Aprendizado Profundo

GANs

Professor: Lucas Silveira Kupssinskü

Agenda

- O que são GANs
- Princípios de Funcionamento
- MinMax Loss
  - Derivação
  - Colapso para a Moda
  - Jensen Shannon Divergence
- Wasserstein GAN

Generative Adversarial Network

- Uma GAN é um modelo não supervisionado que tem como objetivo gerar novas instâncias indistinguíveis do conjunto de dados de treinamento
- Não constroem uma distribuição de probabilidade sobre os dados
- Usualmente é formada por duas redes
  - Gerador: responsável por mapear uma distribuição aleatória para o espaço dos dados
  - Discriminador (crítico): responsável por classificar (criticar) as instâncias entre sintéticas ou reais
- Desenvolvido originalmente por Ian Goodfellow

GANs – Princípios

- Queremos gerar novas instâncias $\{x^{(j)*}\}$ que sigam a mesma distribuição dos dados de treino $\{x^{(i)}\}$
- O processo para gerar uma nova instância é:
  - Escolher uma variável latente $z^{(j)}$ de uma distribuição (normal, uniforme,...)
  - Usar $z^{(j)}$ como entrada para uma rede neural $x^{(j)*} = g[z^{(j)}, \theta]$ onde θ representam os parâmetros da rede que foram aprendidos durante o treino

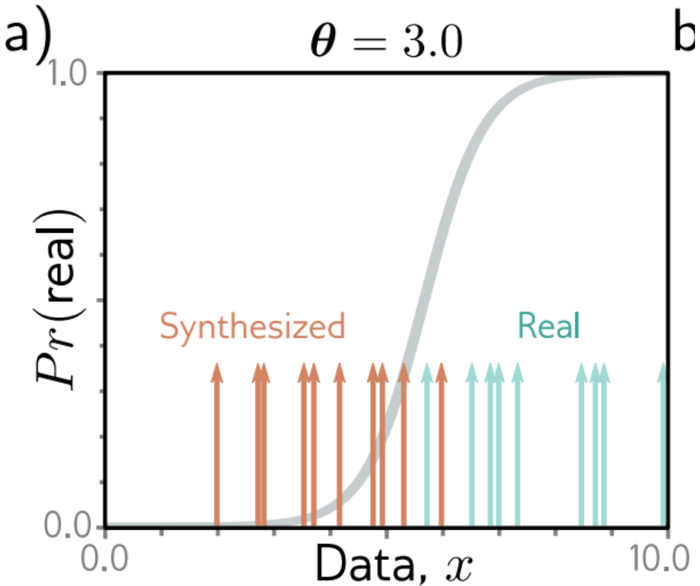
GANs – Princípios

- Teremos atingido o objetivo caso as instâncias geradas sejam indistinguíveis das instâncias do conjunto de dados
- Introduzimos então uma segunda rede $f[\cdot, \phi]$ que vai tentar classificar suas entradas como sendo real ou gerada
  - Se for possível realizar essa classificação, então o discriminador pode gerar um sinal usado para melhorar o processo de geração

GANs – Princípios

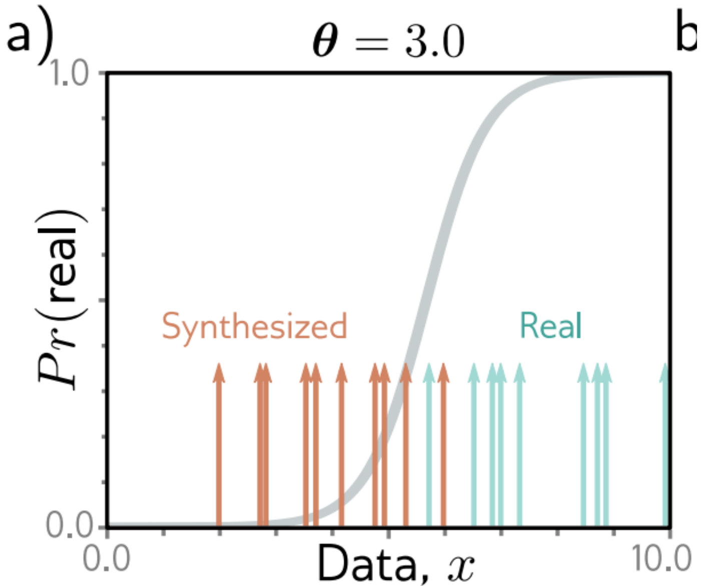
- Um exemplo *toy*
  - Os dados reais foram obtidos usando uma normal $\mathcal{N}(7,1)$
  - O latente é amostrado de uma distribuição $z^{(j)} \sim \mathcal{N}(0,1)$ e o modelo gerador possui apenas um parâmetro que translada essa distribuição

$$x^{(j)*} = z^{(j)} + \theta$$



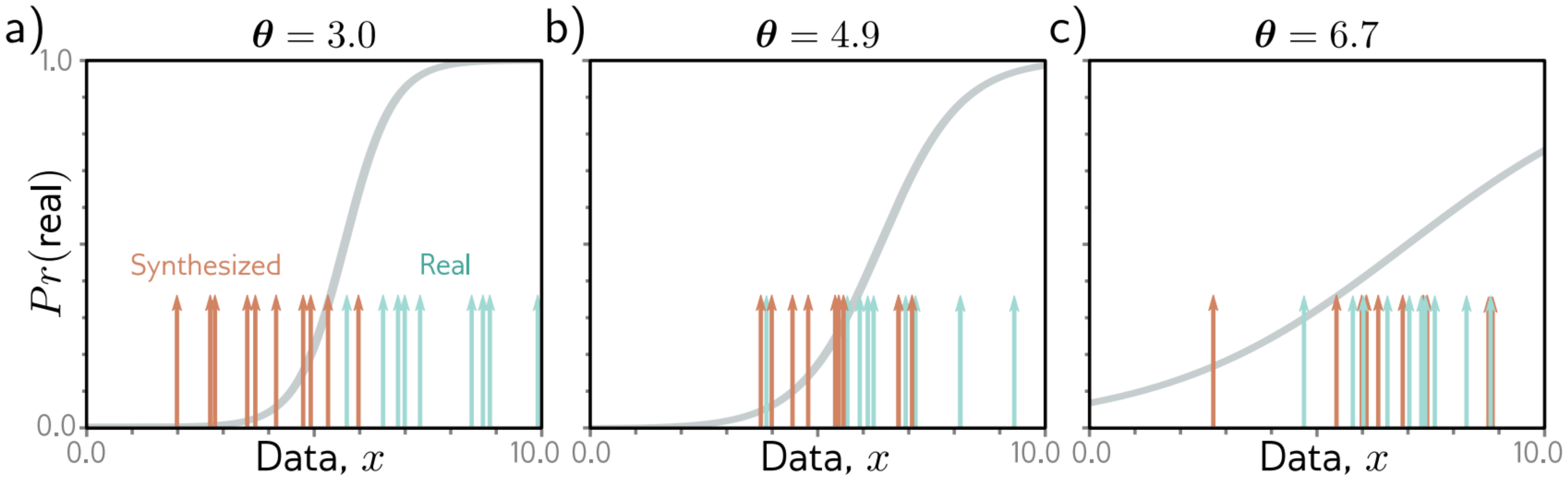
GANs – Princípios

- Um exemplo *toy*
  - A *sigmoid* ilustra como um classificador binário ideal poderia discriminar as duas distribuições



GANs – Princípios

- Um exemplo *toy*
  - A *sigmoid* ilustra como um classificador binário ideal poderia discriminar as duas distribuições
  - Repare como θs diferentes fazem com que o classificador tenha dificuldade em separar as distribuições



Loss

- Para o discriminador, temos um problema de classificação

$$\hat{\phi} = \underset{\phi}{argmin}\left[\sum_i -\left(1-y^{(i)}\right)\ln\left[1-\sigma\left(f\left[x^{(i)},\phi\right]\right)\right] - y^{(i)}\ln\left[\sigma\left(f\left[x^{(i)},\phi\right]\right)\right]\right]$$

Loss

- Para o discriminador, temos um problema de classificação

$$\hat{\phi} = \underset{\phi}{argmin}\left[\sum_i -(1 - y^{(i)})\ln[1 - \sigma(f[x^{(i)}, \phi])] - y^{(i)}\ln[\sigma(f[x^{(i)}, \phi])]\right]$$

- Para tornar mais evidente que algumas instâncias são geradas ($y^{(j)} = 0$), vamos reescrever a *loss*

$$\hat{\phi} = \underset{\phi}{argmin}\left[\sum_j (-\ln[1 - \sigma(f[x^{(j)*}, \phi])]) - \sum_i (\ln[\sigma(f[x^{(i)}, \phi])])\right]$$

Loss

- Agora plugamos a definição da rede geradora e ajustamos o objetivo para maximizar a entropia binária cruzada

$$x^{(j)*} = g[z^{(j)}, \theta]$$

$$\hat{\phi}, \hat{\theta} = \underset{\phi}{\operatorname{argmin}}\left[\underset{\theta}{\operatorname{argmax}}\left[\sum_j \left(-\ln\left[1 - \sigma\left(f\left[g\left[z^{(j)}, \theta\right], \phi\right]\right)\right]\right) - \sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right)\right]\right]$$

Loss

- Agora plugamos a definição da rede geradora e ajustamos o objetivo para maximizar a entropia binária cruzada

$$x^{(j)*} = g[z^{(j)}, \theta]$$

$$\hat{\phi}, \hat{\theta} = \underset{\phi}{\mathrm{argmin}} \left[\underset{\theta}{\mathrm{argmax}} \left[\sum_j \left(-\ln\left[1 - \sigma\left(f\left[g\left[z^{(j)}, \theta\right], \phi\right]\right)\right]\right) - \sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right) \right]\right]$$

Essa é conhecida como a MinMaxLoss

Loss

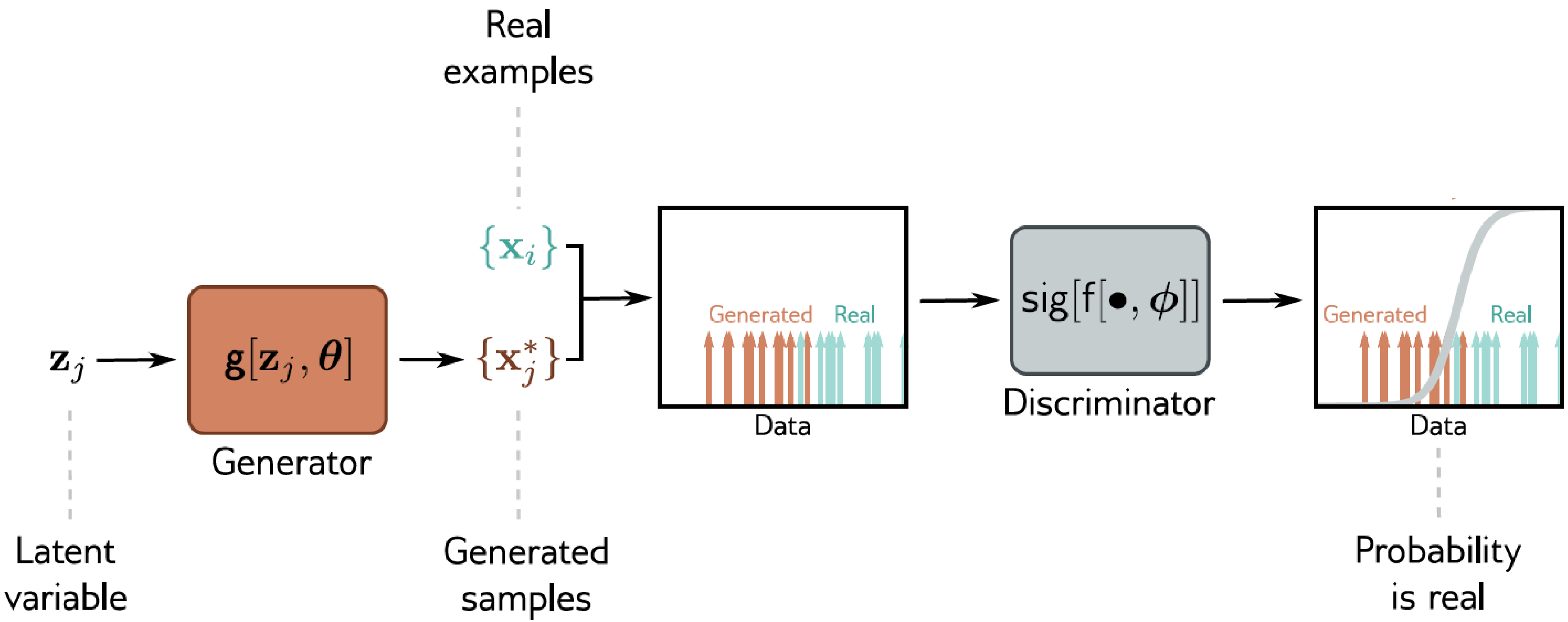
$$\hat{\phi}, \hat{\theta} = \underset{\phi}{\mathrm{argmin}}\left[\underset{\theta}{\mathrm{argmax}}\left[\underbrace{\underbrace{\sum_{j}\left(-\ln\left[1-\sigma\left(f\left[g\left[z^{(j)},\theta\right],\phi\right]\right)\right]\right)}_{\text{Gerador}} - \sum_{i}\left(\ln\left[\sigma\left(f\left[x^{(i)},\phi\right]\right)\right]\right)}_{\text{Discriminador}}\right]\right]$$

Loss

$$\hat{\phi}, \hat{\theta} = \underset{\phi}{\operatorname{argmin}} \left[\underset{\theta}{\operatorname{argmax}} \left[\underbrace{\underbrace{\sum_j \left(-\ln\left[1-\sigma\left(f\left[g\left[z^{(j)},\theta\right],\phi\right]\right)\right]\right)}_{\text{Gerador}} - \sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)},\phi\right]\right)\right]\right)}_{\text{Discriminador}} \right]\right]$$

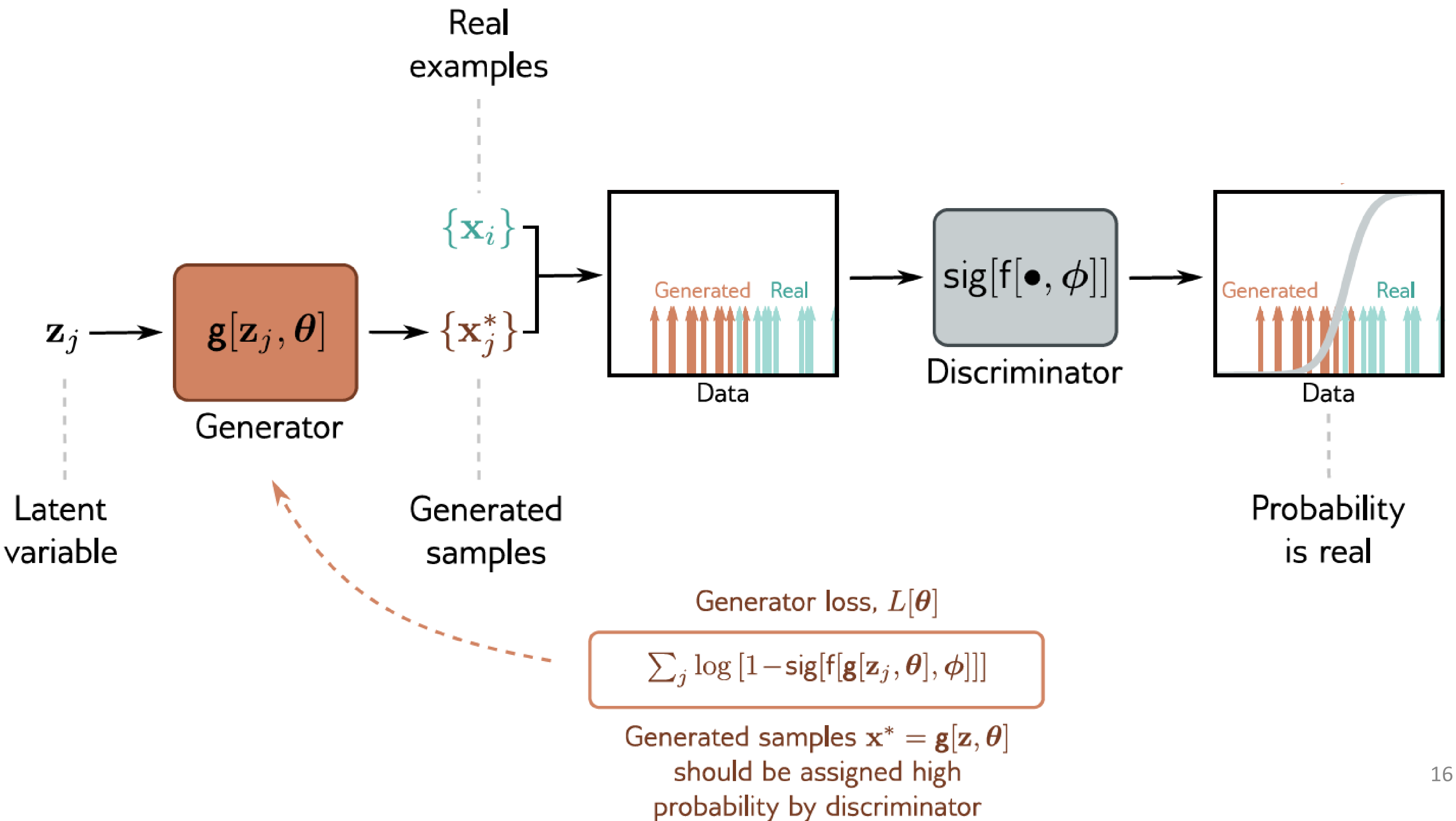
- $L[\phi] = \sum_j \left(-\ln\left[1-\sigma\left(f\left[g\left[z^{(j)},\theta\right],\phi\right]\right)\right]\right) - \sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)},\phi\right]\right)\right]\right)$
- $L[\theta] = \sum_j \left(\ln\left[1-\sigma\left(f\left[g\left[z^{(j)},\theta\right],\phi\right]\right)\right]\right)$

Loss



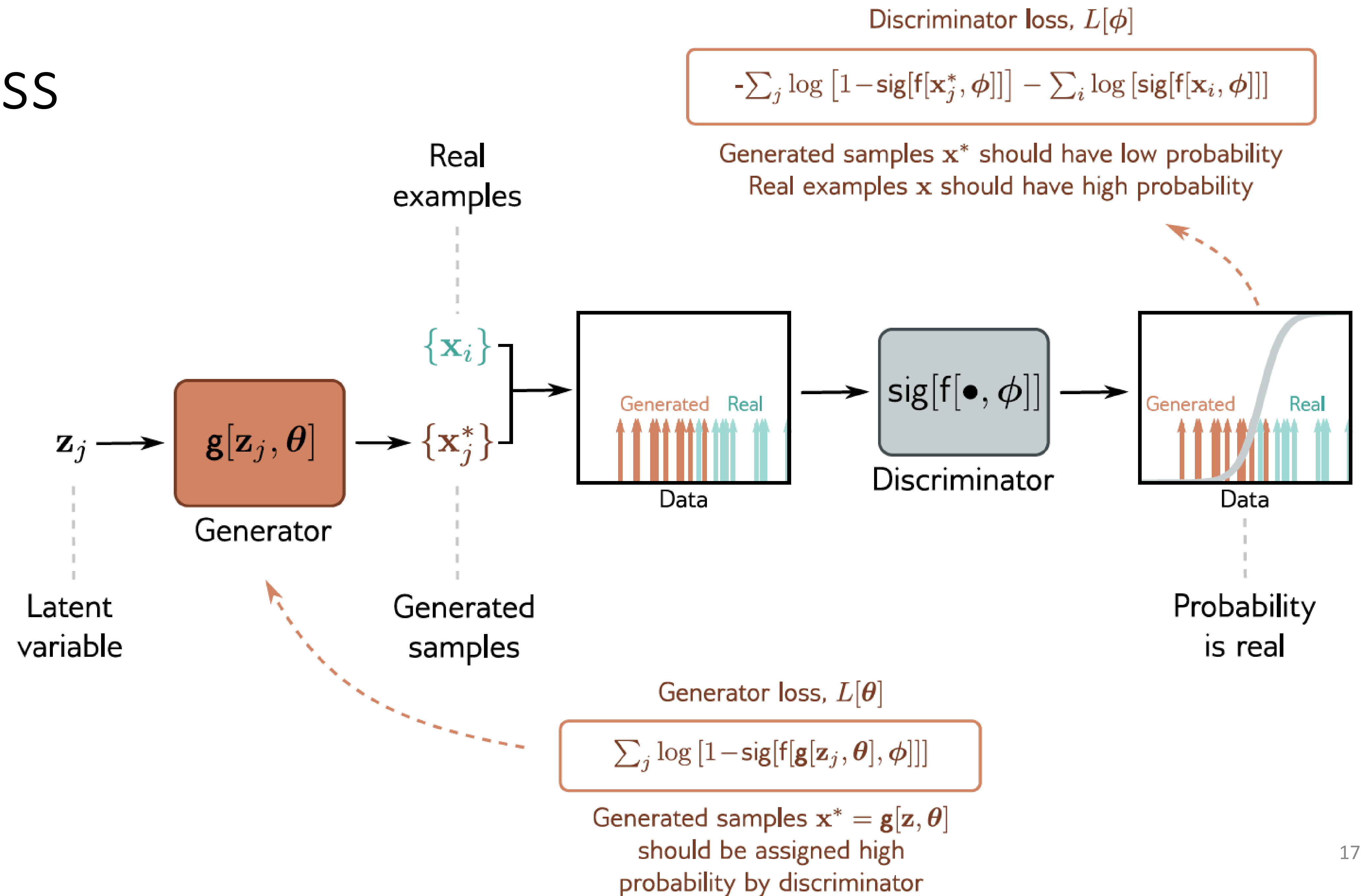
Real
examples
$\{\mathbf{x}_i\}$
$\mathbf{z}_j$
$\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}]$
Generator
$\{\mathbf{x}_j^*\}$
Generated
Real
Data
$\text{sig}[\text{f}[\bullet, \phi]]$
Discriminator
Generated
Real
Data
Latent
variable
Generated
samples
Probability
is real

Loss



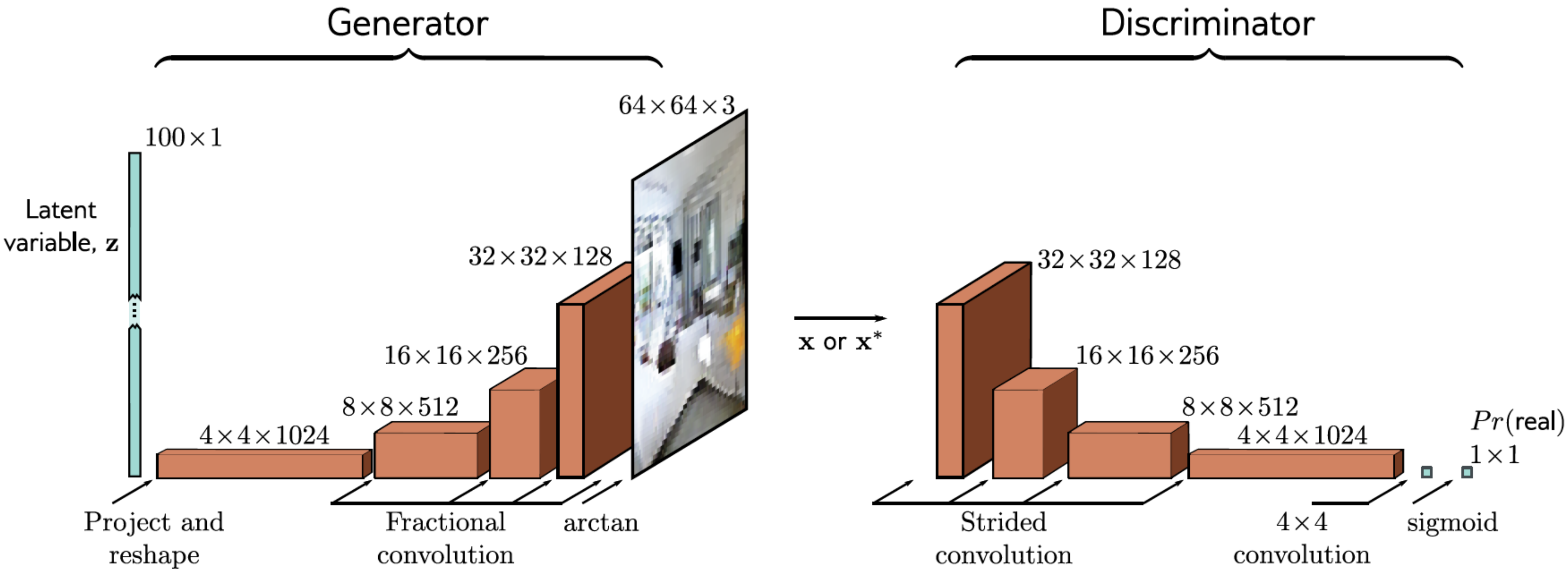
Real
examples
$\{\mathbf{x}_i\}$
$\mathbf{z}_j$
$\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}]$
Generator
$\{\mathbf{x}^*_j\}$
Generated Real
Data
$\text{sig}[\text{f}[\bullet, \phi]]$
Discriminator
Generated Real
Data
Latent
variable
Generated
samples
Probability
is real
Generator loss, $L[\boldsymbol{\theta}]$
$\sum_j \log\left[1-\text{sig}[\text{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \phi]]\right]$
Generated samples $\mathbf{x}^* = \mathbf{g}[\mathbf{z}, \boldsymbol{\theta}]$
should be assigned high
probability by discriminator

Loss



Discriminator loss, $L[\phi]$
$-\sum_j \log\left[1 - \text{sig}[\text{f}[\mathbf{x}_j^*, \phi]]\right] - \sum_i \log\left[\text{sig}[\text{f}[\mathbf{x}_i, \phi]]\right]$
Generated samples $\mathbf{x}^*$ should have low probability
Real examples $\mathbf{x}$ should have high probability
Real examples
$\{\mathbf{x}_i\}$
$\mathbf{z}_j$
$\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}]$
Generator
$\{\mathbf{x}_j^*\}$
Generated
Real
Data
$\text{sig}[\text{f}[\bullet, \phi]]$
Discriminator
Generated
Real
Data
Latent variable
Generated samples
Probability is real
Generator loss, $L[\boldsymbol{\theta}]$
$\sum_j \log\left[1 - \text{sig}[\text{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \phi]]\right]$
Generated samples $\mathbf{x}^* = \mathbf{g}[\mathbf{z}, \boldsymbol{\theta}]$ should be assigned high probability by discriminator

DCGAN



RADFORD, Alec; METZ, Luke; CHINTALA, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. **arXiv preprint arXiv:1511.06434**, 2015.

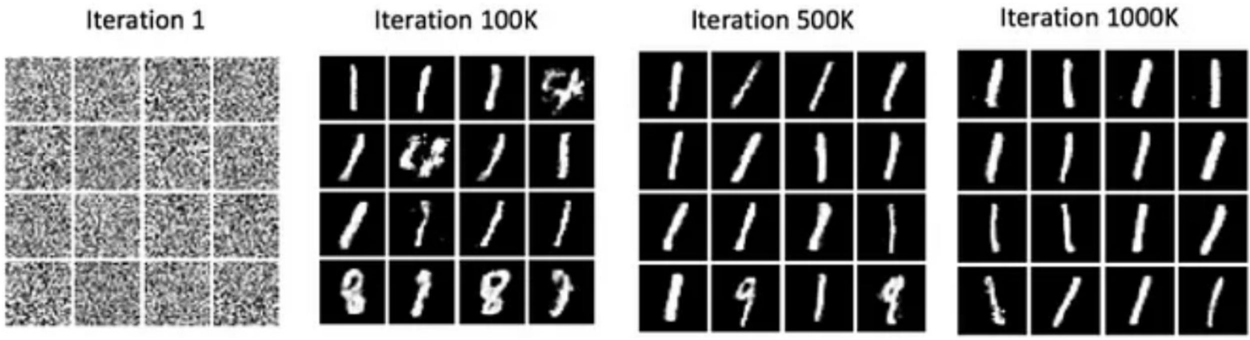
DCGAN

Figure 15.4 Synthesized images from the DCGAN model. a) Random samples drawn from DCGAN trained on a faces dataset. b) Random samples using the ImageNet database (see figure 10.15). c) Random samples drawn from the LSUN scene understanding dataset. Adapted from Radford et al. (2015).

Mode Collapse

- É comum que o gerador “colapse para a moda” da distribuição, gerando instâncias sem diversidade
- O motivo disso fica aparente ao analisarmos a função de custo

Figure 15.5 Mode collapse. Synthesized images from a GAN trained on the LSUN scene understanding dataset using an MLP generator with a similar number of parameters and layers to the DCGAN. The samples are low quality, and many are similar. Adapted from Arjovsky et al. (2017).



Análise da Loss

$$L[\phi] = \sum_j \left(\ln\left[1 - \sigma\left(f\left[x^{(j)*}, \phi\right]\right)\right]\right) + \sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right)$$

$$L[\phi] \approx \frac{1}{J}\sum_j \left(\ln\left[1 - \sigma\left(f\left[x^{(j)*}, \phi\right]\right)\right]\right) + \frac{1}{I}\sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right)$$

Podemos modificar a loss da seguinte forma:
Dividimos cada termo pelo número de instâncias geradas J ou reais I

Análise da Loss

$$L[\phi] = \sum_j \left(\ln\left[1 - \sigma\left(f\left[x^{(j)*}, \phi\right]\right)\right]\right) + \sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right)$$

$$L[\phi] \approx \frac{1}{J}\sum_j \left(\ln\left[1 - \sigma\left(f\left[x^{(j)*}, \phi\right]\right)\right]\right) + \frac{1}{I}\sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right)$$

$$L[\phi] = \mathbb{E}_{x^*}\left[\ln[1 - \sigma(f[x^*, \phi])]\right] + \mathbb{E}_x[\ln[\sigma(f[x, \phi])]]$$

Ao fazer isso cada termo vira uma esperança/expectativa

Análise da Loss

$$L[\phi] = \sum_j \left(\ln\left[1 - \sigma\left(f\left[x^{(j)*}, \phi\right]\right)\right]\right) + \sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right)$$

$$L[\phi] \approx \frac{1}{J}\sum_j \left(\ln\left[1 - \sigma\left(f\left[x^{(j)*}, \phi\right]\right)\right]\right) + \frac{1}{I}\sum_i \left(\ln\left[\sigma\left(f\left[x^{(i)}, \phi\right]\right)\right]\right)$$

$$L[\phi] = \mathbb{E}_{x^*}\left[\ln\left[1 - \sigma\left(f\left[x^*, \phi\right]\right)\right]\right] + \mathbb{E}_{x}\left[\ln\left[\sigma\left(f\left[x, \phi\right]\right)\right]\right]$$

$$\boxed{L[\phi] = \int P(x^*) \ln\left[1 - \sigma\left(f\left[x^*, \phi\right]\right)\right] dx^* + \int P(x) \ln\left[\sigma\left(f\left[x, \phi\right]\right)\right] dx}$$

Aplicando a definição de esperança/expectativa temos duas integrais

Análise da Loss

- Suponha que $I = J$, nesse caso o discriminador ótimo de uma instância desconhecida $\tilde{x}$ segue:

$$P(real|\tilde{x}) = \sigma\big[f[\tilde{x}, \phi]\big] = \frac{P(\tilde{x}|real)}{P(\tilde{x}|real) + P(\tilde{x}|gen)} = \frac{P(x)}{P(x) + P(x^*)}$$

Análise da Loss

$$P(real|\tilde{x}) = \sigma\left[f[\tilde{x},\phi]\right] = \frac{P(\tilde{x}|real)}{P(\tilde{x}|real) + P(\tilde{x}|gen)} = \boxed{\frac{P(x)}{P(x) + P(x^*)}}$$

$$L[\phi] = \int P(x^*)\ln[1 - \boxed{\sigma(f[x^*,\phi])}]dx^* + \int P(x)\ln\boxed{[\sigma(f[x,\phi])]}dx$$

$$L[\phi] = \int P(x^*)\ln\left[1 - \frac{P(x)}{P(x) + P(x^*)}\right]dx^* + \int P(x)\ln\left[\frac{P(x)}{P(x) + P(x^*)}\right]dx$$

$$L[\phi] = \int P(x^*)\ln\left[\frac{P(x^*)}{P(x) + P(x^*)}\right]dx^* + \int P(x)\ln\left[\frac{P(x)}{P(x) + P(x^*)}\right]dx$$

Análise da Loss

- Por conveniência matemática vamos fazer uma manipulação que não altera o resultado final ☺

$$L[\phi] = \frac{1}{2}\int P(x^*)\ln\left[\frac{2P(x^*)}{P(x)+P(x^*)}\right]dx^* + \frac{1}{2}\int P(x)\ln\left[\frac{2P(x)}{P(x)+P(x^*)}\right]dx$$

Análise da Loss

$$L[\phi] = \frac{1}{2}\boxed{\int P(x^*) \ln\left[\frac{2P(x^*)}{P(x) + P(x^*)}\right] dx^*} + \frac{1}{2}\boxed{\int P(x) \ln\left[\frac{2P(x)}{P(x) + P(x^*)}\right] dx}$$

Repare que, cada um desses termos é muito semelhante a *KL Divergence*:

$$D_{KL}[p(x)||q(x)] = \boxed{\int p(x) \ln\left[\frac{p(x)}{q(x)}\right] dx}$$

Análise da Loss

Reescrevendo em termos de *KL Divergence*

$$L[\phi] = \frac{1}{2} D_{KL}\left[P(x^*) || \frac{P(x^*) + P(x)}{2}\right] + \frac{1}{2} D_{KL}\left[P(x) || \frac{P(x^*) + P(x)}{2}\right]$$

Obtemos a definição da *Jensen-Shannon Divergence*

$$D_{JS}[P(x^*) || P(x)] = \frac{1}{2} D_{KL}\left[P(x^*) || \frac{P(x^*) + P(x)}{2}\right] + \frac{1}{2} D_{KL}\left[P(x) || \frac{P(x^*) + P(x)}{2}\right]$$

Análise da Loss

$$D_{JS}[P(x^*)||P(x)] = \underbrace{\frac{1}{2}D_{KL}\left[P(x^*)||\ \frac{P(x^*)+P(x)}{2}\right]}_{} + \underbrace{\frac{1}{2}D_{KL}\left[P(x)||\frac{P(x^*)+P(x)}{2}\right]}_{}$$

- O primeiro termo penaliza regiões nas quais a mistura de probabilidades $\frac{P(x^*)+P(x)}{2}$ tem probabilidade alta porém a distribuição gerada $P(x^*)$ não tem
- O segundo termo penaliza regiões nas quais a mistura de probabilidades $\frac{P(x^*)+P(x)}{2}$ tem probabilidade alta porém a distribuição original $P(x)$ não tem

Análise da Loss

$$D_{JS}[P(x^*)||P(x)] = \underbrace{\frac{1}{2}D_{KL}\left[P(x^*)||\ \frac{P(x^*)+P(x)}{2}\right]}_{} + \underbrace{\frac{1}{2}D_{KL}\left[P(x)||\frac{P(x^*)+P(x)}{2}\right]}_{}$$

- O primeiro termo penaliza regiões nas quais a distribuição gerada $P(x^*)$ possui alta probabilidade porém a mistura de probabilidades $\frac{P(x^*)+P(x)}{2}$ não tem
- O segundo termo penaliza regiões nas quais a distribuição original $P(x)$ possui alta probabilidade porém a mistura de probabilidades $\frac{P(x^*)+P(x)}{2}$ não tem

Você consegue identificar qual dos termos avalia a **qualidade** e qual avalia a **cobertura** das instâncias geradas?

Análise da Loss

$$D_{JS}[P(x^*)||P(x)] = \underbrace{\frac{1}{2}D_{KL}\left[P(x^*)||\ \frac{P(x^*)+P(x)}{2}\right]}_{\text{Qualidade}} + \underbrace{\frac{1}{2}D_{KL}\left[P(x)||\frac{P(x^*)+P(x)}{2}\right]}_{\text{Cobertura}}$$

- Repare que o segundo termo não depende do gerador que, por consequência, vai tentar gerar instâncias com qualidade independente da cobertura

Vanishing Gradient

- Vimos que, no caso de um discriminador ótimo, a função de custo minimiza uma distância entre as instâncias reais e as instâncias geradas
- Contudo, essa abordagem traz outros problemas
  - Caso as distribuições de probabilidades sejam disjuntas a distância entre elas (conforme definido anteriormente) é infinita
    - Nesse caso, nenhuma “pequena” mudança pode melhorar a loss

$$D_{KL}[p(x)||q(x)] = \int p(x) \ln \left[\frac{p(x)}{q(x)}\right] dx$$

Vanishing Gradient

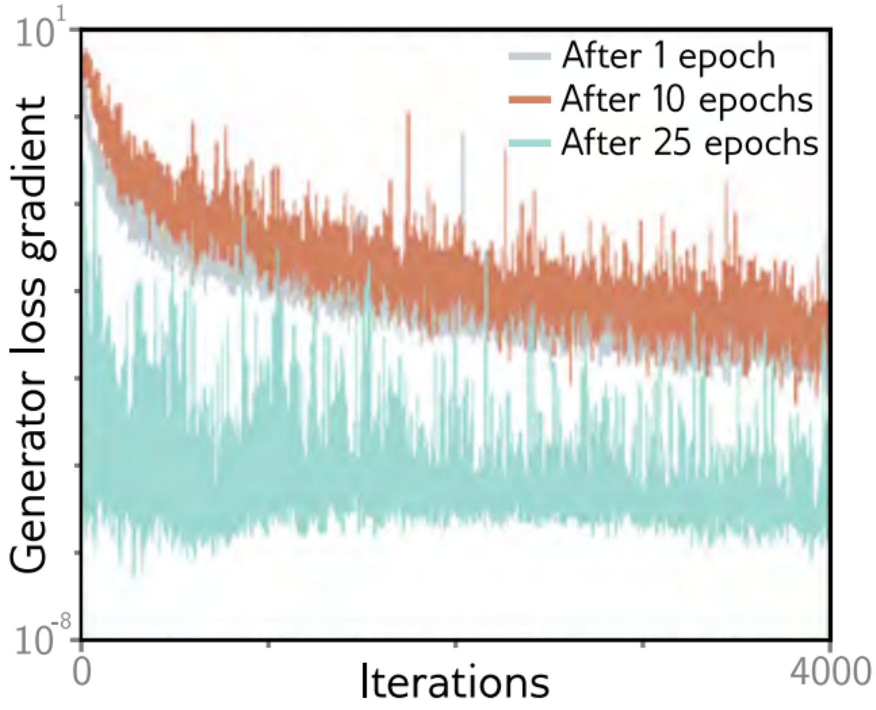


Figure 15.7 Vanishing gradients in the generator of a DCGAN. The generator is frozen after 1, 10, and 25 epochs, and the discriminator is trained further. The gradient of the generator decreases rapidly (note log scale); if the discriminator becomes too accurate, the gradients for the generator vanish. Adapted from Arjovsky & Bottou (2017).

Loss

- Vimos que a loss original de GANs pode ser interpretada como uma distância entre duas distribuições de probabilidades
    - Porém é uma distância com propriedades que não são adequadas para otimização (por exemplo, distribuições disjuntas tem distância infinita)
- Vamos escolher uma distância com propriedades mais adequadas para otimização

Wasserstein Distance

- Também conhecida como *Earth Mover's Distance* (para distribuições discretas
- É definida pela quantidade de trabalho necessária para transportar uma massa de probabilidade de uma distribuição para transforma-la em outra
  - Trabalho é massa multiplicada pela distância movida

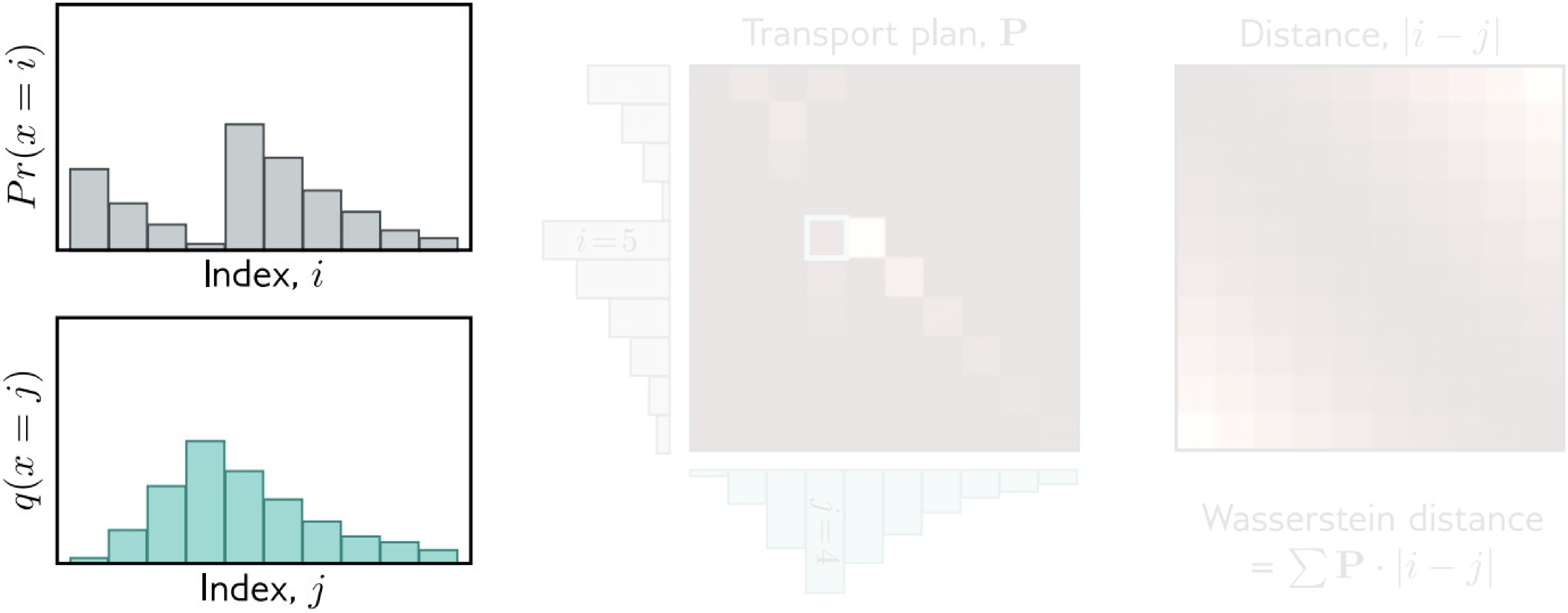
Wasserstein Distance

- Vamos analisar o caso discreto primeiro

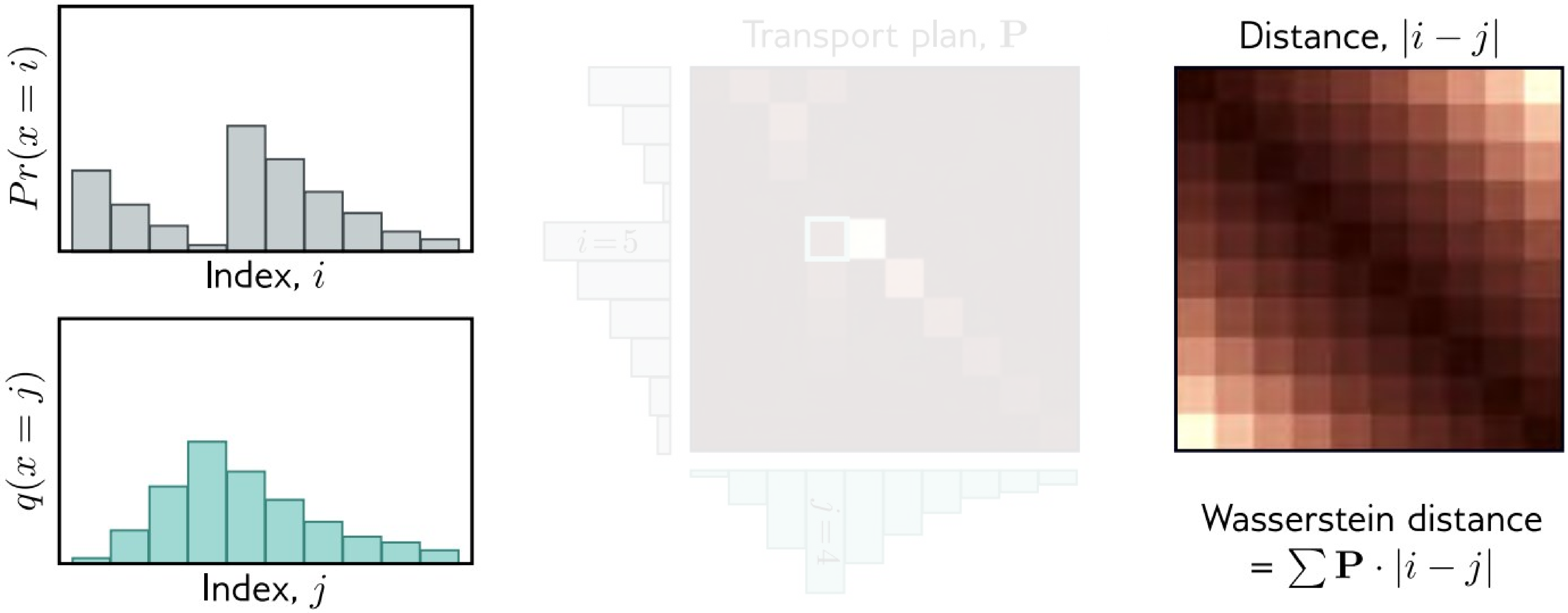
$$D_w[Pr(x)||Qr(x)] = \min_P \left[\sum_{i,j} P_{ij}|i-j| \right]$$

- Onde P armazena o "*plano de transporte*", indicando quanto de massa de probabilidade deve ser movida de i para j
- Sujeito as seguintes restrições:
  - $\sum_j P_{ij} = P(x=i)$
  - $\sum_i P_{ij} = Q(x=i)$
  - $P_{ij} \geq 0$

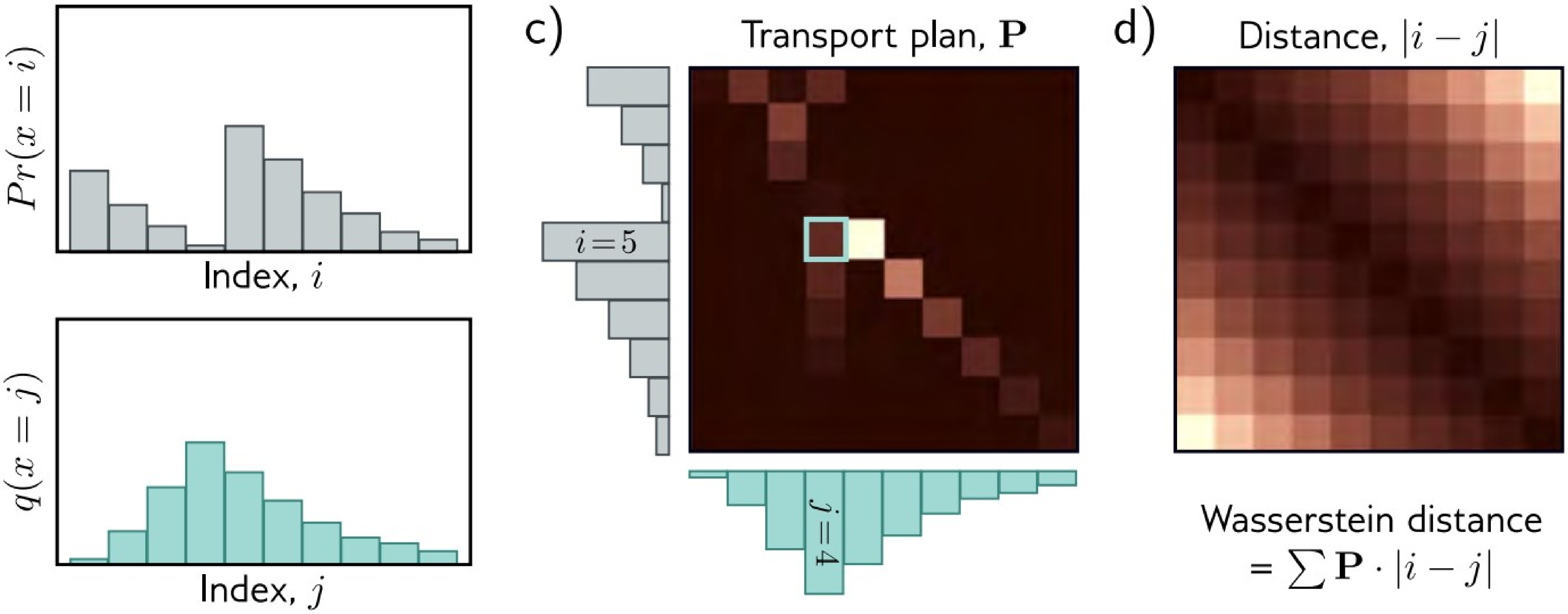
Wasserstein Distance



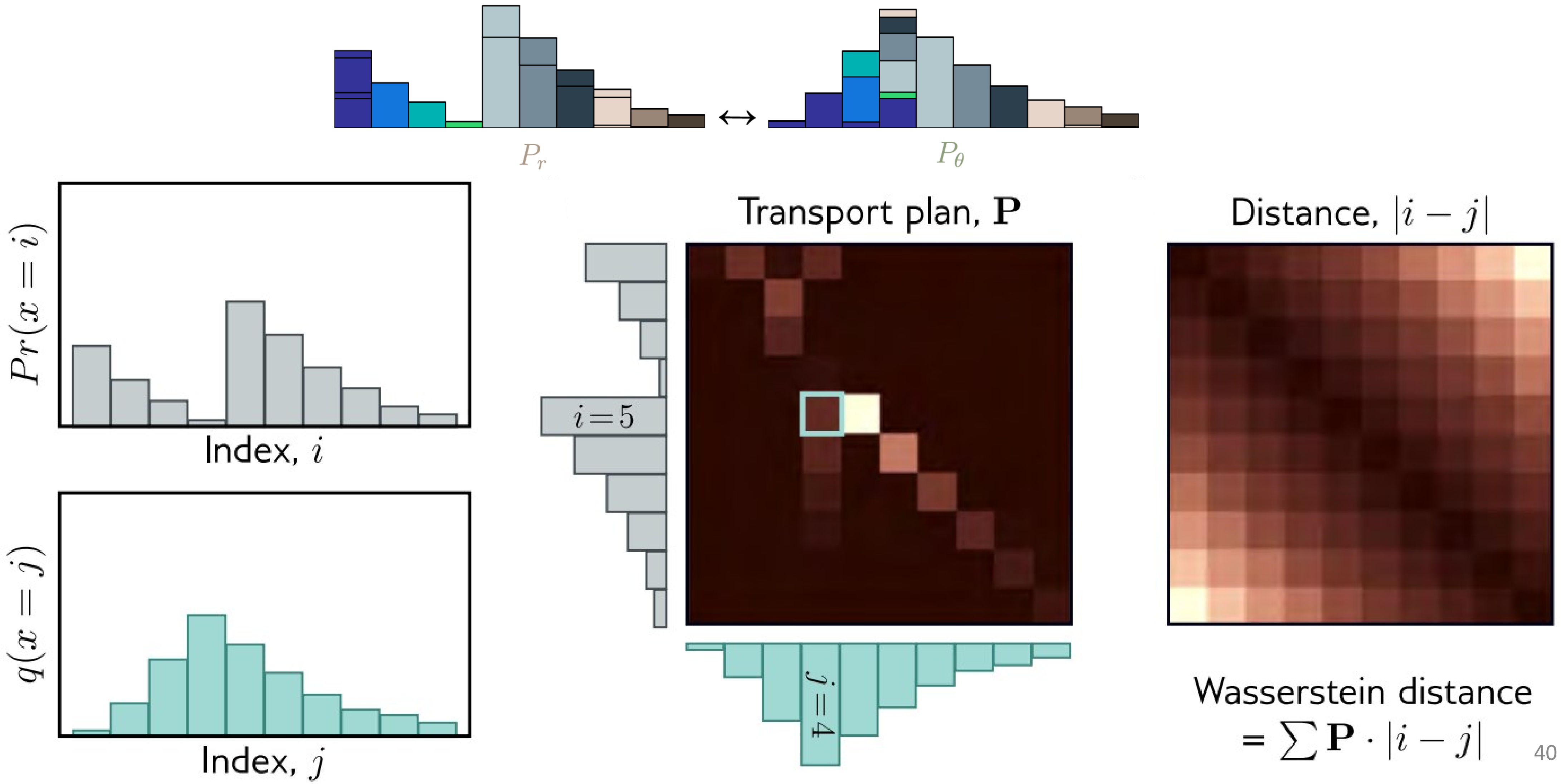
Wasserstein Distance



Wasserstein Distance



Wasserstein Distance



Wasserstein Distance

- Para obter o valor da distância de *Wasserstein* precisamos resolver o problema de otimização linear para encontrar a matriz P
- Contudo para esse problema específico, a formalização dual é mais interessante

Wasserstein Distance

- Reescrevendo em sua forma dual, temos uma nova variável de otimização f que deve ser maximizada sob a restrição que os valores adjacentes não podem ser modificados em mais do que uma unidade

$$D_w[Pr(x)||Qr(x)] = \max_f \left[\sum_i \mathrm{Pr}(x = 1)\, f_i - \sum_j \mathrm{Qr}(x = 1)\, f_j \right]$$

Sob a restrição: $|f_{i+1} - f_i| < 1$

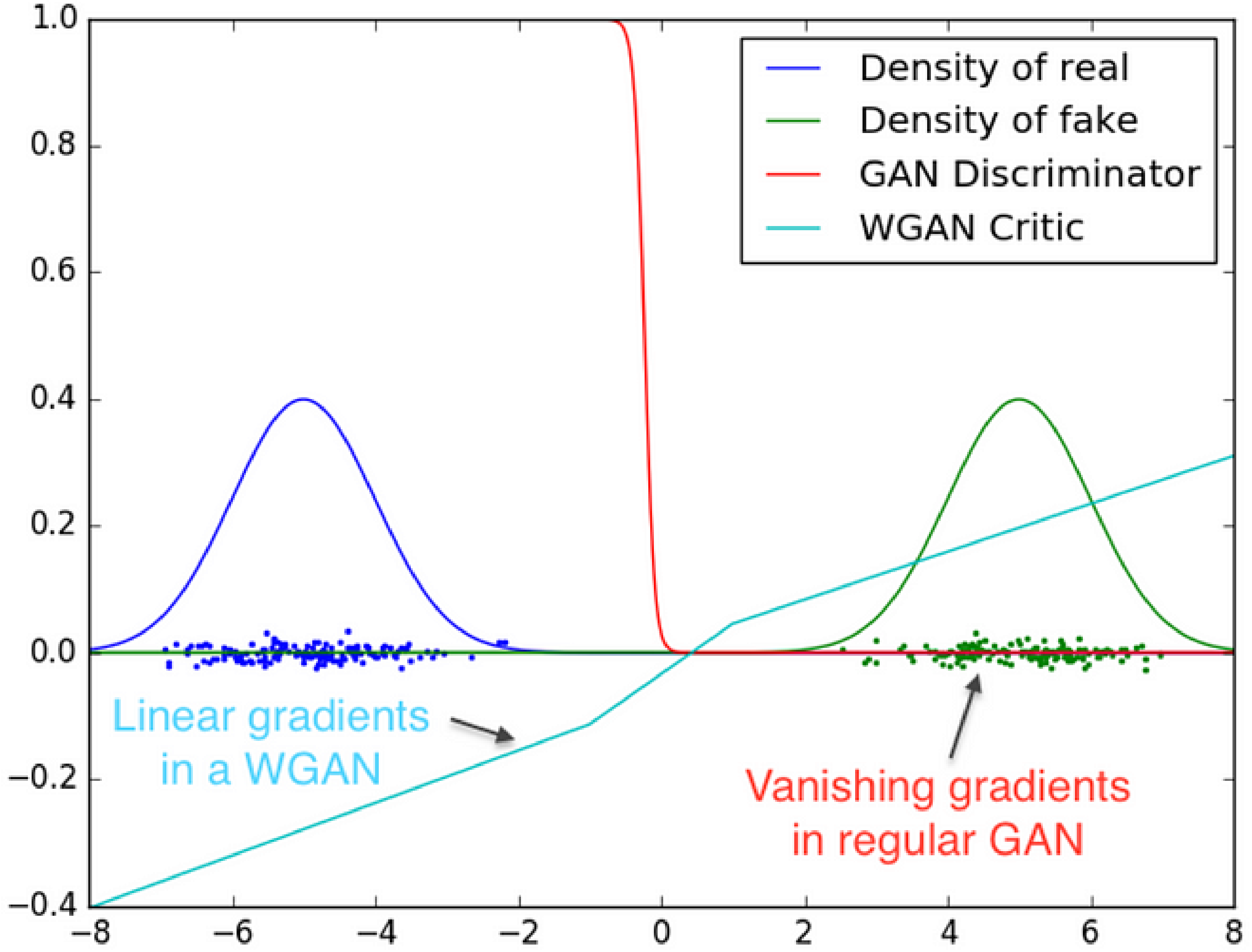
Wasserstein Distance

- Considerando agora que as distribuições são aproximadas pelas instâncias reais do treinamento e as instâncias geradas temos a loss

$$L[\phi, \theta] = \sum_j f\left[g\left[z^{(j)}, \theta\right], \phi\right] - \sum_i f\left[x^{(i)}, \phi\right]$$

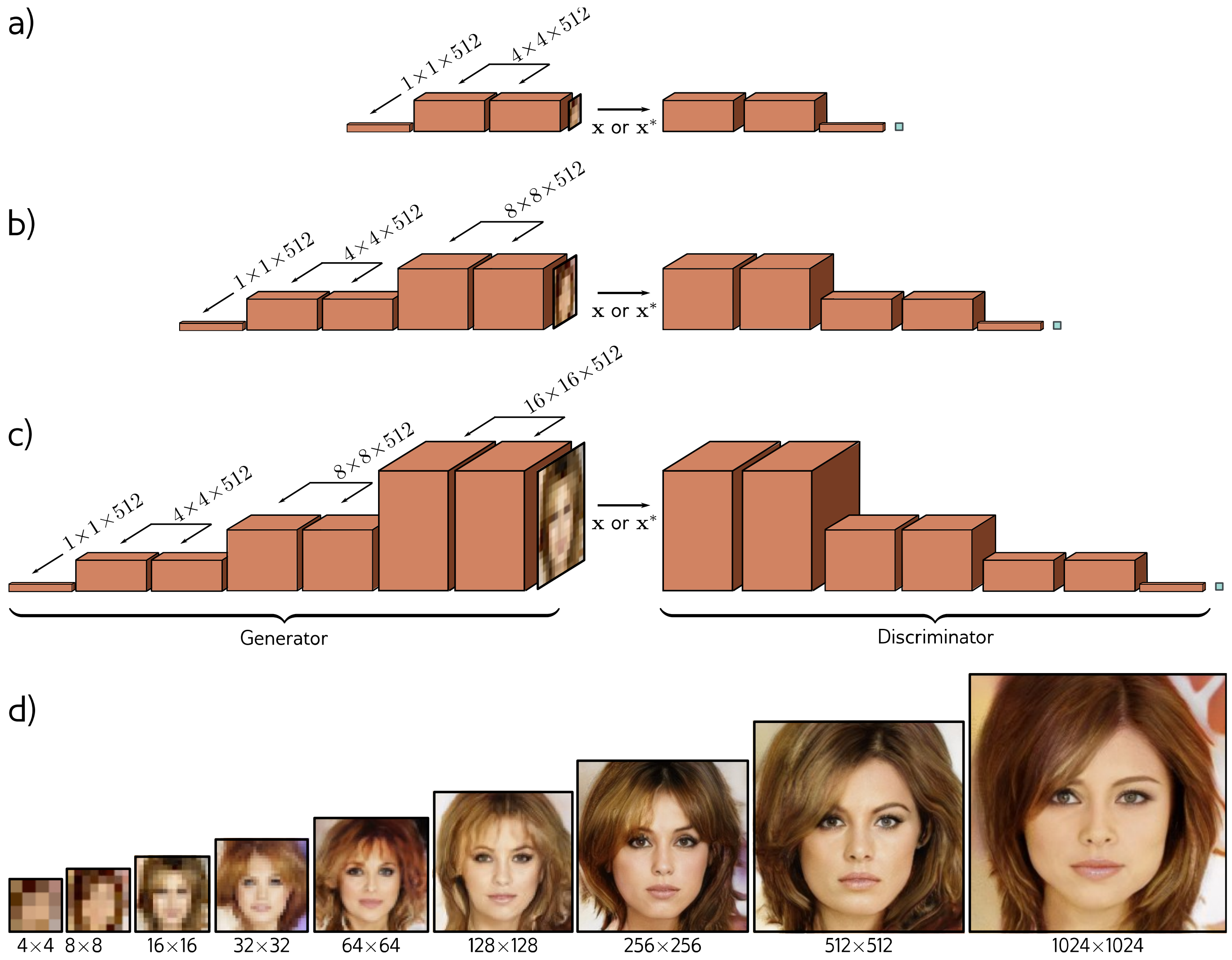
Sob a restrição: $\left|\frac{\partial f(x,\phi)}{\partial x}\right| < 1$

Wasserstein Distance



https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Trick 1: Progressive growing



Trick 2: Minibatch discrimination

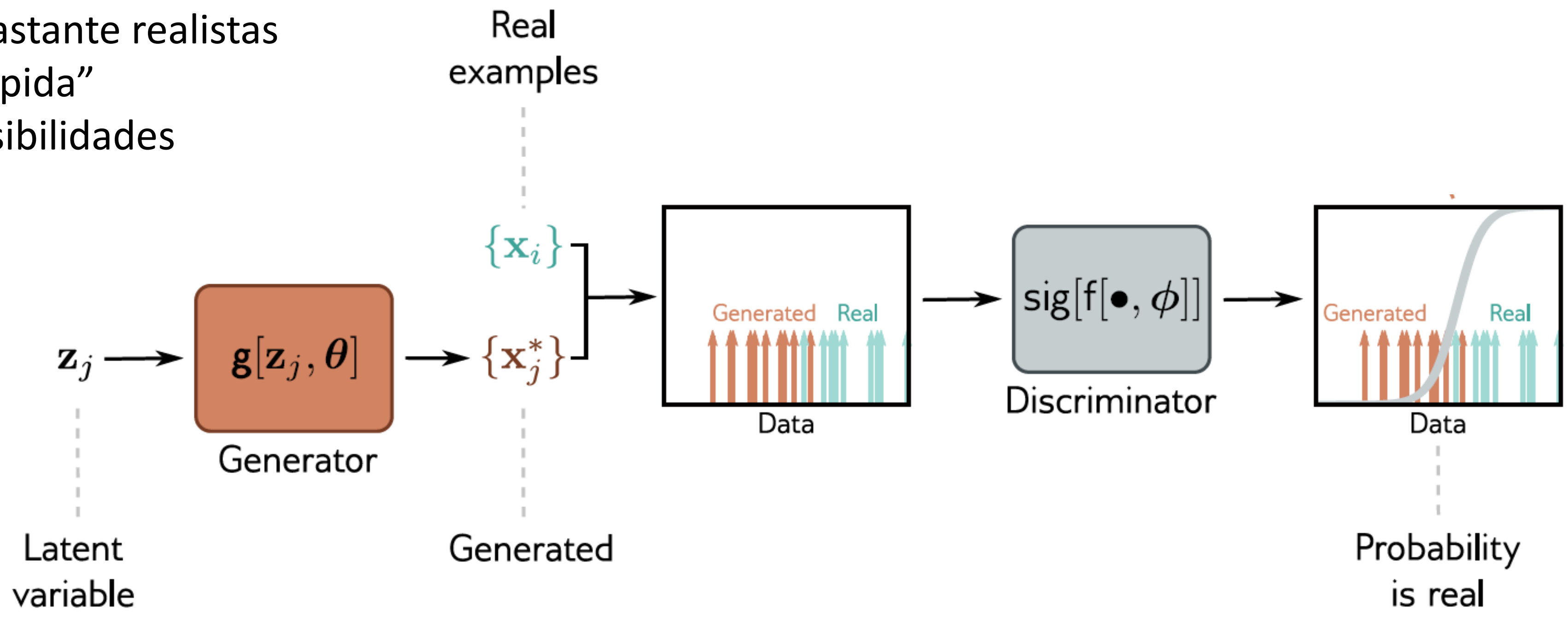
- Add in statistics across minibatch as input to discriminator
- Forces generated batch to have similar variation to real batch

Trick 3: Truncation

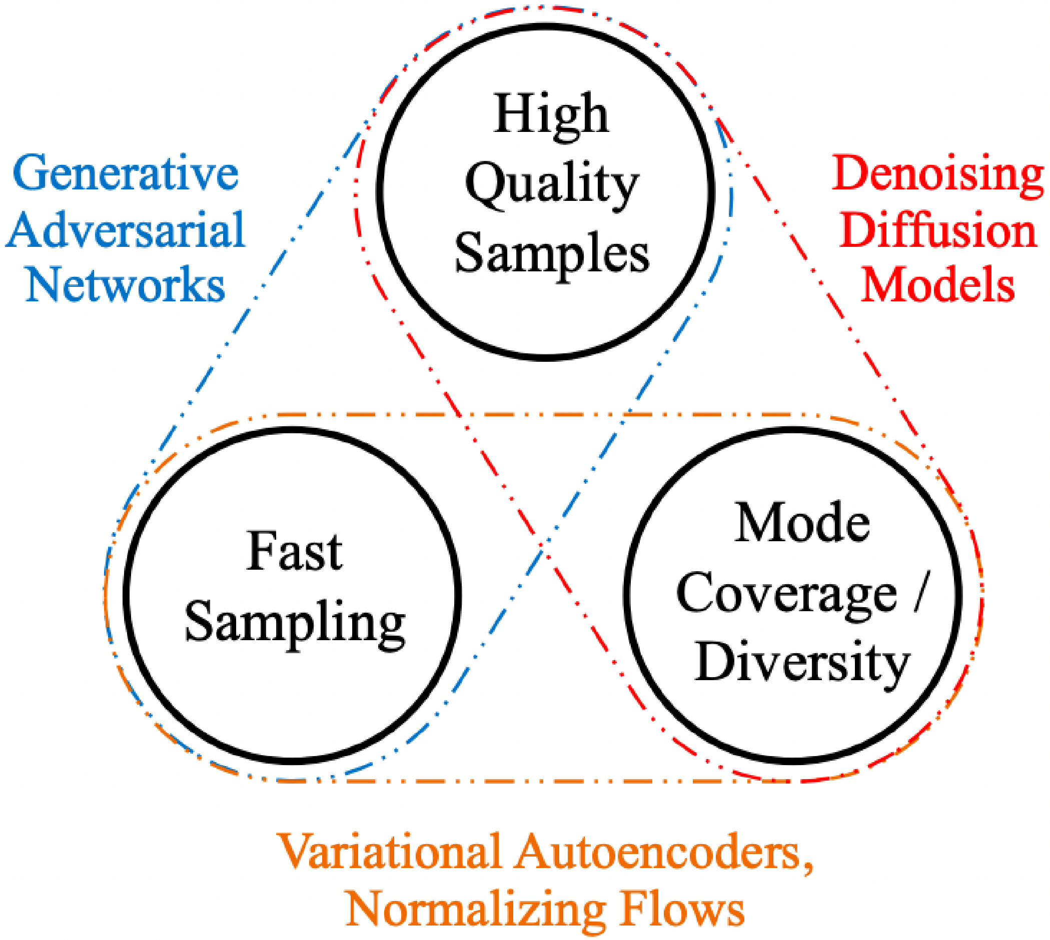
Only choose random values of latent variables that are less than a threshold τ.

GAN

- Existem diversas arquiteturas de GANs que permitem algum controle sobre as imagens geradas
  - GANs condicionadas
  - StyleGAN
- TLDR sobre GANS
  - Difíceis de treinar
  - Produzem resultados bastante realistas
  - Possuem inferência “Rápida”
  - Baixa cobertura de possibilidades



O trilema dos modelos geradores



Referências

- https://vincentherrmann.github.io/blog/wasserstein/
- Capítulo 5 – Understanding Deep Learning – Simon J. Prince