Generative Modeling with (W)GAN

Sean Welleck

NYU

wellecks@nyu.edu

April 2018

Overview

Generative Modeling

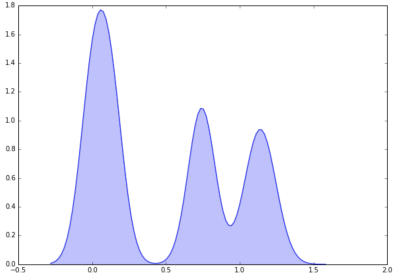
Generative Modeling

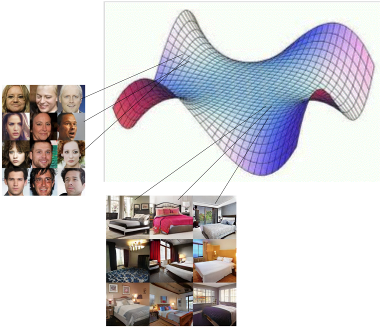
Model a probability distribution $\mathbb{P}_r$ over a domain $\mathcal{X}$.

- $\mathbb{P}_r$ typically high-dimensional
- Model's distribution: $\mathbb{P}_\theta$

Generative Modeling

Model a probability distribution $\mathbb{P}_r$ over a domain $\mathcal{X}$. Example $\mathbb{P}_r$'s:





Potential Uses:

1. **Sampling**: generate samples $\hat{x}_1, \hat{x}_2, ... \sim \mathbb{P}_\theta$ that 'resemble' samples from $\mathbb{P}_r$.
2. **Estimation**: given $x_1, ..., x_n \sim \mathbb{P}_r$, find density p_θ that best describes $\mathbb{P}_r$.
3. **Likelihood Evaluation**: given $x \in \mathcal{X}$, evaluate its likelihood $p_\theta(x)$.

(2,3) require learning a density p_θ.
GANs focus on *sampling*, without explicitly learning a density p_θ.

Generative Modeling - MLE

Maximum Likelihood Estimation (MLE)

Given:

1. parametric family of densities $\{p_\theta\}_{\theta \in \Theta}$
2. samples $x_1, \ldots, x_N$ from real data distribution $\mathbb{P}_R$

Solve:

$$\arg\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x_i)$$

Equivalent to:

$$\arg\min_{\theta \in \Theta} D_{KL}(\mathbb{P}_R || \mathbb{P}_\theta)$$

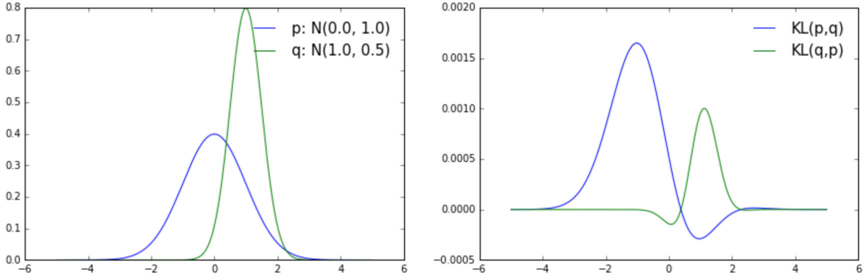
where $\mathbb{P}_\theta$ is the distribution with density p_θ.

KL Divergence

KL Divergence

$$D_{KL}(p||q) = \int_{\mathcal{X}} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx$$

```
KL(p,q)=0.280
KL(q,p)=0.082
```



Generative Modeling - Implicit

Alternative: Only learn to generate samples (*implicit* density)

Implicit Generative Modeling

Consider real data distribution $\mathbb{P}_R$ over domain $\mathcal{X}$. Define:

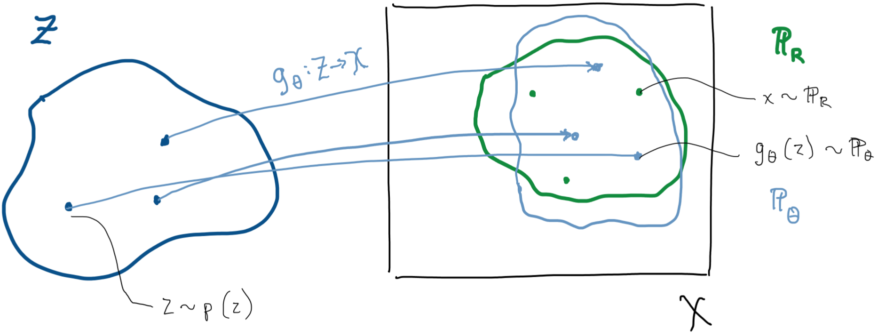
1. Random variable Z with fixed distribution $p(z)$ over $\mathcal{Z}$ (e.g. uniform).
2. Function $g_\theta : \mathcal{Z} \to \mathcal{X}$ (e.g. neural network)

Then samples $g_\theta(z)$ follow a distribution $\mathbb{P}_\theta$.
Problem: Choose θ such that $\mathbb{P}_\theta$ is *close to* $\mathbb{P}_R$.

This Lecture:

1. **Learning** g_θ: Adversarial Training (GAN)
2. **"Close To"**: D_{KL}, D_{JS}, Wasserstein,...?

Generative Modeling - Implicit



Choose θ such that $\mathbb{P}_\theta$ is *close to* $\mathbb{P}_R$

Generative Adversarial Networks (GAN)

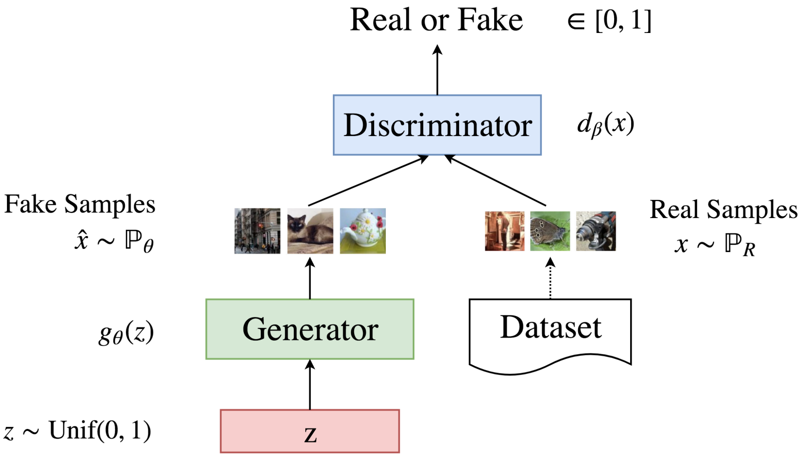
[Goodfellow et al. 2014]

GAN — Key Idea

Learn to generate samples through a **2-player game**.
Discriminator $d_\beta : \mathcal{X} \to [0,1]$: learn to distinguish between real and fake samples.
Generator $g_\theta : \mathcal{Z} \to \mathcal{X}$: learn to fool the discriminator.

1. If g_θ fools the discriminator, then $\mathbb{P}_\theta \approx \mathbb{P}_R$!
2. Easy to sample; evaluate $g_\theta(z)$ where $z \sim \text{Unif}(0,1)$

GAN — Key Idea



Real or Fake $\in [0, 1]$
Discriminator
$d_\beta(x)$
Fake Samples
$\hat{x} \sim \mathbb{P}_\theta$
Real Samples
$x \sim \mathbb{P}_R$
$g_\theta(z)$
Generator
Dataset
$z \sim \text{Unif}(0, 1)$
z

GAN — Objective

$$\min_{\theta} \max_{\beta} \mathbb{E}_{x \sim \mathbb{P}_r} [\log d_\beta(x)] + \mathbb{E}_{x \sim \mathbb{P}_\theta} [\log(1 - d_\beta(x))]$$

d_β: discriminator

g_θ: generator

GAN — Maximization

$$\min_{\theta} \max_{\beta} \underbrace{\mathbb{E}_{x\sim\mathbb{P}_r}[\log d_\beta(x)]}_{d(x)\to 1 \text{ for real samples}} + \underbrace{\mathbb{E}_{x\sim\mathbb{P}_\theta}[\log(1-d_\beta(x))]}_{d(x)\to 0 \text{ for fake samples}}$$

d_β: discriminator

g_θ: generator

GAN — Minimization

$$\min_{\theta} \max_{\beta} \mathbb{E}_{x\sim\mathbb{P}_r}\left[\log d_\beta(x)\right] + \underbrace{\mathbb{E}_{x\sim\mathbb{P}_\theta}\left[\log(1 - d_\beta(x))\right]}_{\text{Make } \mathbb{P}_\theta \to \mathbb{P}_r \text{ so that } d(x) \to 1}$$

d_β: discriminator

g_θ: generator

GAN — Equivalent Differentiable Objective

$$\min_{\theta} \max_{\beta} \mathbb{E}_{x \sim \mathbb{P}_r} [\log d_\beta(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - d_\beta(g_\theta(z)))]$$

$p(z)$: noise distribution, e.g. $U(0, 1)$.
d_β: discriminator
g_θ: generator

GAN — Training

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.

GAN — Properties

1. Optimal Discriminator
2. JS-Divergence

GAN — Properties - Optimal Discriminator

Proposition (Optimal Discriminator)

For a fixed generator G

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_G(x)}$$

GAN — Properties - Optimal Discriminator

Optimal Discriminator.

For a fixed generator G,

$$\begin{aligned} D^* &= \arg\max_D \mathcal{L}(G, D) \\ &= \arg\max_D \int_{\mathcal{X}} \log(D(x))p_r(x)dx + \int_{\mathcal{Z}} \log(1 - D(G(z)))p_z(z)dz \\ &= \arg\max_D \int_{\mathcal{X}} [\log(D(x))p_r(x)dx + \log(1 - D(x))p_G(x)dx] \end{aligned}$$

Observe that $A \log y + B \log(1 - y)$ is maximized at $y = \frac{A}{A+B}$. Thus $D^*(x) = \frac{p_r(x)}{p_r(x)+p_G(x)}$. □

GAN — Properties - JS-Divergence

Given an optimal discriminator, the generator optimization is related to minimizing a Jensen-Shannon divergence.

JS Divergence

KL Divergence

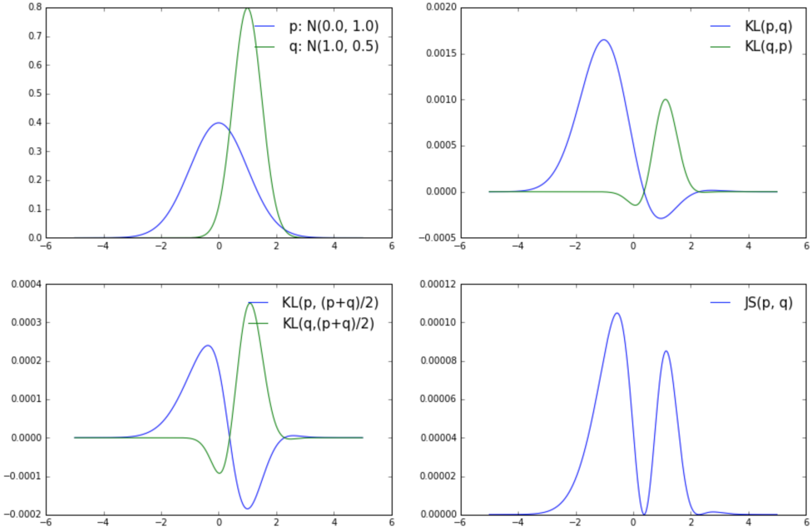
$$D_{KL}(p||q) = \int_{\mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

Jensen-Shannon (JS) Divergence

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

JS Divergence

```
KL(p,q)=0.280
KL(q,p)=0.082
JS(p,q)=JS(q,p)=0.022
```



GAN — Properties - JS-Divergence

Given optimal discriminator D^*, consider finding an optimal generator:

$$\begin{aligned}
\arg\min_G \mathcal{L}(G, D^*) &= \mathbb{E}_{x\sim\mathbb{P}_r} \log D^*(x) + \mathbb{E}_{x\sim\mathbb{P}_G} \log(1 - D^*(x)) \\
&= \mathbb{E}_{x\sim\mathbb{P}_r} \log \frac{p_r}{p_r + p_G} + \mathbb{E}_{x\sim\mathbb{P}_G} \log \frac{p_G}{p_r + p_G}
\end{aligned}$$

GAN — Properties - JS-Divergence

We can find $\mathcal{L}(G, D*)$ by expanding the Jensen-Shannon divergence:

$$
\begin{aligned}
D_{JS}(\mathbb{P}_r, \mathbb{P}_G) &= \frac{1}{2} D_{KL}\left(\mathbb{P}_r, \frac{\mathbb{P}_r + \mathbb{P}_G}{2}\right) + \frac{1}{2} D_{KL}\left(\mathbb{P}_G, \frac{\mathbb{P}_r + \mathbb{P}_G}{2}\right) \\
&= \frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}_r}\left[\log \frac{2p_r}{p_r + p_G}\right] + \frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}_G}\left[\log \frac{2p_G}{p_r + p_G}\right] \\
2D_{JS}(\mathbb{P}_r, \mathbb{P}_G) &= 2\log 2 + \underbrace{\mathbb{E}_{x\sim\mathbb{P}_r}\left[\log \frac{p_r}{p_r + p_G}\right] + \mathbb{E}_{x\sim\mathbb{P}_G}\left[\log \frac{p_G}{p_r + p_G}\right]}_{\mathcal{L}(G,D*)}
\end{aligned}
$$

Therefore:

JS-Divergence Property

$$
\min_G \mathcal{L}(G, D*) \equiv \min_G 2D_{JS}(\mathbb{P}_r, \mathbb{P}_G) - 2\log 2
$$

GAN — Properties - JS-Divergence

JS-Divergence Property

$$\min_G \mathcal{L}(G, D*) \equiv \min_G 2D_{JS}(\mathbb{P}_r, \mathbb{P}_G) - 2\log 2$$

In practice,

- D^* may not be found due to $k - step$ updates.
- We use parametrized D_θ, G_β.

GAN — Recap

1. Learn discriminator d_β, generator g_θ via 2-player game.
2. Use generator to sample from a distribution $\mathbb{P}_\theta$.
3. Training process approximately minimizes $D_{JS}(\mathbb{P}_r, \mathbb{P}_\theta)$.

GAN — Issues

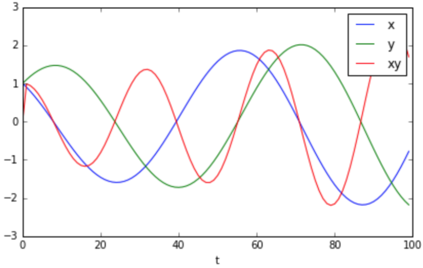
Some issues:

1. Non-convergence
2. Mode collapse
3. JS Divergence (vanishing gradient, instability)

$$J(x, y) = xy$$

Player 1: $\min_x J(x, y)$ — Player 2: $\max_y J(x, y)$

$$x_{t+1} \leftarrow x_t - \eta y_t \qquad y_{t+1} \leftarrow y_t + \eta x_t$$



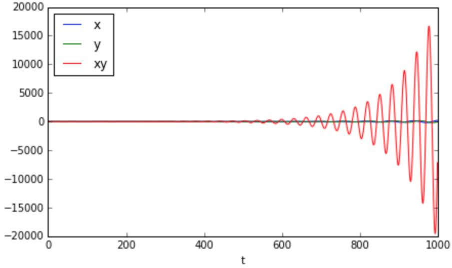


Figure: Difficulty of finding Nash equilibrium using gradient descent.

See [Salimans et al. 2016]

GAN — Issues - Mode Collapse

g_θ maps many z_i to the same x

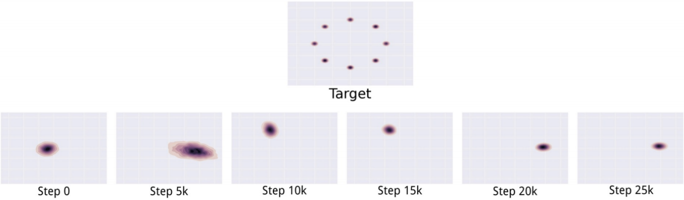
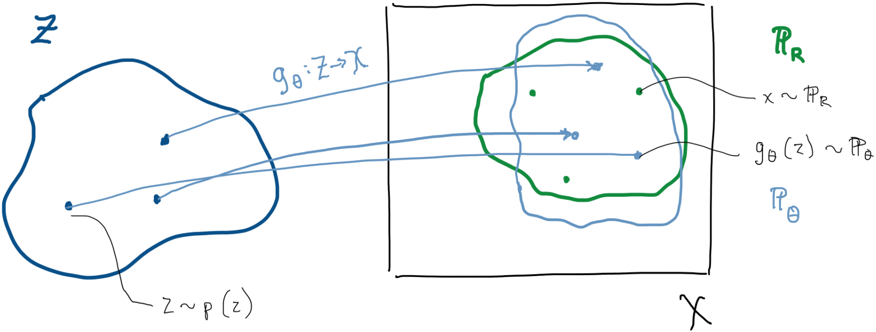


Figure 22: An illustration of the mode collapse problem on a two-dimensional toy dataset. In the top row, we see the target distribution p_{data} that the model should learn. It is a mixture of Gaussians in a two-dimensional space. In the lower row, we see a series of different distributions learned over time as the GAN is trained. Rather than converging to a distribution containing all of the modes in the training set, the generator only ever produces a single mode at a time, cycling between different modes as the discriminator learns to reject each one. Images from Metz *et al.* (2016).

Minibatch Features (Salimans 2016), Unrolled GANs (Metz 2016)



Goal: make $\mathbb{P}_\theta$ *close to* $\mathbb{P}_R$.
What is the proper notion of distance $\rho(\mathbb{P}_\theta, \mathbb{P}_R)$? Is JS-divergence ideal?

GAN — Issues - Choice of distance

Consider using a distance[1] as a loss, $\mathcal{L}(\theta) = \rho(\mathbb{P}_r, \mathbb{P}_\theta)$. Want:

- $\rho(\mathbb{P}_r, \mathbb{P}_\theta)$ continuous
- $\nabla_\theta \rho(\mathbb{P}_r, \mathbb{P}_\theta)$ 'useful'
  - $\nabla_\theta \rho$ exists
  - $\nabla_\theta \rho$ is not everywhere zero

In the typical setting, D_{JS} does not have these properties!
Side-effects: vanishing gradients, unstable training.

[1] or divergence, etc.

GAN — Distance Comparison Example

Suppose we have two probability distributions, P and Q:

$$\forall (x, y) \in P, x = 0 \text{ and } y \sim U(0, 1)$$
$$\forall (x, y) \in Q, x = \theta, 0 \leq \theta \leq 1 \text{ and } y \sim U(0, 1)$$

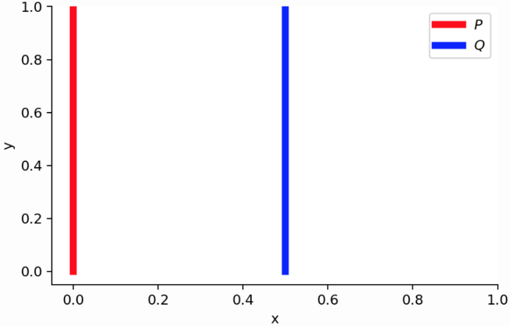


Fig. 8. There is no overlap between P and Q when $\theta \neq 0$.

Diagram:
https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

GAN — Distance Comparison Example

KL

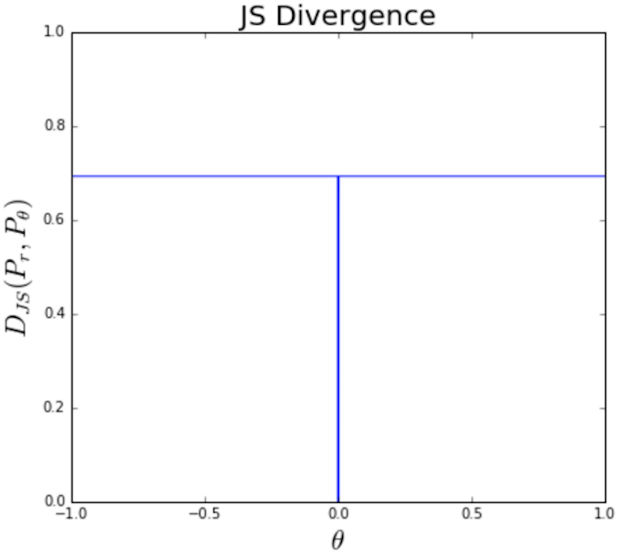
$$D_{KL}(P||Q) = \begin{cases} \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = \infty & \theta \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

JS

$D_{JS} = \log 2$ when $\theta \neq 0$. 0 otherwise.

$\nabla_\theta \mathcal{L}(\theta) = \nabla_\theta D_{JS}(\mathbb{P}_r, \mathbb{P}_\theta)$ is 0 when the distributions don't overlap!

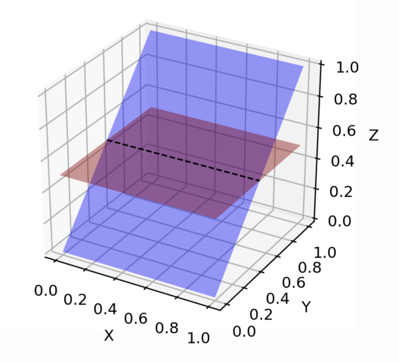
GAN — Distance Comparison Example



GAN — Issues - JS Divergence

More generally, if supports of $\mathbb{P}_1, \mathbb{P}_2$ lie on low dimensional manifolds:

1. With high-probability, their intersection is a set of measure zero[2].



[2] See Lemma 3 [Arjovsky & Bottou 2016]
Diagram:
https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

GAN — Issues - JS Divergence

More generally, if supports of $\mathbb{P}_1, \mathbb{P}_2$ lie on low dimensional manifolds:

- With high-probability, their intersection is a set of measure zero[3].

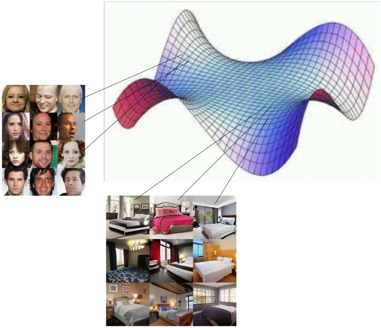
When the supports have measure zero intersection:

- D_{JS} maxes out: $D_{JS}(\mathbb{P}_1, \mathbb{P}_2) = \log 2$.[4]

[3] See Lemma 3 [Arjovsky & Bottou 2016]

[4] See Arjovsky & Bottou 2.3

Our Setting: supports[5] of $\mathbb{P}_R, \mathbb{P}_\theta$ lie on low-dimensional manifolds within a high-dimensional space.



$\mathbb{P}_R$:

$\mathbb{P}_\theta$: Maps low-dimensional z

[5] places where $p(x) \neq 0$

GAN — Issues - JS Divergence

When the supports have measure zero intersection:

- D_{JS} maxes out: $D_{JS} = \log 2$
- There is a perfect discriminator, with **zero gradient everywhere**.

GAN — Issues - JS Divergence

When the supports have measure zero intersection:

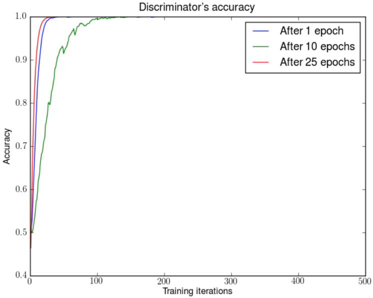
- D_{JS} maxes out: $D_{JS} = \log 2$
- There is a perfect discriminator, with **zero gradient everywhere**.

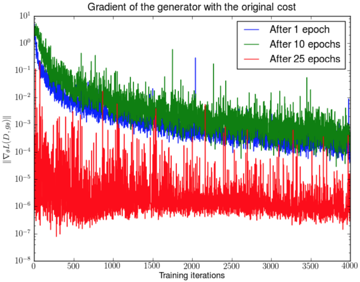
Theorem (Vanishing Gradient)

When the supports of $\mathbb{P}_r$ *and* $\mathbb{P}_\theta$ *are low-dimensional, non-aligned manifolds* M_1,M_2, *there exists a discriminator* D^* *with accuracy 1, and* $\nabla_x D^*(x) = 0$ *for almost every* x *in* M_1 *or* M_2.

See Arjovsky & Bottou 2.2

GAN — Issues - JS Divergence





Perfect discriminator and vanishing gradients.

Arjovsky & Bottou 2016

GAN — Issues - JS Divergence

Workarounds:

- Don't train d_β to convergence (difficult to calibrate, unstable)
- Alternative discriminator loss from GAN paper (unstable)

Want:

- $\rho(\mathbb{P}_R, \mathbb{P}_\theta)$ with useful gradient even when supports do not align.
- Be able to train d_β to convergence.

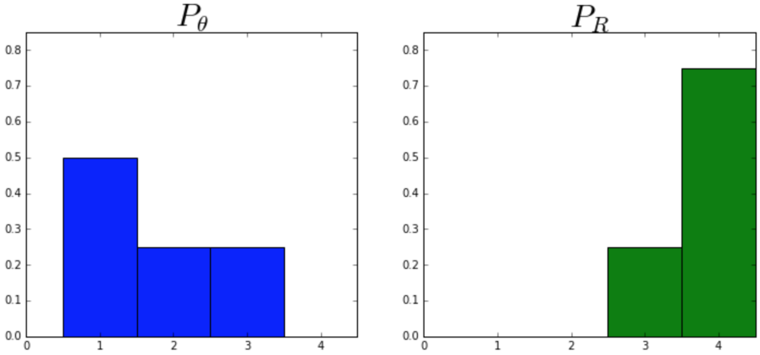
GAN — Wasserstein

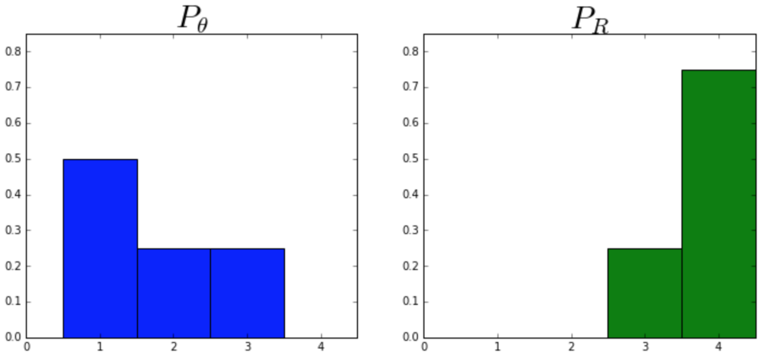
Wasserstein-1 Distance

$$W_1(\mathbb{P}_R, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_R, \mathbb{P}_\theta)} \mathbb{E}_{x,\hat{x} \sim \gamma} ||x - \hat{x}||$$

Earth-Mover Distance

- What is the *minimum cost* of making the distributions equal by *moving mass*?

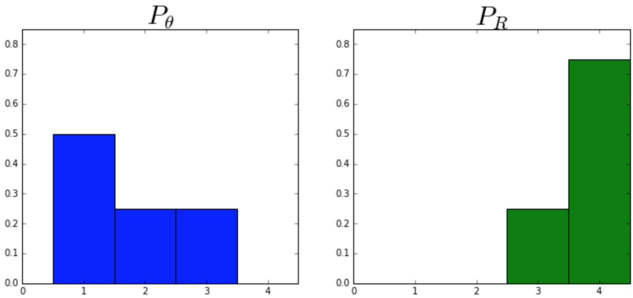




Cost: L_1 distance per unit mass

Define a *coupling* $\gamma(x, y)$ specifying mass to move from x to y.

$\sum_x \gamma(x, y) = P_R(y)$, $\sum_y \gamma(x, y) = P_\theta(x)$.



Example γ:

$$\gamma(1,4) = 0.5$$
$$\gamma(2,4) = 0.25$$
$$\gamma(3,3) = 0.25$$
$$\implies \text{cost} = (0.5) * 3 + (0.25) * 2 + (0.25) * 0$$

Goal: find minimum cost coupling $\gamma \in \Gamma(P_\theta, P_r)$

GAN — Wasserstein-1 / "Earth-Mover" distance

$$W_1(\mathbb{P}_R, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_R, \mathbb{P}_\theta)} \mathbb{E}_{x, \hat{x} \sim \gamma} ||x - \hat{x}||$$

Well-behaved when $\mathbb{P}_R, \mathbb{P}_\theta$ have disjoint or low-dimensional supports!

GAN — Distance Comparison Example

Suppose we have two probability distributions, P and Q:

$$\forall (x, y) \in P, x = 0 \text{ and } y \sim U(0,1)$$
$$\forall (x, y) \in Q, x = \theta, 0 \leq \theta \leq 1 \text{ and } y \sim U(0,1)$$

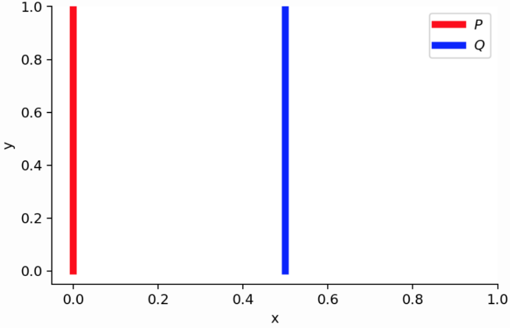


Fig. 8. There is no overlap between P and Q when $\theta \neq 0$.

Diagram: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

GAN — Distance Comparison Example

KL

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

JS

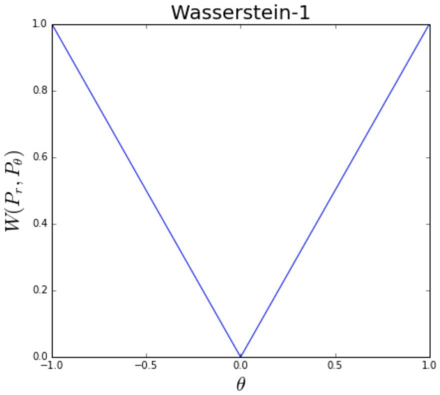
$D_{JS} = \log 2$ when $\theta \neq 0$. 0 otherwise.

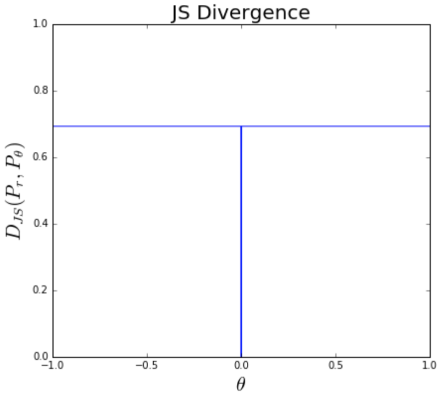
W_1

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

Only Wasserstein is suitable for gradient-based learning.

GAN — Distance Comparison Example





GAN — Optimization

More Generally:[6]

Theorem (Convergence Hierarchy)

Let $\mathbb{P}$ *be a distribution and* $(\mathbb{P}_n)_{n\in\mathbb{N}}$ *be a sequence of distributions over* $\mathcal{X}$*.* $D_{KL}(\mathbb{P}_n||\mathbb{P}) \to 0$ *implies* $D_{JS}(\mathbb{P}_n||\mathbb{P}) \to 0$ *implies* $W(\mathbb{P}_n||\mathbb{P}) \to 0$*.*

Theorem (Wasserstein Continuity and Differentiability)

If $g_\theta(z)$ *is a feedforward neural network and* $p(z) \sim U(0,1)$*, then* $W(\mathbb{P}_r, \mathbb{P}_\theta)$ *is* **continuous everywhere** *and* **differentiable almost everywhere***.*

Continuity and differentiability theorem does not hold for D_{KL}, D_{JS}!

[6] See [Arjovsky 2017] for general statements and proofs.

GAN — Distance Summary

- Setting: **gradient-based** learning on distributions with supports on **low-dimensional manifolds**.
- Standard GAN (D_{JS}) discriminator quickly becomes too good.
- D_{JS} (and D_{KL}) do not provide useful gradients.
- Workarounds are heuristic and typically unstable.
- Wasserstein distance has good properties (continuity, differentiability).

Next: How to minimize Wasserstein distance in practice?

WGAN

[Arjovsky et al 2017]

GAN — Wasserstein-1 / "Earth-Mover" distance

Wasserstein GAN (WGAN): A practical method for optimizing W_1.

$$W_1(\mathbb{P}_R, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_R, \mathbb{P}_\theta)} \mathbb{E}_{x, \hat{x} \sim \gamma} ||x - \hat{x}||$$

Intractable due to $|\Gamma|$.

WGAN: A practical method for optimizing W_1.

$$W_1(\mathbb{P}_R, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_R, \mathbb{P}_\theta)} \mathbb{E}_{x,\hat{x}\sim\gamma} ||x - \hat{x}||$$

Dual form (Kantorovich-Rubinstein):

$$W_1(\mathbb{P}_R, \mathbb{P}_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

$||f||_L \leq 1$: 1-Lipschitz continuous functions.[7]

[7] there is a $K \geq 0$ such that $\forall x_1, x_2, |f(x_1) - f(x_2)| \leq K|x_1 - x_2|$; "bounded slope"

WGAN — Kantorovich-Rubinstein Proof Outline

$$W_1(\mathbb{P}_R, \mathbb{P}_\theta) = \inf_{\gamma \in \Gamma(\mathbb{P}_R, \mathbb{P}_\theta)} \mathbb{E}_{x,\hat{x}\sim\gamma} ||x - \hat{x}||$$

$$\equiv \inf_{\gamma} \mathbb{E}_{x,\hat{x}\sim\gamma} \left[||x - \hat{x}|| + \underbrace{\sup_f \mathbb{E}_{s\sim P_R}[f(s)] - \mathbb{E}_{t\sim P_\theta}[f(t)] - (f(x) - f(\hat{x}))}_{0 \text{ if } \gamma \in \Gamma \text{ else } \infty} \right]$$

$$= \underbrace{\inf_\gamma \sup_f}_{\text{move sup}} \mathbb{E}_{x,\hat{x}\sim\gamma} [||x - \hat{x}|| + \mathbb{E}_{s\sim P_R}[f(s)] - \mathbb{E}_{t\sim P_\theta}[f(t)] - (f(x) - f(\hat{x}))]$$

$$= \underbrace{\sup_f \inf_\gamma}_{\text{minimax thm.}} \mathbb{E}_{x,\hat{x}\sim\gamma} [||x - \hat{x}|| + \mathbb{E}_{s\sim P_R}[f(s)] - \mathbb{E}_{t\sim P_\theta}[f(t)] - (f(x) - f(\hat{x}))]$$

$$= \sup_f \mathbb{E}_{s\sim P_R}[f(s)] - \mathbb{E}_{t\sim P_\theta}[f(t)] + \underbrace{\inf_\gamma [\mathbb{E}_{x,\hat{x}\sim\gamma} ||x - \hat{x}|| - (f(x) - f(\hat{x}))]}_{0 \text{ if } ||f||_L \leq 1 \text{ else } -\infty}$$

$$= \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \quad \square$$

WGAN — Interpretation

WGAN Objective

$$W_1(\mathbb{P}_R, \mathbb{P}_\theta) = \max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

where $\{f_w\}_{w \in \mathcal{W}}$ are parametrized 1-Lipschitz functions.

Lipschitz constraint:
[Arjovsky et al. 2017]: *weight clipping*
[Gulrajani et al. 2017]: *gradient penalty*

WGAN — Interpretation

$$\underbrace{\max_{w \in \mathcal{W}}}_{\text{find } critic\ f_w} \quad \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

WGAN — Interpretation

$$\max_{w \in \mathcal{W}} \underbrace{\mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)]}_{\text{critic gives high score for real samples}} - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

WGAN — Wasserstein Distance

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] \quad \underbrace{-\mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]}_{\text{critic gives low score for fake samples}}$$

WGAN — Wasserstein Distance

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

"Generator" gradient: $\underbrace{-\mathbb{E}_{z \sim p(z)}[\nabla_\theta f_w(g_\theta(z))]}_{\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta)}$

WGAN — Algorithm

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

WGAN — Results (Arjovsky et al 2017)

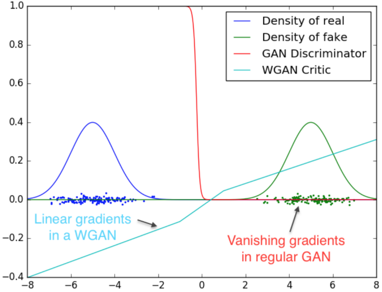


Figure: The WGAN critic can provide useful gradients at optimality

Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]).

Figure: WGAN was more robust to various architectural changes

WGAN — Results (Arjovsky et al 2017)

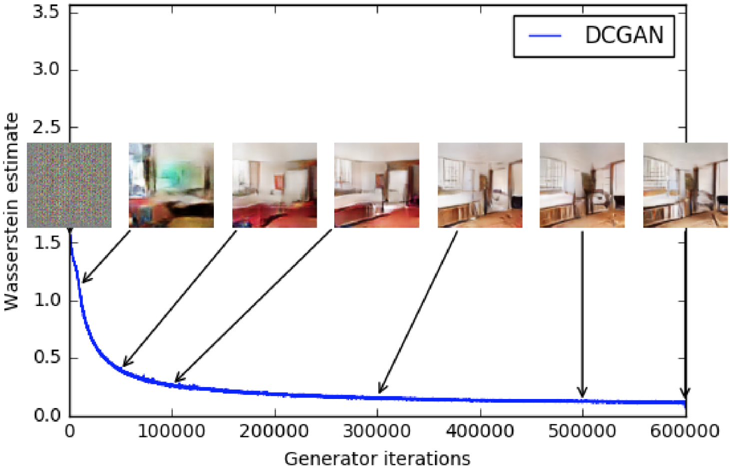


Figure: WGAN has a meaningful loss metric

WGAN — Summary

- Wasserstein distance is more suitable than D_{JS}.
- Optimize Wasserstein distance using a dual objective.
- Objective analogous to GAN ('critic' + 'generator').

WGAN — Summary II

156

[R] [1701.07875] Wasserstein GAN (arxiv.org)
submitted 1 year ago by ajmooch
167 comments share save hide give gold report

all 167 comments
sorted by: best

[–] danielvarga 34 points 1 year ago

- For mathematicians: it uses Wasserstein distance instead of Jensen-Shannon divergence to compare distributions.
- For engineers: it gets rid of a few unnecessary logarithms, and clips weights.
- For others: it employs an art critic instead of a forgery expert.

Applications and Extensions

Image Modeling

$\mathbb{P}_r$: Distribution over images
$\hat{x} \sim \mathbb{P}_\theta$: Sampled image

Image Modeling

Original GAN

Goodfellow et al. 2014

Toronto Face Dataset

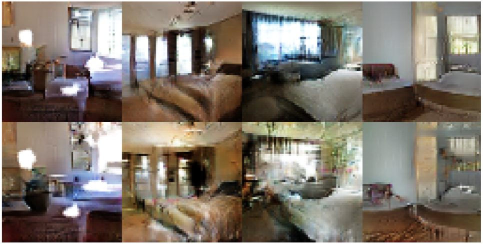
CIFAR 10

Image Modeling

DC-Gan

Radford et al. 2015

Face Dataset



LSUN Bedroom

Image Modeling

WGAN-GP

Gulrajani et al. 2017

CelebA

LSUN Bedroom

Progressive GAN

Karras et al. 2018

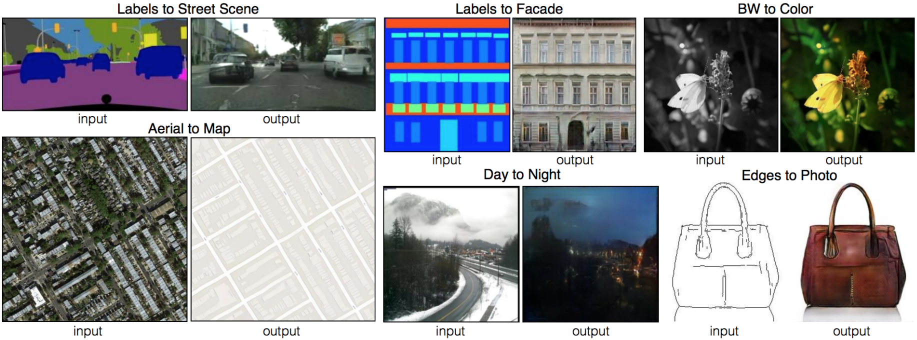
CelebA

LSUN Bedroom

Image-to-Image Translation

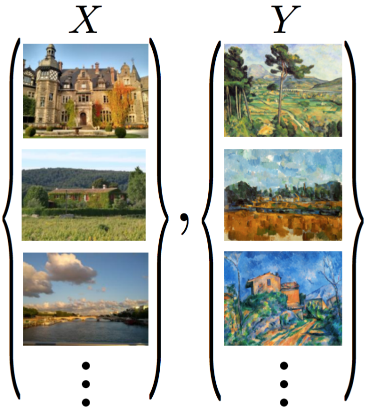
diagram: Zhu et al. 2017

Image-to-Image Translation



Isola et al. 2016

Unpaired Image-to-Image Translation



Zhu et al. 2017

Unpaired Image-to-Image Translation

Zhu et al. 2017

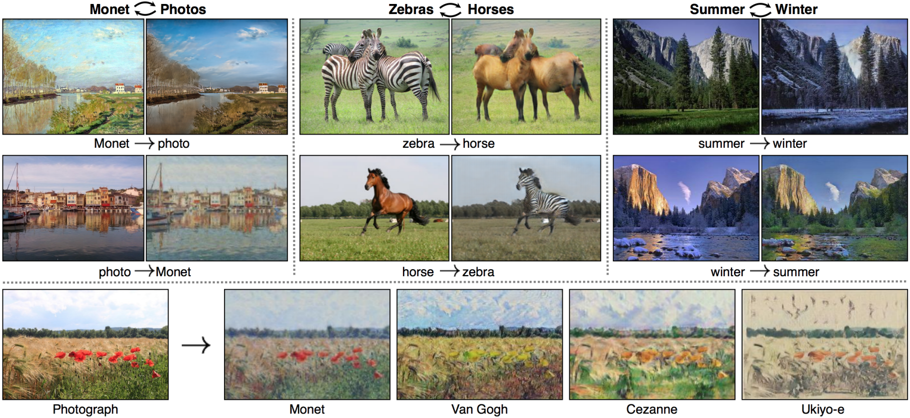


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

$$\mathcal{L} = \mathcal{L}_{GAN_1} + \mathcal{L}_{GAN_2} + \mathcal{L}_{cycle}$$

The End

Other Resources

- Kantorovich-Rubinstein Duality Proof: https://vincentherrmann.github.io/blog/wasserstein/