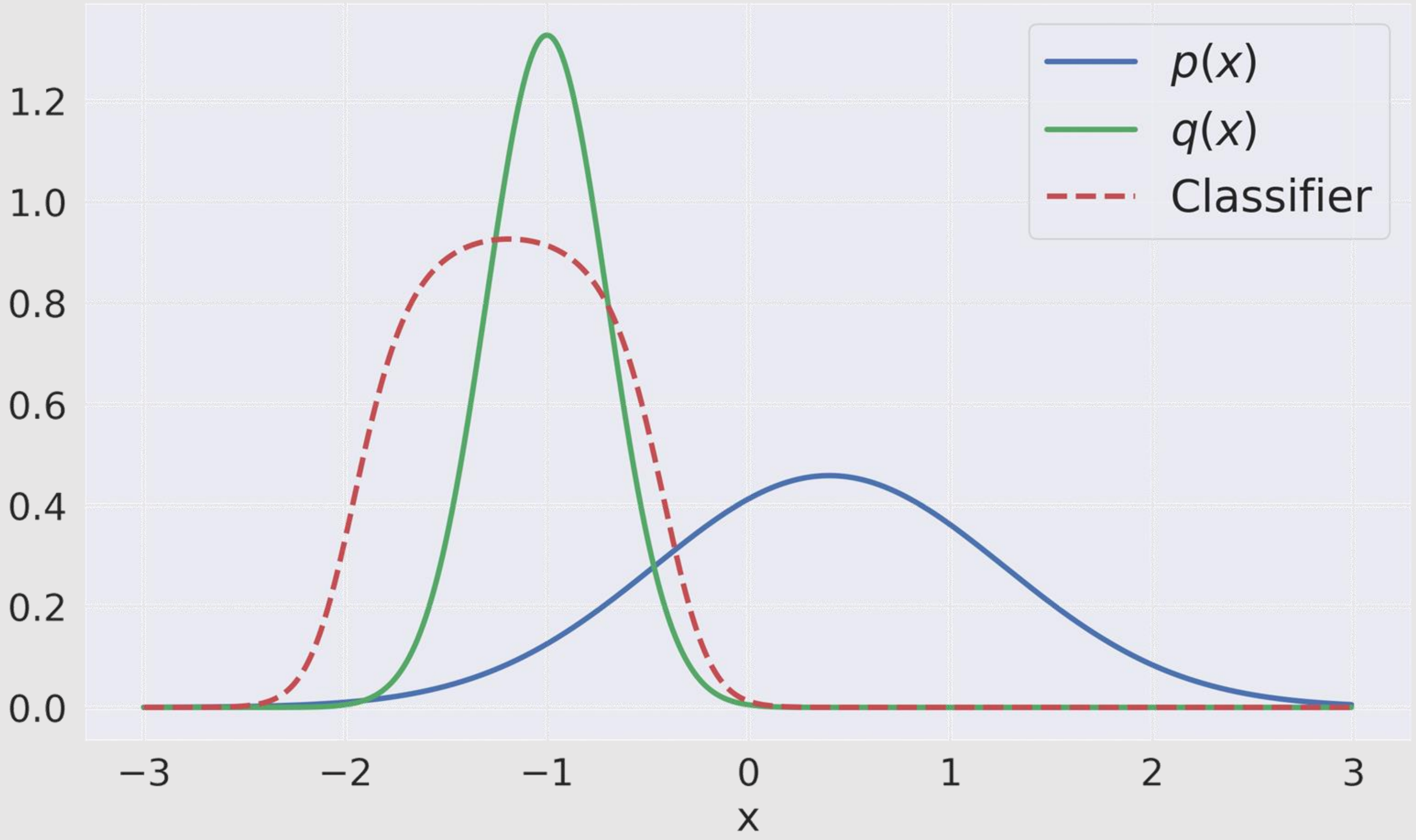
- So far we had fixed *p(x), q(x)* and only trained classifier

- How do we use classifier's output to move *q(x)* towards *p(x)*?

Using classifier's feedback



Using classifier's feedback



Parametrization

1. What do we need to learn a classifier?

Only samples from p(x) and q(x)!

2. How do we parametrize model distribution q(x) ?

Parametrize density function

e. g. $$q_\theta(x) = \mathcal{N}(x; \theta, I)$$

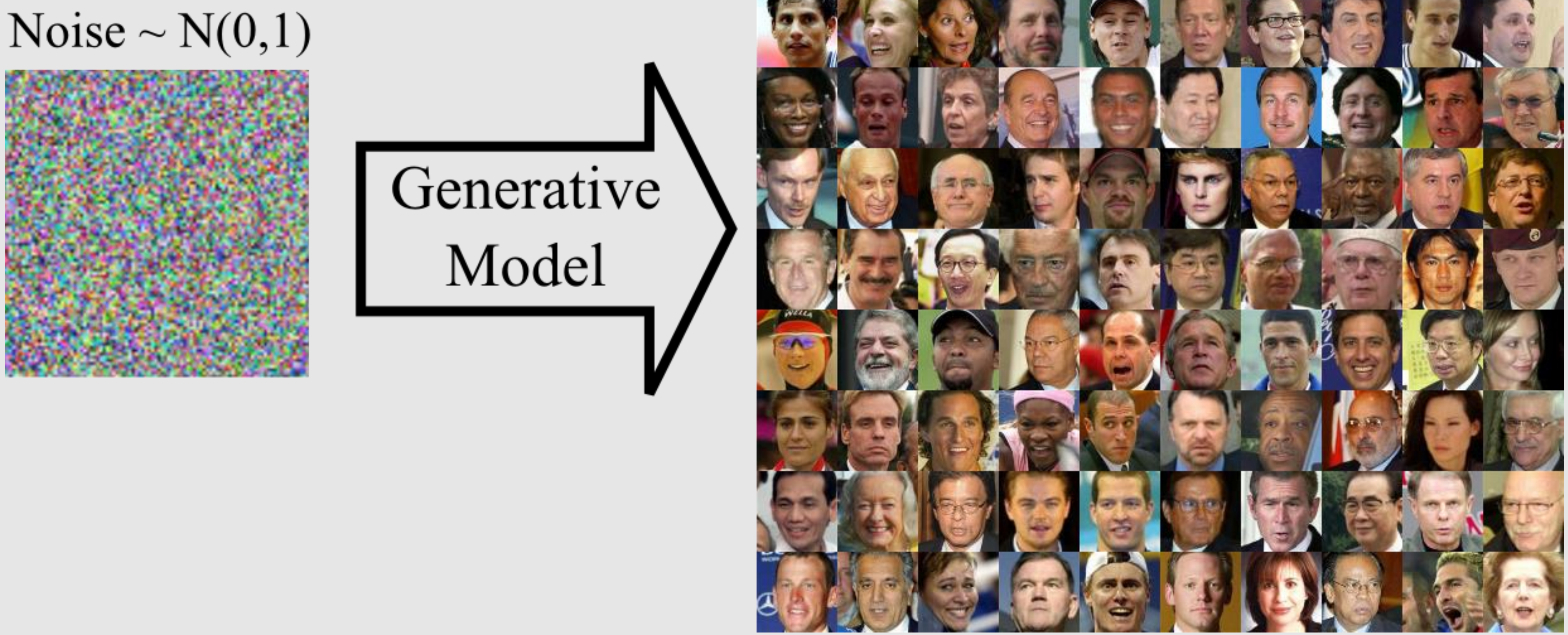
- We should be able to sample from q_θ
- Have access to density at any point.

Define implicitly

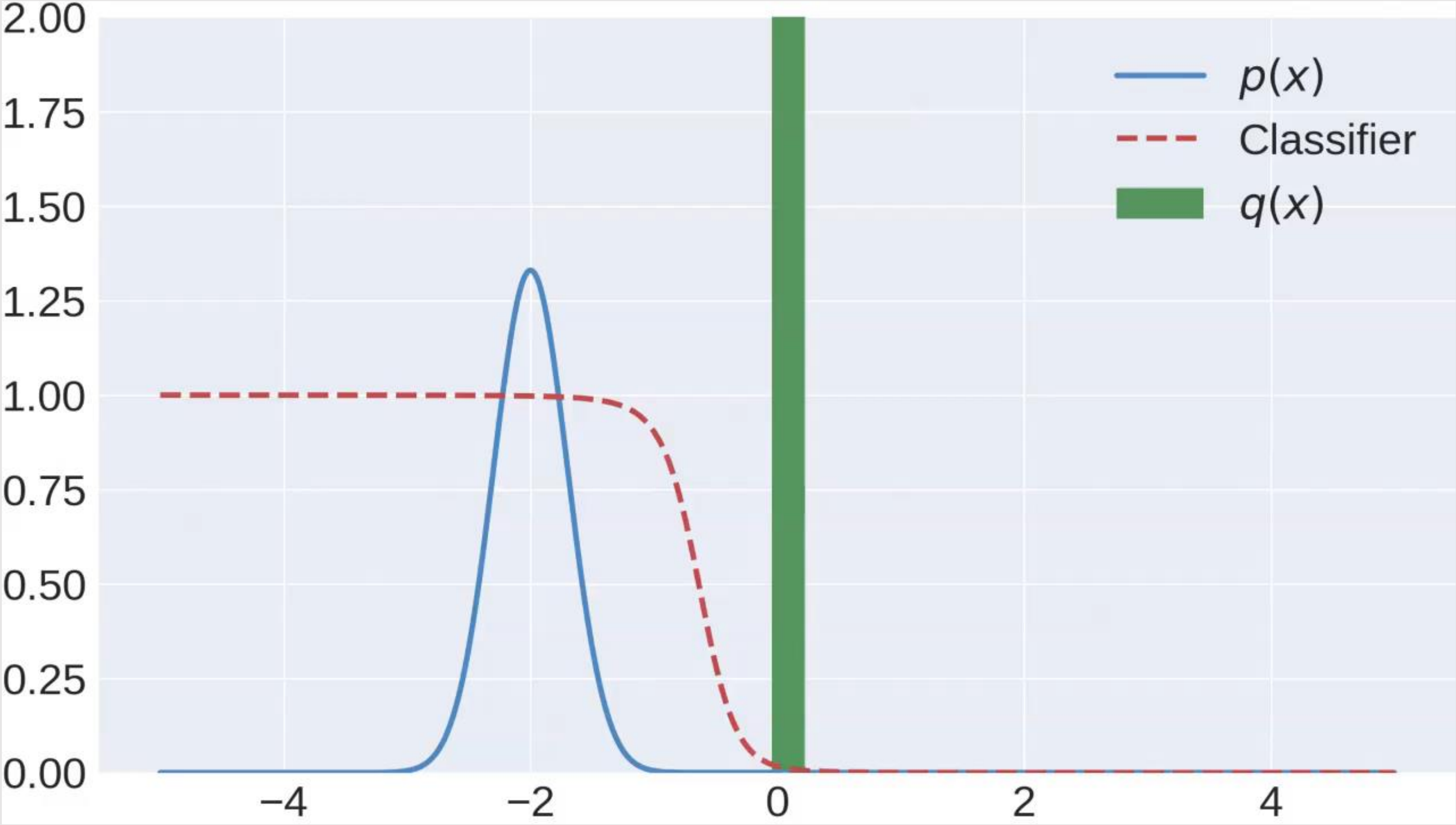
$$z \sim \mathcal{N}(0, I), \quad G_\theta(z) \sim q_\theta(x)$$

- Sampling is always easy
- Hard to evaluate point density $q_\theta(x)$

In case of images



Simulation



GAN Game

- Classifier

$$f_\phi(\mathbf{x}) = p_\phi(y = 1|\mathbf{x})$$

- Classification loss

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{p(\mathbf{x})}[-\log f_\phi(\mathbf{x})] + \mathbb{E}_{p(\mathbf{z})}[-\log(1 - f_\phi(G_\theta(\mathbf{z})))]$$

Algorithm

1. **Update classifier**

$$\phi^* = \arg\min_\phi \mathcal{L}(\phi, \theta)$$

2. **Update generator**

$$\theta^{new} = \theta^{old} + \frac{\partial \mathcal{L}(\phi^*, \theta^{old})}{\partial \theta}$$

3. **Repeat**

In general

General learning scheme:

1. **Update guide**
    - $\frac{p(\mathbf{x})}{p(\mathbf{x})+q(\mathbf{x})}$
    - $\frac{p(\mathbf{x})}{q(\mathbf{x})}$
    - $p(\mathbf{x}) - q(\mathbf{x})$
    - $\mathcal{D}(p \| q)$
2. Use guide to **update generator**
    - Move $q(\mathbf{x})$ closer to $p(\mathbf{x})$
3. **Repeat**

Prescribed vs implicit models

Prescribed (think of VAE)

- $p(z)$
- $q(x)$
- $p(x|z)$
- $q(z|x)$
- $q(x,z)$

Evaluate and sample

Implicit (think of GAN)

- Evaluate and sample from $p(z)$
- Sample from $p(x)$, $q(x)$
- Approximate $q(x)$ using samples
- Approximate $q(z|x)$

GAN Game

- Classification loss

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{p(\mathbf{x})}[-\log f_\phi(\mathbf{x})] + \mathbb{E}_{p(\mathbf{z})}[-\log(1 - f_\phi(G_\theta(\mathbf{z})))]$$

Algorithm

1. **Update classifier**

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

2. **Update generator**

$$\theta^{new} = \theta^{old} + \frac{\partial \mathcal{L}(\phi^*, \theta^{old})}{\partial \theta}$$

3. **Repeat**

GAN Game

- Classification loss

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{p(\mathbf{x})}[-\log f_\phi(\mathbf{x})] + \mathbb{E}_{p(\mathbf{z})}[-\log(1 - f_\phi(G_\theta(\mathbf{z})))]$$

Algorithm

1. **Update classifier**

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

2. **Update generator**

$$\theta^{new} = \theta^{old} + \frac{\partial \mathcal{L}(\phi^*, \theta^{old})}{\partial \theta}$$

3. **Repeat**

GAN Game

- Classification loss

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{p(\mathbf{x})}[-\log f_\phi(\mathbf{x})] + \mathbb{E}_{p(\mathbf{z})}[-\log(1 - f_\phi(G_\theta(\mathbf{z})))]$$

Algorithm

1. **Update classifier**

$$\max_{\phi} -\mathcal{L}(\phi, \theta) = -\log(4) + 2\mathcal{D}_{JS}(p \| q_\theta)$$

2. **Update generator**

$$\theta^{new} = \theta^{old} + \frac{\partial \mathcal{L}(\phi^*, \theta^{old})}{\partial \theta}$$

3. **Repeat**

GAN Game

- Classification loss

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{p(\mathbf{x})}[-\log f_\phi(\mathbf{x})] + \mathbb{E}_{p(\mathbf{z})}[-\log(1 - f_\phi(G_\theta(\mathbf{z})))]$$

Algorithm

Estimate (kind of) distance between $p(\mathbf{x})$ and $q(\mathbf{x})$

1. **Update classifier**

$$\max_\phi -\mathcal{L}(\phi, \theta) = -\log(4) + 2\mathcal{D}_{JS}(p \| q_\theta)$$

2. **Update generator**

$$\theta^{new} = \theta^{old} + \frac{\partial \mathcal{L}(\phi^*, \theta^{old})}{\partial \theta}$$

3. **Repeat**

Minimize the distance

(GAN) Game

- (Classification) loss

$$\mathcal{L}(\phi, \theta) = ?$$

Algorithm

Estimate (kind of) distance between $p(\mathbf{x})$ and $q(\mathbf{x})$

1. **Update classifier**

$$\boxed{\max_{\phi} -\mathcal{L}(\phi, \theta) = \mathcal{D}(p \| q_\theta)}$$

2. **Update generator**

$$\theta^{new} = \theta^{old} + \frac{\partial \mathcal{L}(\phi^*, \theta^{old})}{\partial \theta}$$

3. **Repeat**

Minimize the distance

f-Divergence

- For distributions P and Q **f-divergence** is defined as:

$$D_f(P\|Q) = \int_{\mathcal{X}} f\left(\frac{p(x)}{q(x)}\right) q(x)\,\mathrm{d}x,$$

- **KL-divergence:** $f(t) = t\,log(t)$

$$D_f\left(P \parallel Q\right) = KL\left(P \parallel Q\right)$$

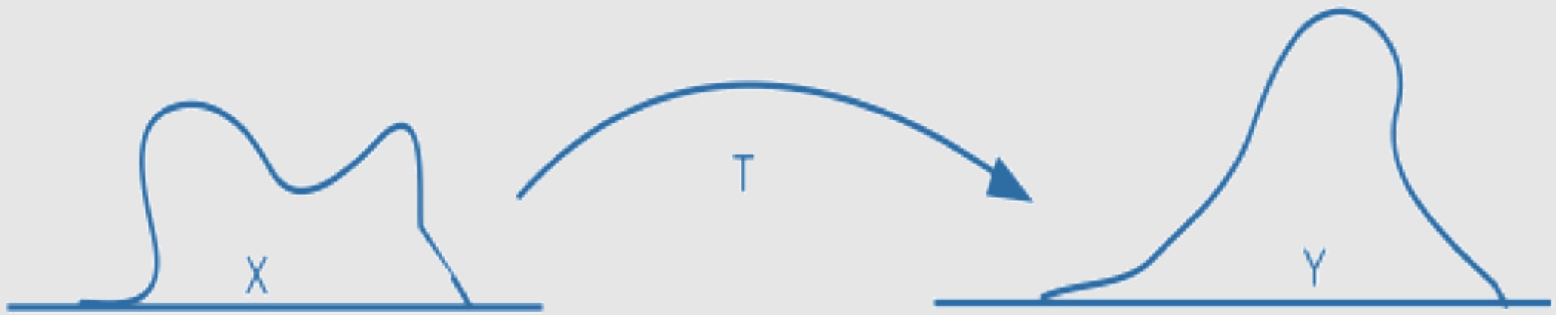
- **Reversed KL-divergence:** $f(t) = -log(t)$

$$D_f\left(P \parallel Q\right) = KL\left(Q \parallel P\right)$$

- **Total variation:** $f(t) = \frac{1}{2}|t - 1|$

$$D_f\left(P \parallel Q\right) = \frac{1}{2}\int_{\mathcal{X}} |p(x) - q(x)|\,\mathrm{d}x$$

Optimal transport

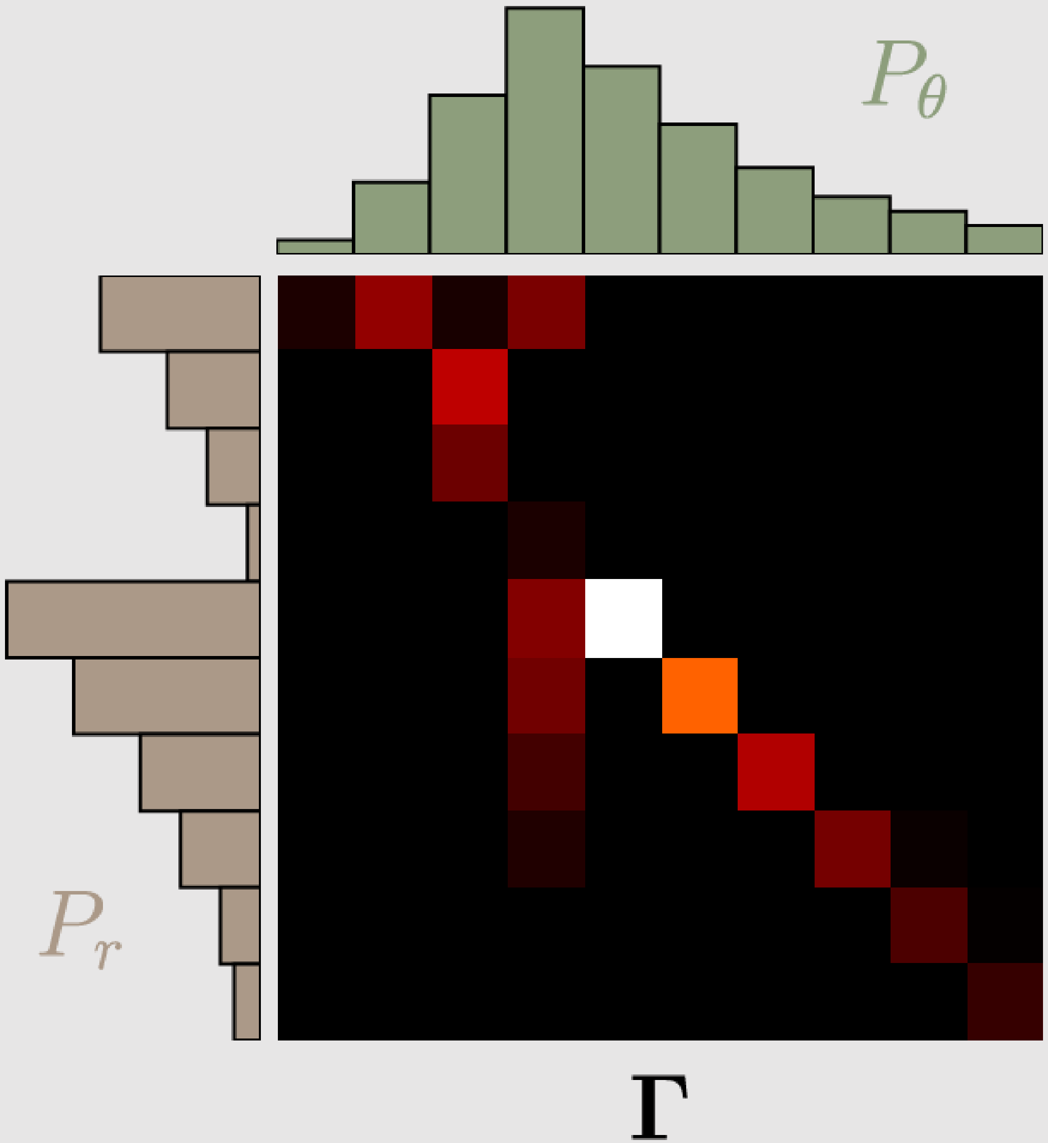


- Define a **cost** of transporting from x to y as $c(x, y)$
  - e.g. $c(x, y) = ||x - y||$
- **Optimal transport** cost is then defined as:

$$T(P, Q) = \inf_{\Gamma \in \mathcal{P}(x \sim P, y \sim Q)} \mathbb{E}_{(x,y)\sim\Gamma}\left[c(x, y)\right]$$

  - where $\mathcal{P}(x \sim P, y \sim Q)$ is a set of all joint distributions of (x, y) with marginals P and Q respectively.

Optimal transport: example



Optimal transport dual

- **Primal**:

$$T(P,Q) = \inf_{\Gamma \in \mathcal{P}(x \sim P, y \sim Q)} \mathbb{E}_{(x,y)\sim\Gamma}\left[c(x,y)\right]$$

- **Dual (Wasserstein-1 metric)**:

$$T(P,Q) = W_1(P,Q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x)$$

How to satisfy lipschitz continuity condition?

Two options:

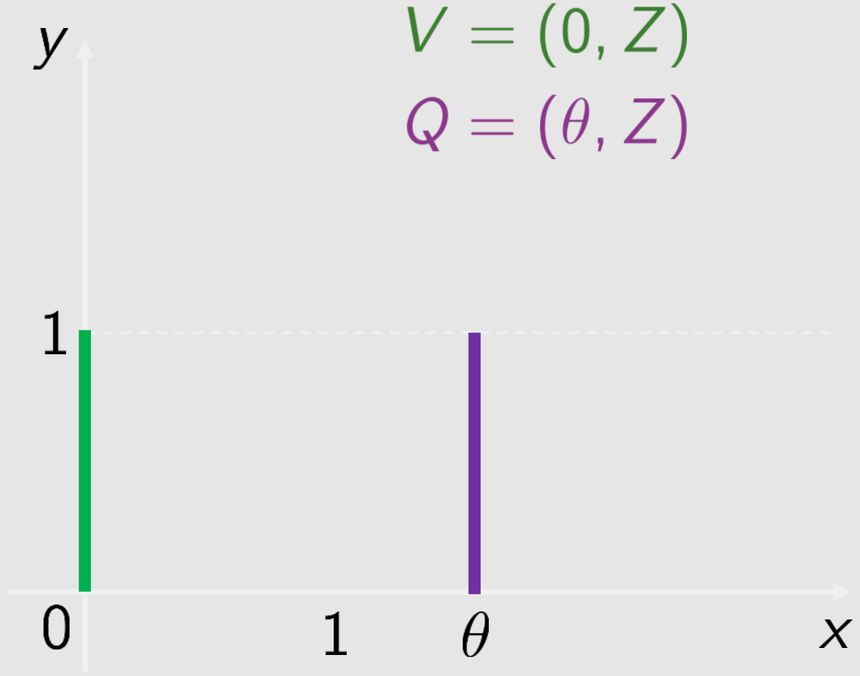
- Weight clipping (used in original WGAN paper)
- Gradient penalty $\lambda \mathbb{E}_{\tilde{x}\sim\mathbb{P}(\tilde{x})}(||\nabla_{\tilde{x}} D(\tilde{x})||_2 - 1)^2$

Optimal transport vs f-Divergence

$$Z \sim U[0,1]$$

$$V = (0, Z)$$

$$Q = (\theta, Z)$$



- $W_1(P, Q) = \theta$
- $JS(P \| Q) = \begin{cases} log(2), & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}$
- $KL(P \| Q) = \begin{cases} \infty, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}$

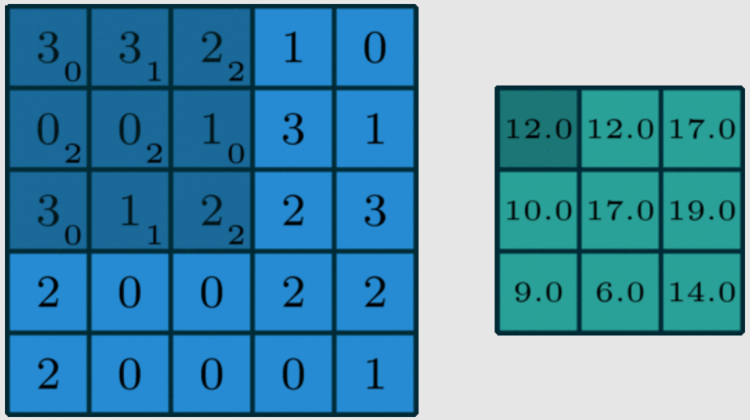
Outline

- Part I
  - Intuition behind GANs
  - Implicit vs explicit generative models
  - JS and f-divergences
  - Wasserstein GAN
- Part II
  - Spectral normalization
  - Moving averaging, weight averaging
  - Quality measurements
  - Applications

Spectral Normalization

- On practice, gradient penalty $\lambda \mathbb{E}_{\tilde{x} \sim \mathbb{P}(\tilde{x})}(||\nabla_{\tilde{x}} D(\tilde{x})||_2 - 1)^2$
  - is a restrictive constraint
  - introduces significant bias

- Is there an easier way to introduce Lipchitz-1 continuity?
  - Reminder of how the convolutions work:



Convolutions as linear operators

$$\underset{x}{\begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix}} \qquad \underset{\omega}{\begin{pmatrix} k_1 & k_2 \\ k_3 & k_4 \end{pmatrix}} \qquad x \circ \omega - ?$$

$$x \circ \omega = \begin{pmatrix} x_1k_1 + x_2k_2 + x_4k_3 + x_5k_4 & x_2k_1 + x_3k_2 + x_5k_3 + x_6k_4 \\ x_1k_4 + x_5k_2 + x_7k_3 + x_8k_4 & x_5k_1 + x_6k_2 + x_8k_3 + x_9k_4 \end{pmatrix}$$

Convolutions as linear operators

$$\underbrace{\begin{pmatrix} k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 \end{pmatrix}}_{A_\omega} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix}}_{\tilde{x}} = \begin{pmatrix} x_1k_1 + x_2k_2 + x_4k_3 + x_5k_4 \\ x_2k_1 + x_3k_2 + x_5k_3 + x_6k_4 \\ x_1k_4 + x_5k_2 + x_7k_3 + x_8k_4 \\ x_5k_1 + x_6k_2 + x_8k_3 + x_9k_4 \end{pmatrix}$$

Lipchitz continuity of a linear operator

$$||f(x_1) - f(x_2)||_2 \leq L||x_1 - x_2||_2$$

$$||Ax_1 - Ax_2||_2 \leq L||x_1 - x_2||_2$$

$$\frac{||Ax||_2}{||x||_2} \leq L$$

$$\sigma_{\max} = \underbrace{\sup_x \frac{||Ax||_2}{||x||_2}}_{\text{Spectral norm}} \leq L$$

Singular values and where to find them

- All singular values can be calculated via the singular value decomposition (SVD):

$$\underbrace{U}_{\text{"left" singular vectors}} \quad A = U\Sigma V^T \quad \underbrace{V}_{\text{"right" singular vectors}}$$

- Given a "left" and "right" singular vectors u and v, the corresponding singular value equals to:

$$\sigma = u^T A v$$

- Knowing that, there is a simpler way to get a maximum singular value

Power iteration

- For simplicity, assume that A is square and diagonalizable (instead of an SVD we can do a simpler EVD: $A = Q\Lambda Q^{-1}$)
  - Let x_1 be initialized at random and have a unit norm
  - Run the following iteration:

$$1.\ x_{i+1} = Ax_i$$

$$2.\ x_{i+1} = \frac{x_{i+1}}{||x_{i+1}||_2}$$

- In the end, we obtain an eigenvector q, corresponding to a maximum eigenvalue $\lambda_{\max}$

$$\lambda_{\max} = q^T A q$$

Theory vs practice

- In practice, instead of A_ω, a reshaped version W of the weights ω is used

$$\underbrace{A_\omega}_{C_{\text{out}}H_{\text{out}}W_{\text{out}} \times C_{\text{in}}H_{\text{in}}W_{\text{in}}} \qquad \underbrace{W}_{C_{\text{out}} \times C_{\text{in}}K_h K_w}$$

- Can be proven that the resulting $\tilde{\sigma}_{\max}$ is a lower bound to the true $\sigma_{\max}$

- This method consistently outperforms "true" spectral normalization

- Takeaway: explanation in the original paper is at least incomplete, further research is needed

Miyato et al., "Spectral Normalization for Generative Adversarial Networks"

Game theory view on GANs

$$\min_{\theta} \max_{\psi} \underbrace{\mathbb{E}_{x \sim p(x)}[\log D_{\psi}(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D_{\psi}(G_{\theta}(z)))]}_{\mathcal{L}(\theta, \phi)}$$

In theory:

1. $\psi^* \;=\; \arg\max_{\psi} \mathcal{L}(\theta^{\text{old}}, \psi)$
2. $\theta^{\text{new}} = \; \theta^{\text{old}} - \alpha \nabla_{\theta} \mathcal{L}(\theta^{\text{old}}, \psi^*)$
3. $\theta^{\text{old}} \;=\; \theta^{\text{new}}$
4. Go to step 1

$\mathcal{D}(P||Q)$

In practice:

1. $\psi^{\text{new}} \;=\; \psi^{\text{old}} + \alpha \nabla_{\psi} \mathcal{L}(\theta^{\text{old}}, \psi^{\text{old}})$
2. $\theta^{\text{new}} \;=\; \theta^{\text{old}} - \alpha \nabla_{\theta} \mathcal{L}(\theta^{\text{old}}, \psi^{\text{new}})$
3. $\theta^{\text{old}} = \theta^{\text{new}}, \quad \psi^{\text{old}} = \psi^{\text{new}}$
4. Go to step 1

Example

Is the saddle point reachable?

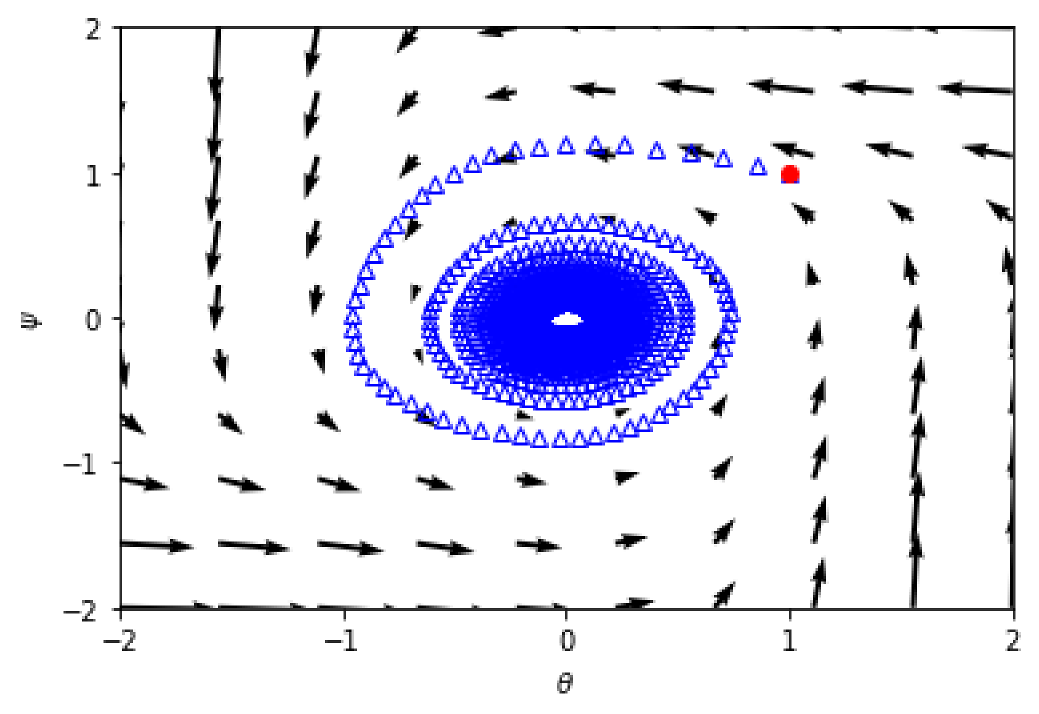
$$\min_\theta \max_\psi \mathbb{E}_{x\sim p(x)} f(D_\psi(x)) + \mathbb{E}_{z\sim p(z)} f(1 - D_\psi(G_\theta(z)))$$

$$\min_\theta \max_\psi \mathbb{E}_{x\sim \delta_0} f(\psi \cdot x) + \mathbb{E}_{\hat{x}\sim \delta_\theta} f(\psi \cdot \hat{x})$$

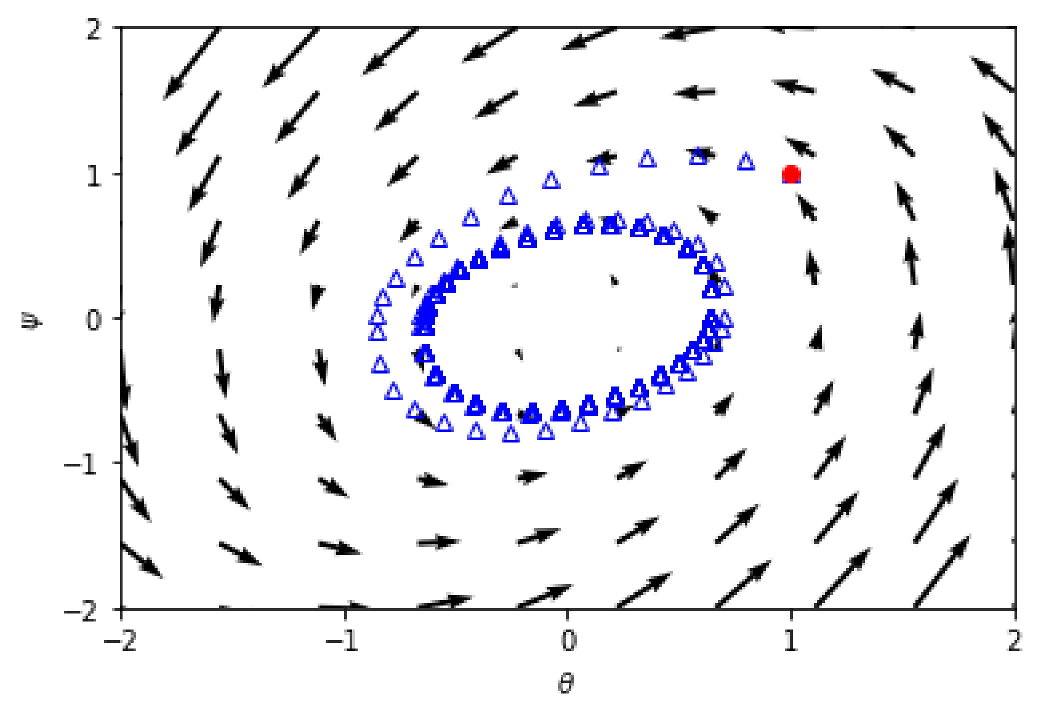
$$\min_\theta \max_\psi f(0) + f(\psi\theta)$$

$$\boxed{\min_\theta \max_\psi f(\psi\theta)}$$

Is the saddle point reachable?



Non-saturating GAN



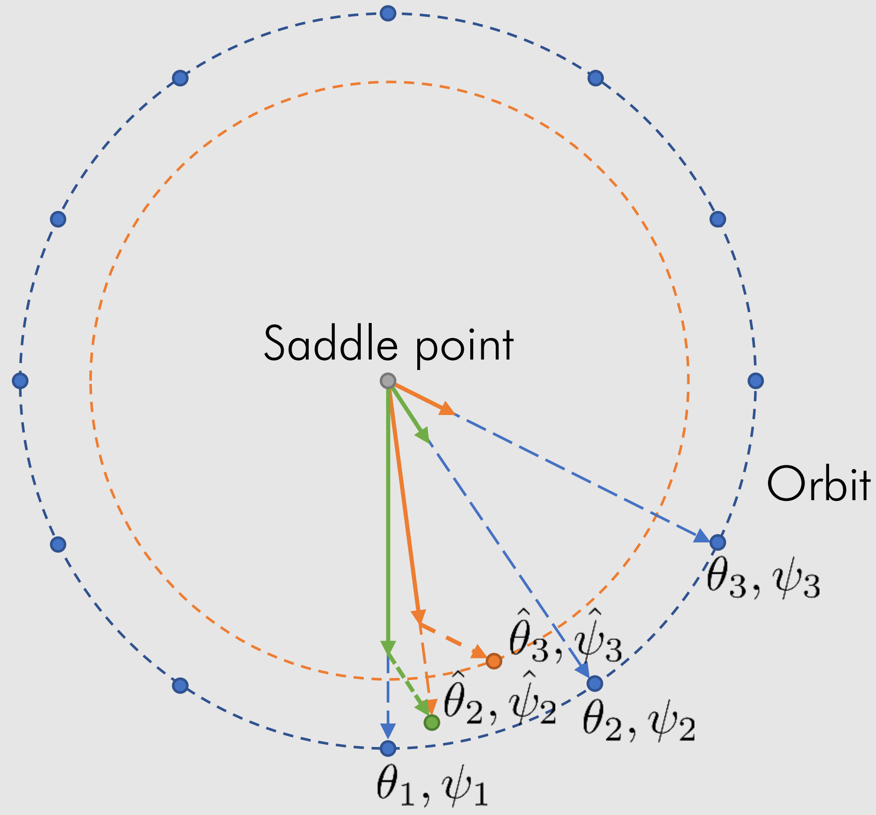
WGAN-GP

Mescheder et al., “Which Training Methods for GANs do actually Converge?”

Exponential moving average

$$\hat{\theta}_{i+1} = (1-\alpha)\hat{\theta}_i + \alpha\theta_{i+1}$$

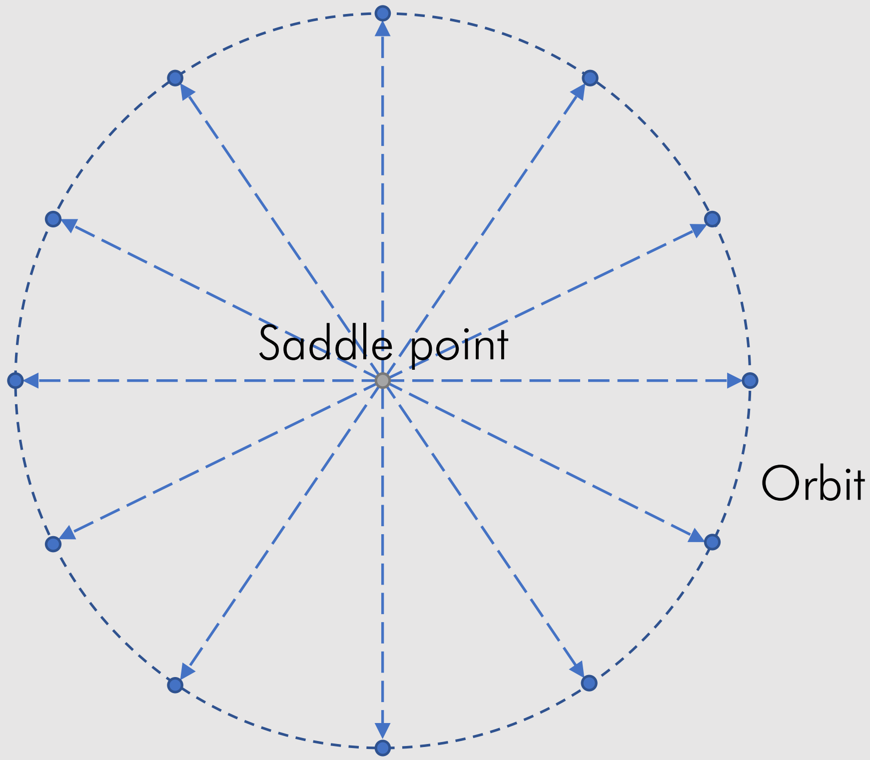
- Reduces the radius of the orbit
- Works “out of the box” (momentum is adjusted according to the planned training time)



Weight averaging

$$\hat{\theta} = \frac{1}{N-n} \sum_{i=n}^{N} \theta_i$$

- Averaging the weights over the orbit gets you directly to the saddle point
- Problem: when to start averaging?



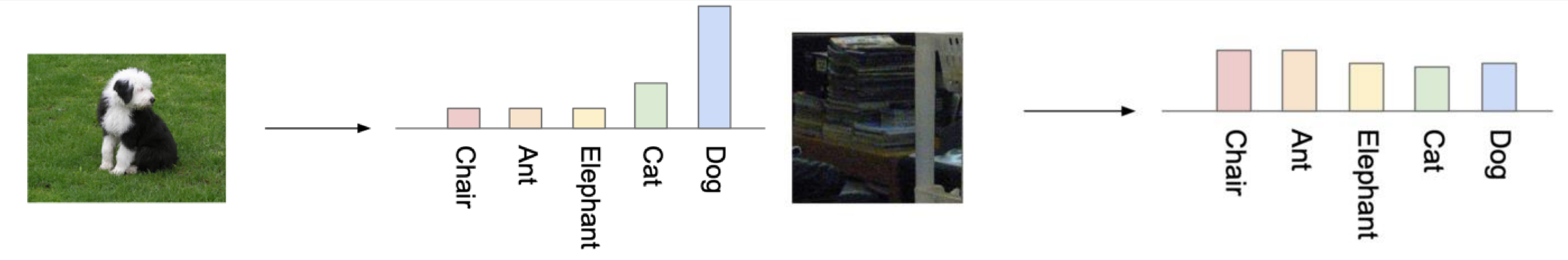
Izmailov et al., “Averaging Weights Leads to Wider Optima and Better Generalization”

Quality measurements

- Assume that you have trained a GAN, how to tell if it’s good?
- All automated metrics suffer from being “bad”, i.e. good models may get low scores
- Best metrics are usually hand-crafted and problem-specific
    - Example: perceptual image quality may be measured via a user study
- Let’s discuss some metrics used for the images

Inception Score (IS)

- Evaluates “objectiveness” and class distribution
  - Assumption: each generated image has a single object in it
  - The overall distribution of the classes has to be uniform



Images credit: David Mack, “A simple explanation of the Inception Score”

Inception Score (IS)

low entropy $p(y|x)$ high entropy $p(y) = \sum_x p(y|x)p(x)$

$$\boxed{\log \mathrm{IS} = \mathbb{E}_x \mathrm{KL}(p(y|x)||p(y))} = \sum_x p(x) \sum_y p(y|x) \log \frac{p(y)}{p(y|x)}$$

$$= \underbrace{\sum_y \log p(y) \sum_x p(y|x)p(x)}_{H(y)} - \underbrace{\sum_x \sum_y p(y|x)p(x) \log p(y|x)}_{H(y|x)}$$

Images credit: David Mack, "A simple explanation of the Inception Score"

Frechet Inception Distance (FID)

- Measures the quality of the generated images
- tl;dr; compare the activations of a pre-trained convnet between real and generated datasets
- We compare only means and pairwise correlations

$$\mathrm{FID} = ||\mu_r - \mu_g||^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

Why you may need GANs

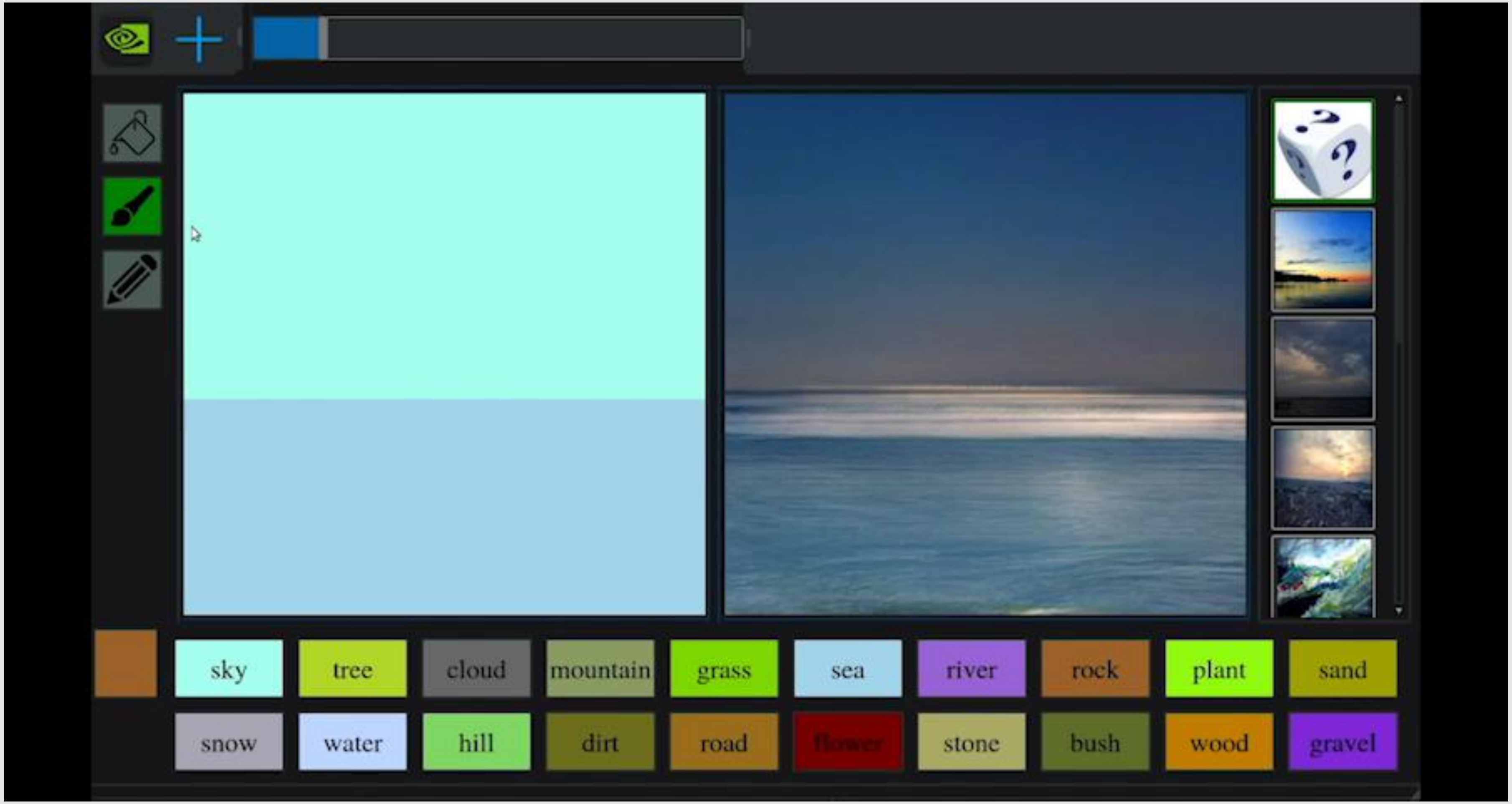
2014 2015 2016 2017 2018

Images credit: Ian Goodfellow, twitter.com/goodfellow_ian

Why you may need GANs

Image credit: Peter Baylies, twitter.com/pbdaylies

Why you may need GANs



Park et al., “Semantic Image Synthesis with Spatially-Adaptive Normalization"

Takeaway message

- GANs are powerful implicit generative models that achieve SotA “quality” of samples
- Recent works show that the improvement in quality of the “outputs” corresponds to improvements in the trained latent space
- Some intuitive explanations of why the GANs work may be incomplete
- Application of GANs and related techniques go far beyond the problems of sampling