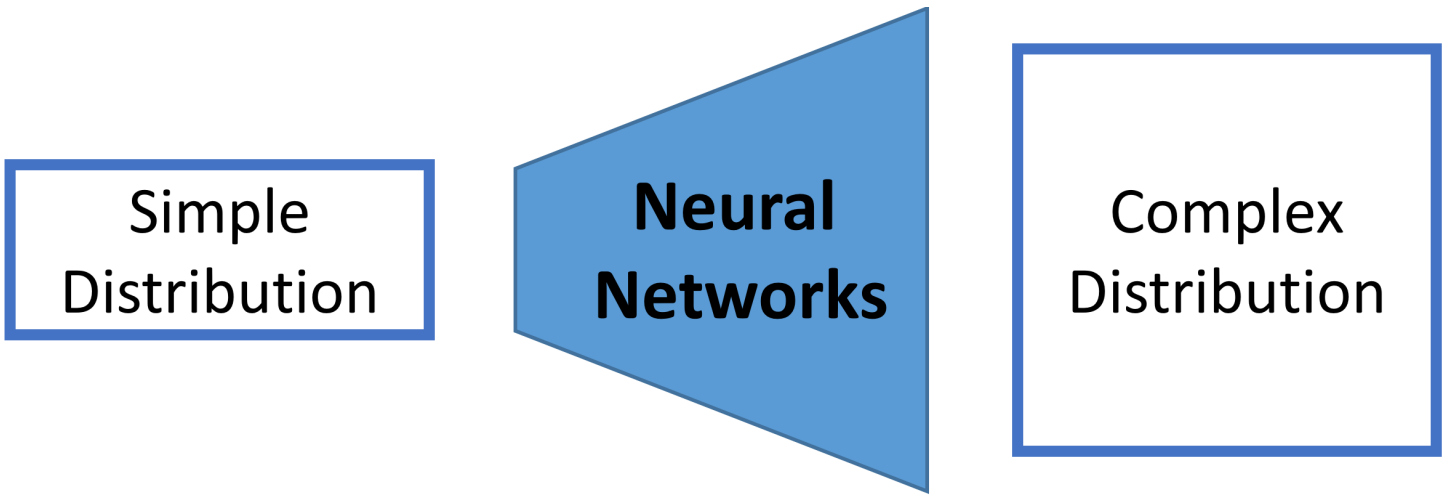
Nonconvex Optimization in Deep Learning: Generative Adversarial Networks (GAN)

Chee Wei Tan

Thanks to Prof. Xudong Mao for most of the materials in this lecture

Deep Generative Models

- Sample from a simple distribution (e.g., Gaussian distribution).
- Use neural networks to learn the transformation (from simple distribution to complex distribution).



Recent Deep Generative Models

- Variational Autoencoder (VAE)
  - (Kingma and Welling 2013)
  - Explicit density
- Flow-based Generative Model
  - (Dinh et al. 2014)
  - Explicit density
- **Generative Adversarial Network (GAN)**
  - **(Goodfellow et al. 2014)**
  - **Implicit density**
- Diffusion-based Generative Model
  - (Sohl-Dickstein et al. 2015)
  - Explicit density

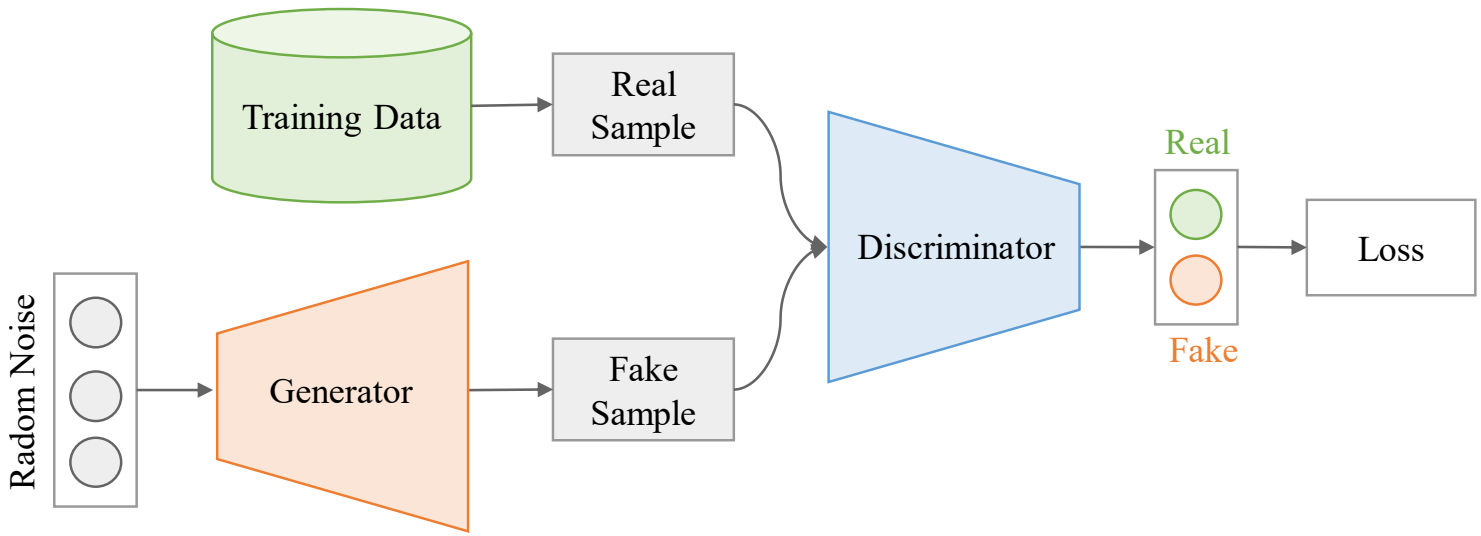
Generative Adversarial Networks (GANs)

- Optimization problem:

$$\min_G \max_D V_{\text{GAN}}(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{d}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \tag{1}$$

where G is the generator, D is the discriminator, $\boldsymbol{x}$ is a real sample, $\boldsymbol{z}$ is a noise vector, p_{d} is the real data distribution, and $p_{\boldsymbol{z}}$ is the noise distribution

- D uses binary cross-entropy loss to classify "1" for real and "0" for fake



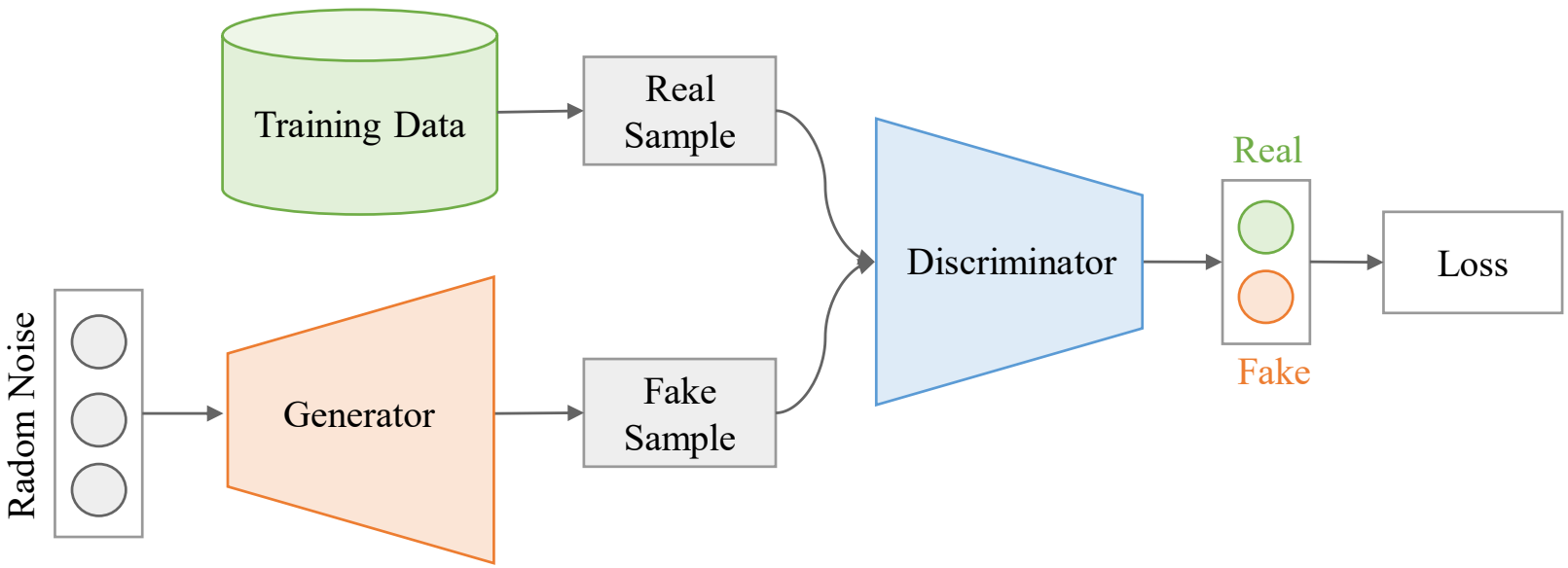
Generative Adversarial Networks (GANs)

- Minimax loss (M-GAN):

$$\min_G \max_D V_{\text{GAN}}(D, G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{d}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2}$$

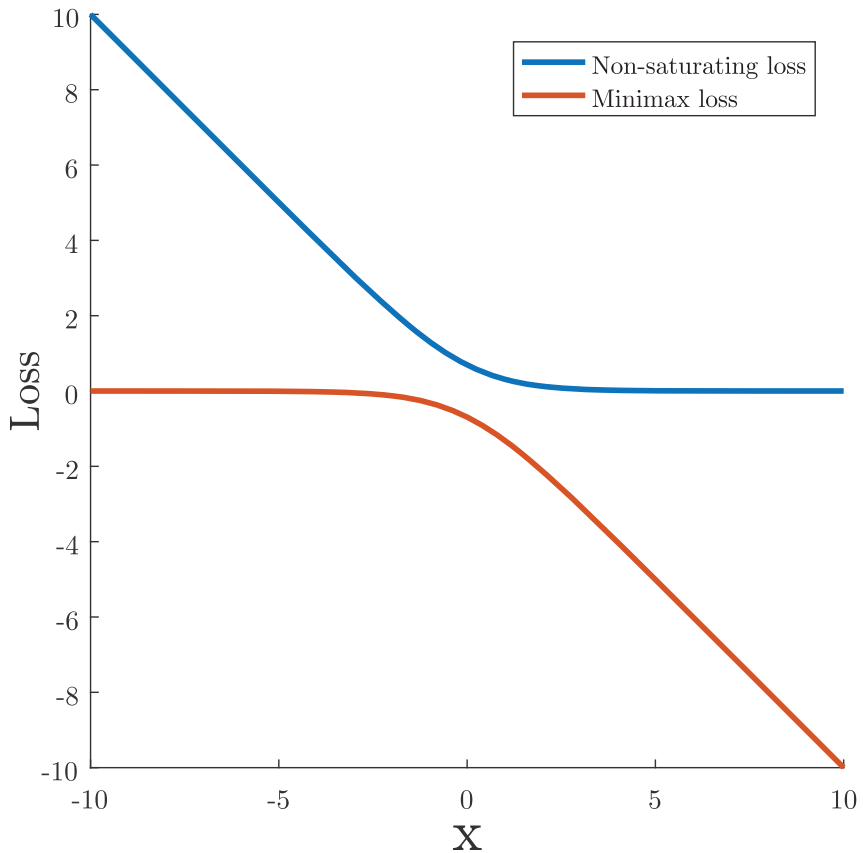
- In practice, M-GAN saturates
- Non-Saturating loss (NS-GAN) for G provides stronger gradients:

$$\min_G V_{\text{GAN}}(G) = -\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))] \tag{3}$$



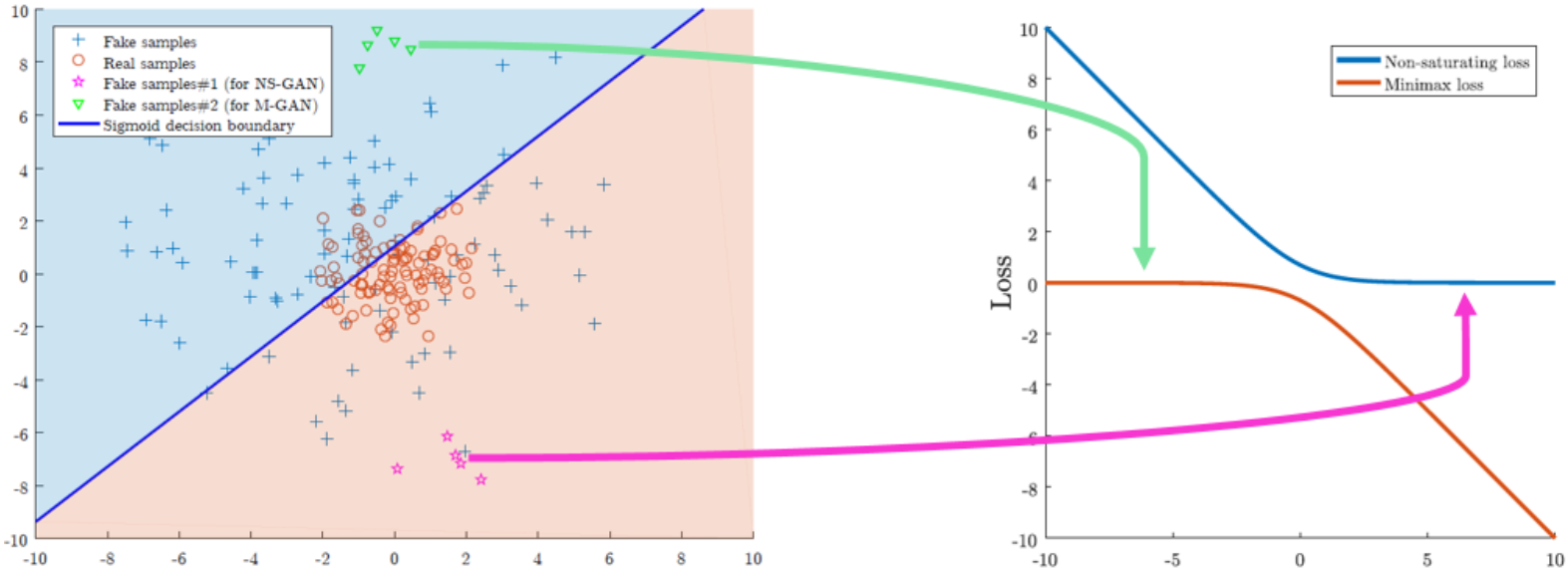
Problem of Cross-Entropy Loss

- The non-saturating loss will saturate when the input is relatively large.
- The minimax loss will saturate when the input is relatively small.



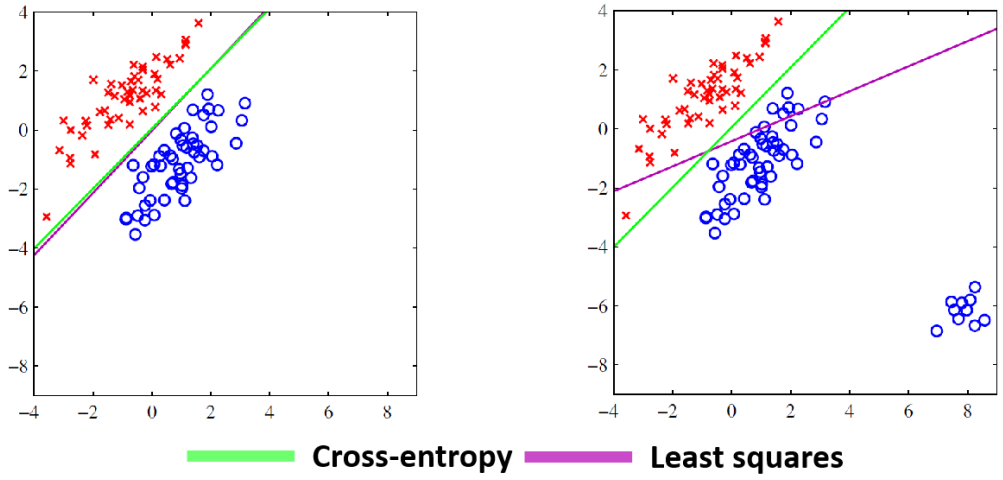
Problem of Cross-Entropy Loss

- Consequently (when updating the generator):
- The non-saturating loss will cause almost no gradient for the fake samples in pink
- The minimax loss will cause almost no gradient for the fake samples in green

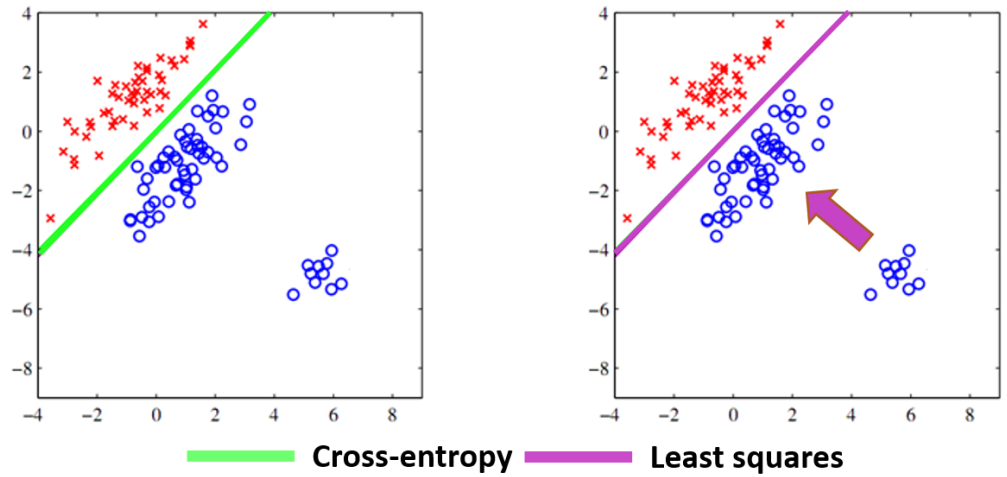


Intuition of Using Least Squares Loss

- Problem of using least squares for classification:

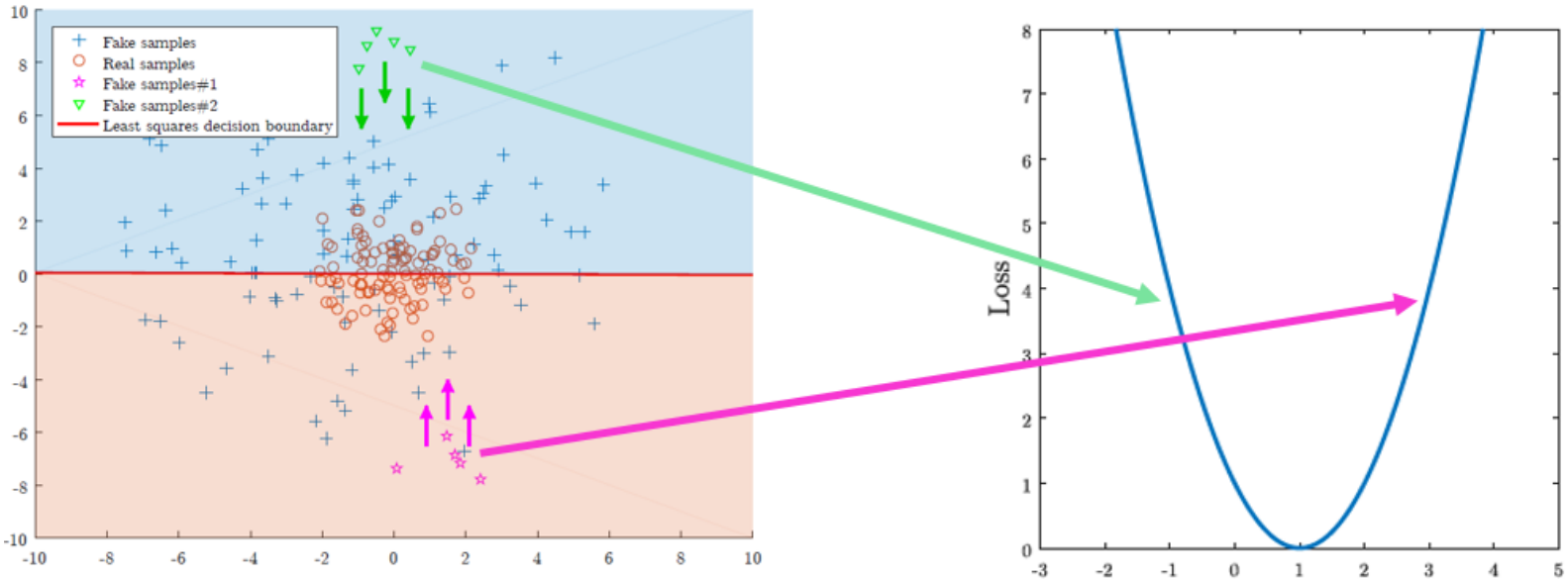


- When updating G, the decision boundary (D) is fixed:



Intuition of Using Least Squares Loss

- The least squares loss pulls the fake samples toward the decision boundary:



Least Squares GANs (LSGANs)

- The objective of LSGANs:

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{d}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right] \\ \min_G V_{\text{LSGAN}}(G) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right]\end{aligned} \tag{4}$$

  where a is for real, b is for fake, and c is for G.

- Theoretical analysis:
  LSGANs minimizes the Pearson χ^2 divergence between $p_{\text{d}} + p_g$ and $2p_g$, if a, b, and c satisfy the conditions of $b - c = 1$ and $b - a = 2$ in Equation (4).

Selection of Parameters

- Following the conditions ($b - c = 1$ and $b - a = 2$) in theoretical analysis by setting $a = -1$, $b = 1$, and $c = 0$ (denoted as LSGANs$_{-110}$):

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{d}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - 1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) + 1)^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right]$$

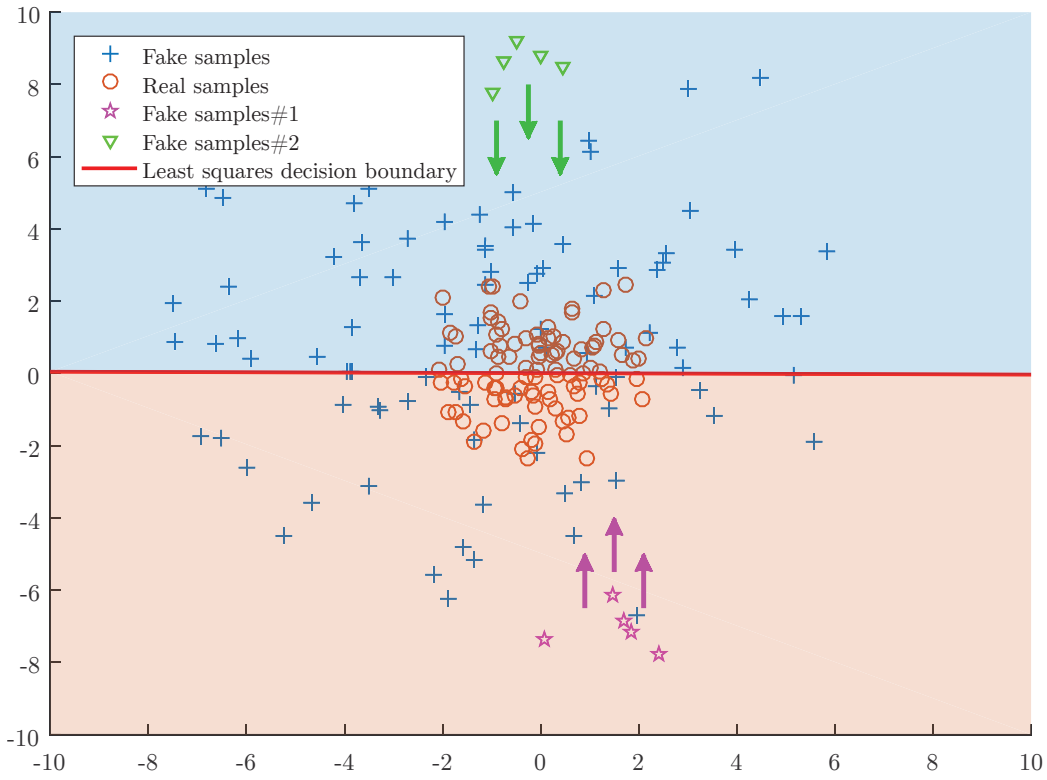
- Following the traditional way (i.e., 0-1 binary coding scheme) of using least squares for classification (denoted as LSGANs$_{011}$):

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{d}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - 1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - 1)^2\right]$$

Improved Image Quality

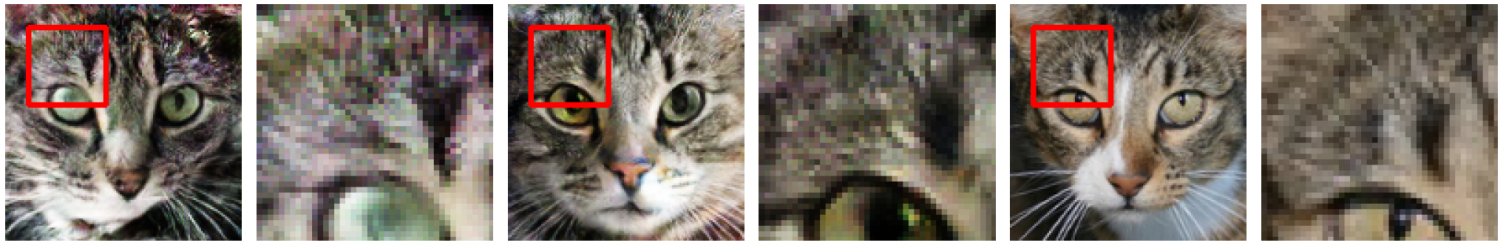
- LSGAN pulls the fake samples toward the decision boundary.
- On the other hand, the decision boundary should go across the manifold of real data for a successful GANs learning.
- Thus LSGAN generates samples closer to the manifold of real data.



Improved Image Quality

(a) Generated cats (128 × 128) by NS-GANs.

(b) Generated cats (128 × 128) by LSGANs.



(c) NS-GANs (d) LSGANs (e) Real Sample

Improved Image Quality

(a) Church outdoor.

(b) Dining room.

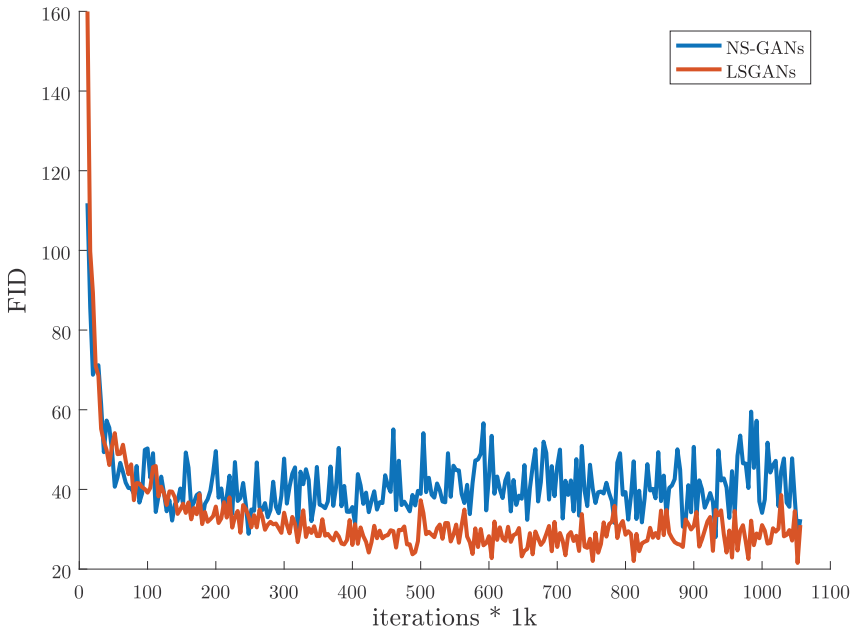
(c) Kitchen.

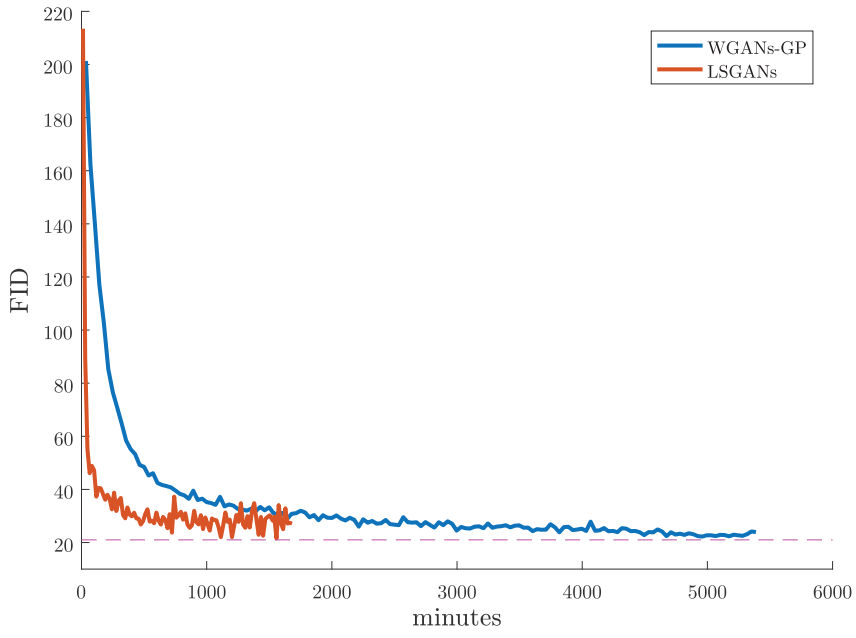
(d) Conference room.

Improved Image Quality

- FID comparison:

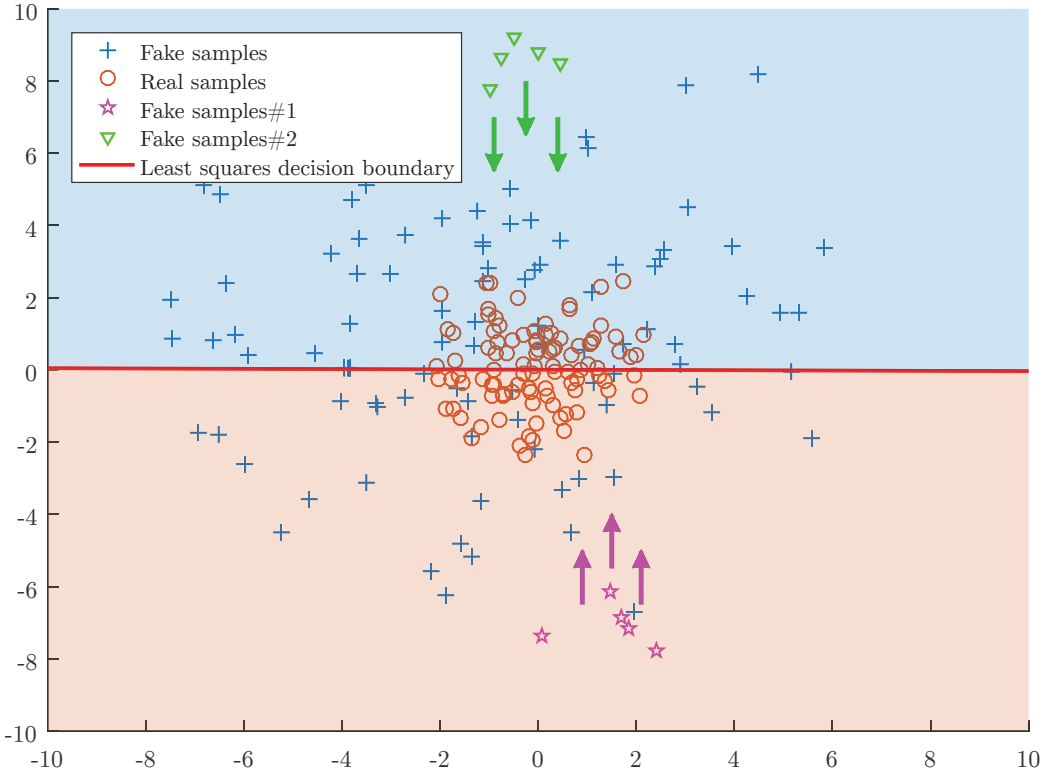
Method	LSUN	Cat	ImageNet	CIFAR10
NS-GANs	28.04	15.81	74.15	35.25
WGANs-GP	22.77	29.03	**62.05**	40.83
$\text{LSGANs}_{(011)}$	27.21	15.46	72.54	36.46
$\text{LSGANs}_{(-110)}$	**21.55**	**14.28**	68.95	**35.19**





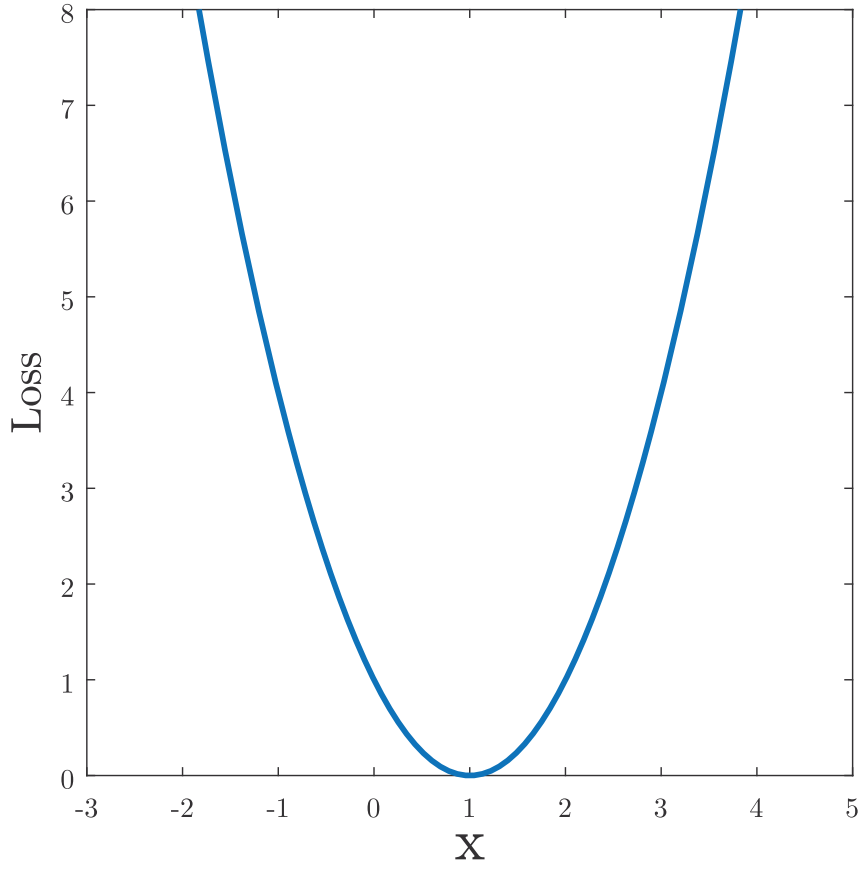
Improved Training Stability

- Penalizing the samples lying in a long way to the decision boundary provides stronger gradients



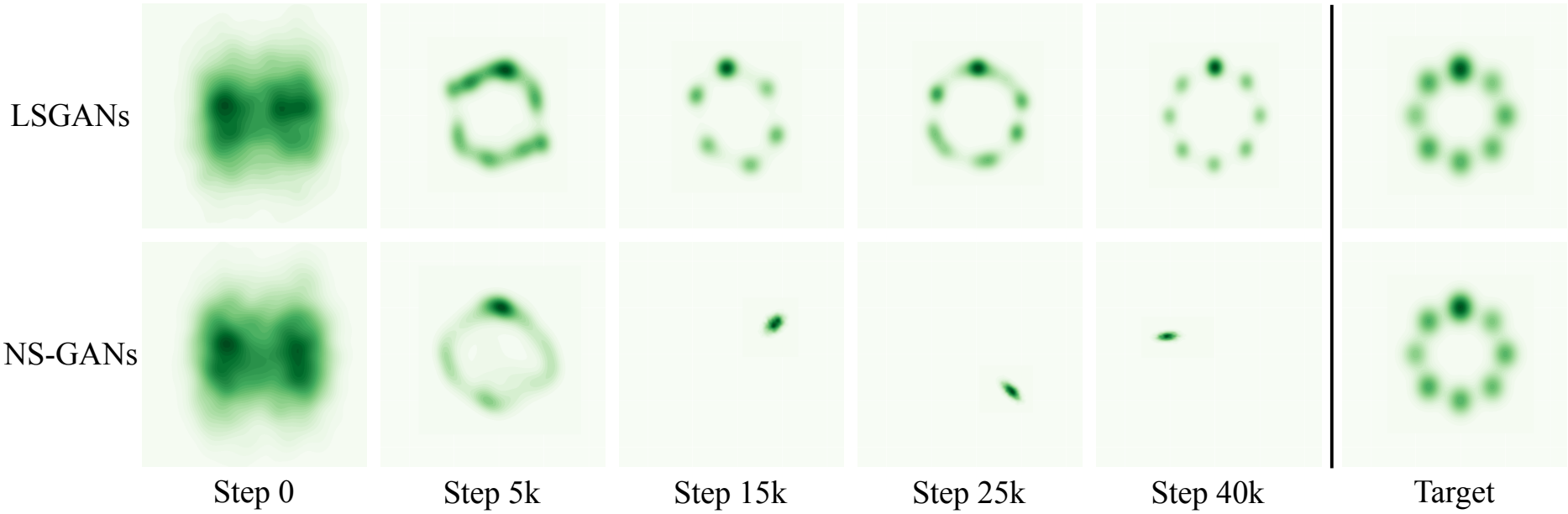
Improved Training Stability

- Penalizing the samples lying in a long way to the decision boundary provides stronger gradients.
- The least squares loss function is flat only at one point.



Improved Training Stability

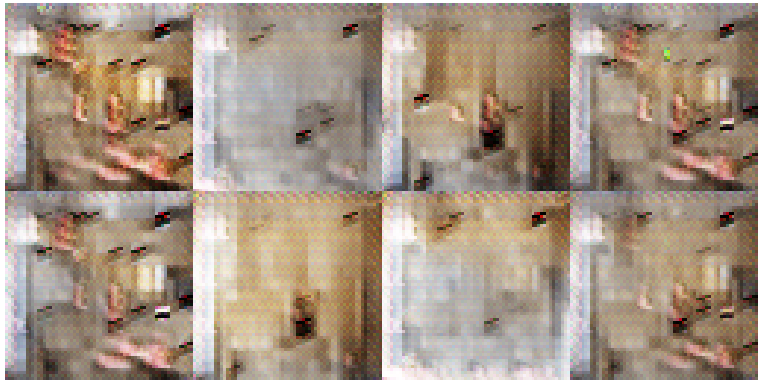
- Gaussian kernel estimation:



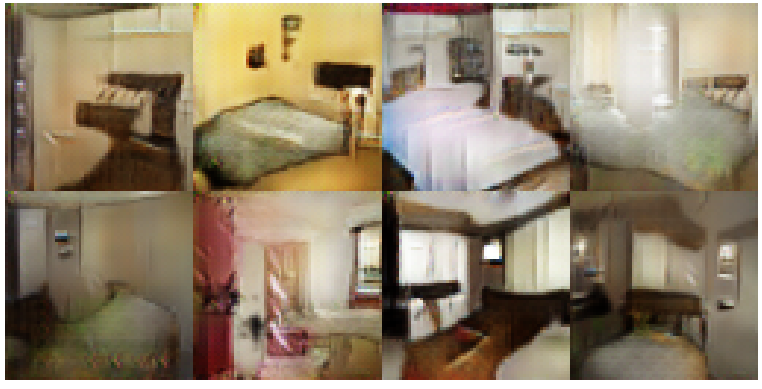
Improved Training Stability

- Exclude batch normalization (BN) in G or D:

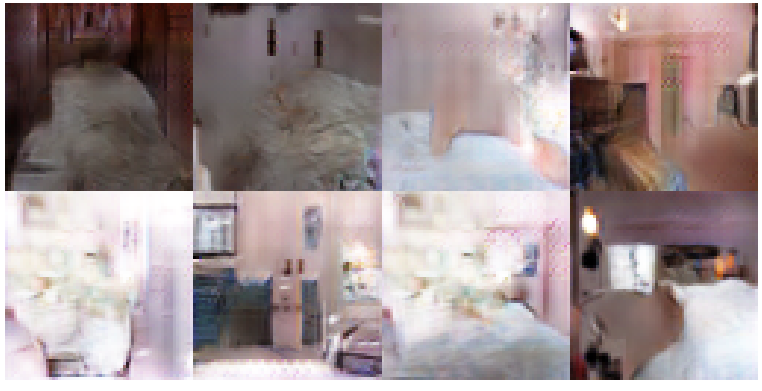
NS-GANs

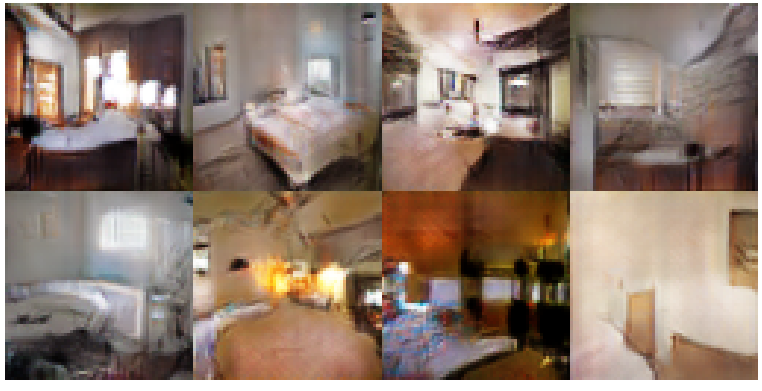


LSGANs



(a) No BN in G.



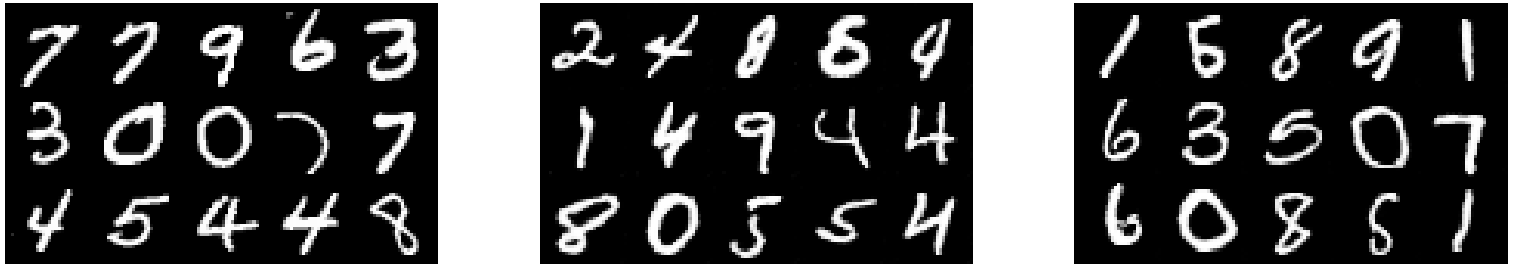


(b) No BN in either G or D.

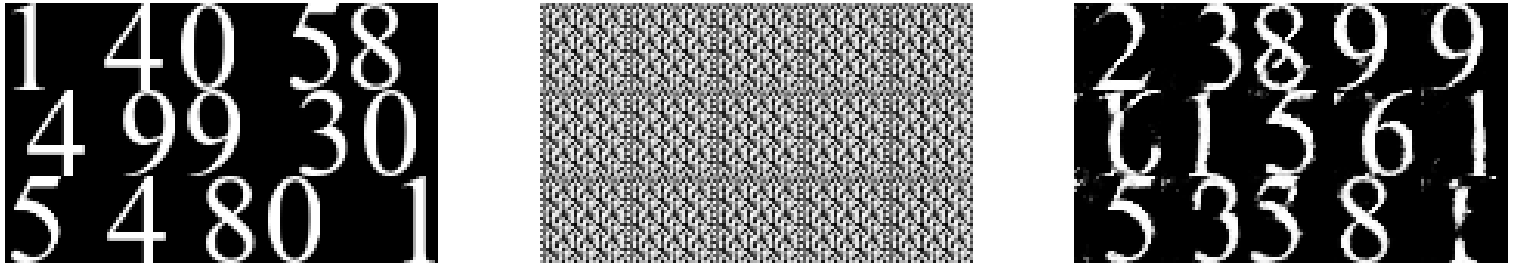
Improved Training Stability

- Datasets with Small Variability:

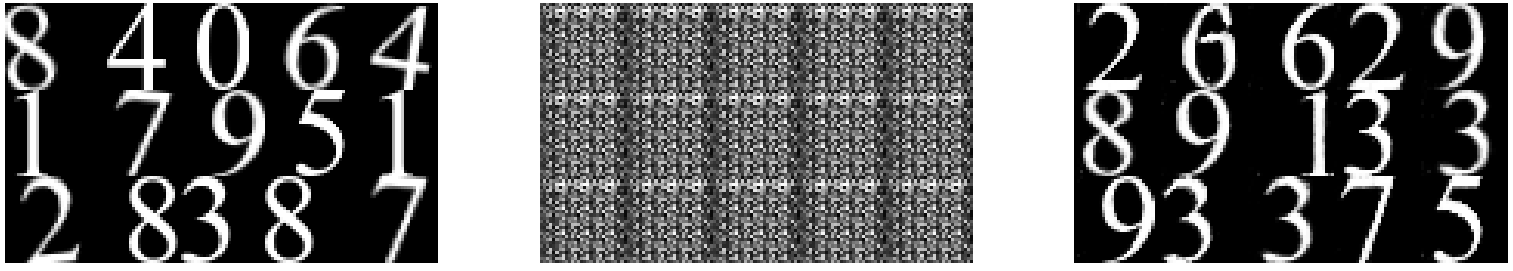
Real Samples | NS-GANs | LSGANs



(a) MNIST.



(b) A synthetic digit dataset with a random horizontal shift.



(c) A synthetic digit dataset with random horizontal shift and rotation.

Minimax Optimization of f-divergence

- Closeness will be measured by some divergence function $D(\cdot, \cdot)$ between probability distributions. So the generator is typically solving

$$P_g^* = \arg\inf_{P_g} D\left(P_g, P_r\right)$$

  where P_g is in some class of distributions specified by the generator and $D(\cdot, \cdot)$ is an f-divergence:
- f-divergence: For any convex $f(t)$ with $f(1) = 0$, define the f-divergence of P_g to P_r as

$$D_f\left(P_g, P_r\right) = \mathbb{E}_{x \sim P_r}\left[f\left(\frac{P_g(x)}{P_r(x)}\right)\right]$$

- Examine differences between P_r and P_g by feeding them to a binary classifier (discriminator) with corresponding labels $y = \pm 1$ in equal proportion so that the data input to the discriminator are assigned a positive label if they are real data:

$$p := \Pr(y = +1) = \frac{1}{2}, P_r(x) = \Pr(x \mid y = +1), P_g(x) = \Pr(x \mid y = -1)$$

Minimax Optimization of f-divergence

- Optimize generator to simulate P_g^* to be "close" to P_r under some f-divergence D_f by solving

$$
\begin{aligned}
P_g^* &= \arg\inf_{P_g} D_f\left(P_g, P_r\right) = \arg\sup_{P_g}\left[-D_f\left(P_g, P_r\right)\right] \\
&= \operatorname{argsup}_{\Pr(x|y=-1)}\left[\inf_{h\in\mathcal{H}} \mathbb{E}_{(x,y)}\left[\ell(y, h(x))\right] - \epsilon(\mathcal{H})\right] \\
&\approx \arg\sup_{\Pr(x|y=-1)}\left[\inf_{h\in\mathcal{H}} \mathbb{E}_{(x,y)}\left[\ell(y, h(x))\right]\right]
\end{aligned}
$$

- Minimax optimization decomposition as an adversarial game interaction between the generator and powerful enough discriminators, for any $\ell, \mathcal{H}$.

Minimax Optimization of f-divergence

Loss ℓ	Partial losses	$f(s)$	$h^*(s)$	f-divergence
$0-1$	$\ell_\pm(g) = \frac{1}{2}(1 \mp g)$	$\frac{1}{2}\lvert s-1\rvert$	$\operatorname{sgn}(s-1)$	Total variation dist.
log	$\ell_\pm(g) = \ln\left(\frac{2}{1\pm g}\right)$	$-\ln(1+s)-$ $s\ln\left(\frac{1+s}{s}\right)$	$\frac{1-s}{1+s}$	Jensen-Shannon dist.
Square	$\ell_\pm(g) = (1 \mp g)^2$	$-\frac{s}{1+s}+\frac{1}{2}$	$\frac{1-s}{1+s}$	Triangular discrimination dist.
CW (param. c)	$\ell_\pm(g) = \left(\frac{1}{2}-\left(\frac{1}{2}-c\right)\right)(1 \mp g)$	$\lvert 1-c-cs\rvert-$ $cs+c-\lvert 1-2c\rvert$	$\operatorname{sgn}(1-c-cs)$	–
Exponential	$\ell_\pm(g) = \exp(\mp g)$	$-2\sqrt{s}+2$	$-\frac{1}{2}\ln s$	Hellinger dist.
"Boosting"	$\ell_\pm(g) = \sqrt{\frac{1\mp g}{1\pm g}}$	$-2\sqrt{s}+2$	$\frac{1-s}{1+s}$	Hellinger dist.

Summary

- Unsupervised learning of generative models: Minimax algorithm by alternating optimization techniques
- f-divergence between probability distributions of generated data and real data: GAN, LSGAN, f-GAN, CycleGAN, GAN Zoo
- Many interesting nonconvex optimization problems in deep learning

References:

- *Generative Adversarial Networks*, I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, Advances in Neural Information Processing Systems 2014.
- *Deep Learning*, I. J. Goodfellow, Y. Bengio and A. Courville, MIT Press 2016.
- *Least Squares Generative Adversarial Networks*, X. Mao, Q. Li, H. Xie, R. Lau, Z. Wang, S. Paul Smolley, Computer Vision and Pattern Recognition 2016.