Wasserstein GANs

Yong Zhuang

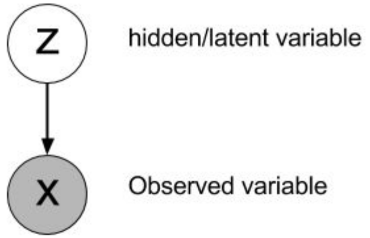
Agenda

- ❖ Variational Inference
- ❖ Generative Adversarial Networks (GANs)
- ❖ What’s wrong with training GANs?
- ❖ Wasserstein GANs
- ❖ Improved Training of Wasserstein GANs

Variational Inference

In short, variational inference is akin to what happens at every presentation you've attended. Someone in the audience asks the presenter a very difficult answer which he/she can't answer. The presenter conveniently reframes the question in an easier manner and gives an exact answer to that reformulated question rather than answering the original difficult question.

In many interesting statistical problems, we can't directly calculate the posterior because the normalization constant is intractable. This happens often in latent variable models. For example assume that X represents a set of observations and Z represents a set of latent variables. If we are interested in the posterior P(Z|X), we know that :



$$P(Z|X) = \frac{P(Z,X)}{\int_z P(Z,X)}$$

but often times we can't calculate the denominator.

Variational Inference

Variational inference seeks to approximate the true posterior, P(Z|X), with an approximate variational distribution, which we can calculate more easily. For notation, let V be the parameters of the variational distribution.

$$P(Z|X) \approx Q(Z|V) = \prod_i Q(Z_i|V_i)$$

Typically, in the true posterior distribution, the latent variables are not independent given the data, but if we restrict our family of variational distributions to a distribution that factorizes over each variable in Z (this is called a mean field approximation), our problem becomes a lot easier. We can easily pick each V_i so that Q(Z|V) is as close to P(Z|X) as possible when measured by Kullback Leibler (KL) divergence. Thus, our problem of interest is now selecting a V* such that

$$V^\star = \arg\min_V KL(Q(Z|V)||P(Z|X))$$

Once we arrive at a V*, we can use Q(Z|V*) as our best guess at the posterior when performing estimation or inference.

Connections to deep learning : Variational Autoencoders (VAE) GENERATIVE ADVERSARIAL NETWORKS (GAN)

Agenda

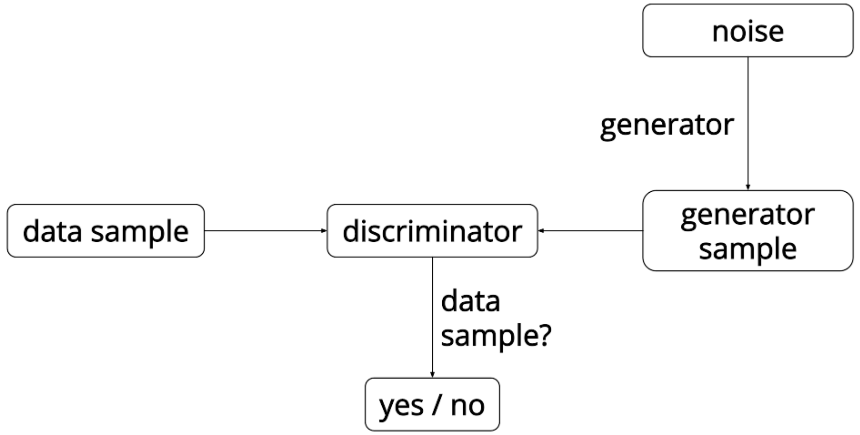
- ❖ Variational Inference
- ❖ Generative Adversarial Networks (GANs)
- ❖ What’s wrong with training GANs?
- ❖ Wasserstein GANs
- ❖ Improved Training of Wasserstein GANs

Generative Adversarial Networks (GAN)

- Two networks:
  1. Discriminator D
  2. Generator G

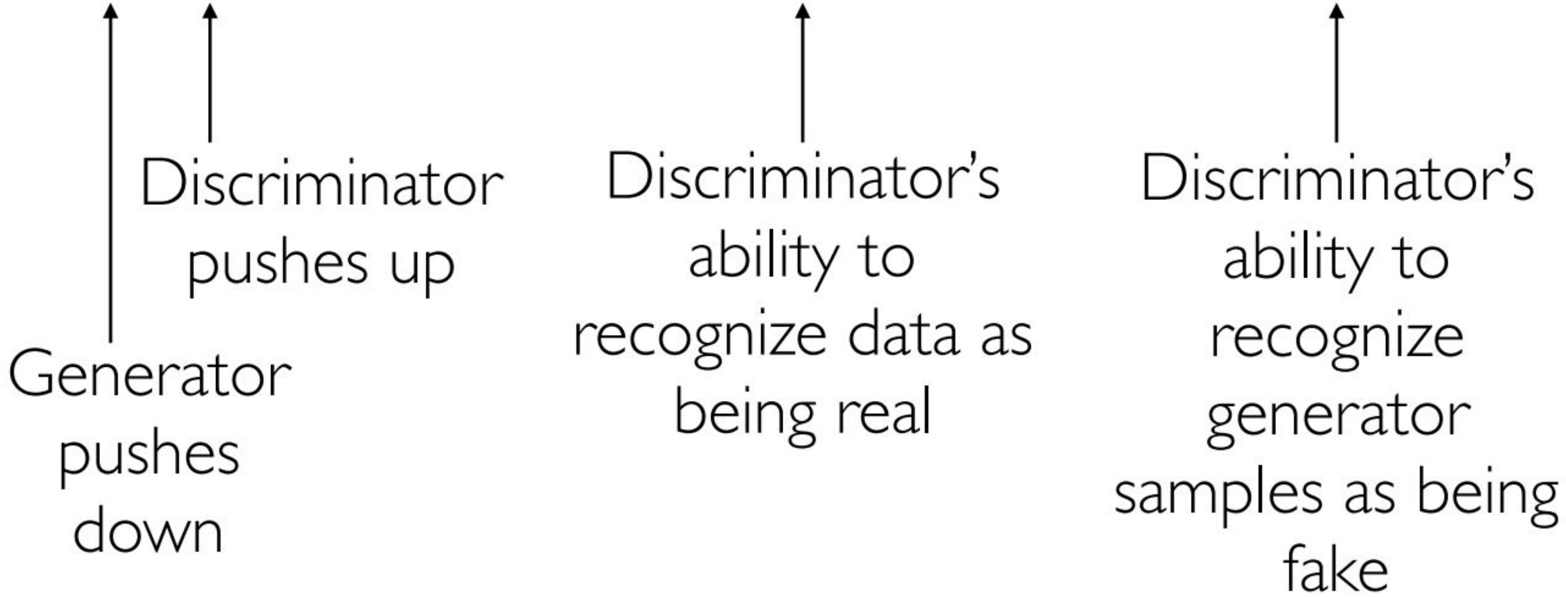
- D tries to discriminate between:
  1. A sample from the data distribution.
  2. And a sample from the generator G.

- G tries to "trick" D by generating samples that are hard for D to distinguish from data.



Generative Adversarial Networks (GAN)

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1 - D(x))] \tag{1}$$


Generator pushes down
Discriminator pushes up
Discriminator's ability to recognize data as being real
Discriminator's ability to recognize generator samples as being fake

Generative Adversarial Networks (GAN)

- Loss for the generator (version 1) L1: $\mathbb{E}_{x\sim P_g}[\log(1 - D(x))]$ (2)

To avoid gradients vanishing when the discriminator is very confident, people have chosen to use a different gradient step for the generator

- Loss for the generator (version 2) L2: $\mathbb{E}_{x\sim P_g}[-\log D(x)]$ (3)

Agenda

- ❖ Variational Inference
- ❖ Generative Adversarial Networks (GANs)
- ❖ What’s wrong with training GANs?
- ❖ Wasserstein GANs
- ❖ Improved Training of Wasserstein GANs

Train Discriminator

In the Discriminator, the cross entropy is :

$$-P_r(x)\log D(x) - P_g(x)\log[1 - D(x)]$$

Let derivative of D(x) = 0 then we get:

$$-\frac{P_r(x)}{D(x)} + \frac{P_g(x)}{1 - D(x)} = 0$$

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)} \qquad (4)$$

Train Generator

Loss : $\mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1 - D(x))]$

And: $D^*(x) = \dfrac{P_r(x)}{P_r(x) + P_g(x)}$

then we get:

$$\mathbb{E}_{x\sim P_r} \log \frac{P_r(x)}{\frac{1}{2}[P_r(x) + P_g(x)]} + \mathbb{E}_{x\sim P_g} \log \frac{P_g(x)}{\frac{1}{2}[P_r(x) + P_g(x)]} - 2\log 2$$

Train Generator

Kullback-Leibler (KL) divergence: $KL(\mathbb{P}_r \| \mathbb{P}_g) = \mathbb{E}_{x \sim P_r} \log \frac{P_r(x)}{P_g(x)}$

Jensen-shannon divergence: $JS(\mathbb{P}_r \| \mathbb{P}_g) = \frac{1}{2} KL(\mathbb{P}_r \| \mathbb{P}_A) + \frac{1}{2} KL(\mathbb{P}_g \| \mathbb{P}_A)$

where $\mathbb{P}_A$ is the 'average' distribution, with density $\frac{P_r + P_g}{2}$

then we get: $\mathbb{E}_{x \sim P_r} [\log D(x)] + \mathbb{E}_{x \sim P_g} [\log(1 - D(x))]$

$$= \mathbb{E}_{x \sim P_r} \log \frac{P_r(x)}{\frac{1}{2}[P_r(x) + P_g(x)]} + \mathbb{E}_{x \sim P_g} \log \frac{P_g(x)}{\frac{1}{2}[P_r(x) + P_g(x)]} - 2 \log 2$$

$$= 2JS(P_r || P_g) - 2 \log 2$$

Train Generator

Lemma 3. *Let $\mathcal{M}$ and $\mathcal{P}$ be two regular submanifolds of $\mathbb{R}^d$ that don't perfectly align and don't have full dimension. Let $\mathcal{L} = \mathcal{M} \cap \mathcal{P}$. If $\mathcal{M}$ and $\mathcal{P}$ don't have boundary, then $\mathcal{L}$ is also a manifold, and has strictly lower dimension than both the one of $\mathcal{M}$ and the one of $\mathcal{P}$. If they have boundary, $\mathcal{L}$ is a union of at most 4 strictly lower dimensional manifolds. In both cases, $\mathcal{L}$ has measure 0 in both $\mathcal{M}$ and $\mathcal{P}$.*

Theorem 2.3. *Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distributions whose support lies in two manifolds $\mathcal{M}$ and $\mathcal{P}$ that don't have full dimension and don't perfectly align. We further assume that $\mathbb{P}_r$ and $\mathbb{P}_g$ are continuous in their respective manifolds. Then,*

$$JSD(\mathbb{P}_r \| \mathbb{P}_g) = \log 2$$
$$KL(\mathbb{P}_r \| \mathbb{P}_g) = +\infty$$
$$KL(\mathbb{P}_g \| \mathbb{P}_r) = +\infty$$

So $\mathbb{E}_{x \sim P_r}[\log D(x)] + \mathbb{E}_{x \sim P_g}[\log(1 - D(x))]$ = $2JS(P_r || P_g) - 2\log 2$ = 0

Train Generator

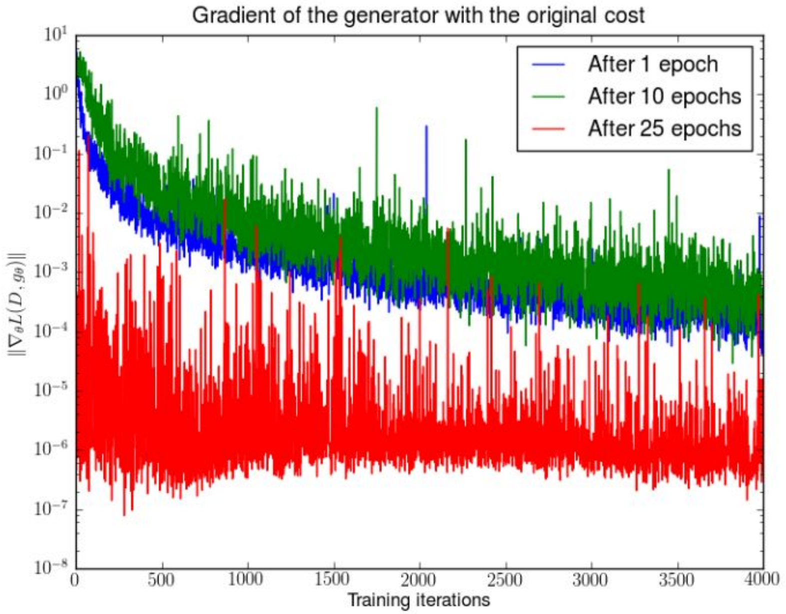


Figure 2: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch and measure the gradients with the original cost function. We see the gradient norms decay quickly, in the best case 5 orders of magnitude after 4000 discriminator iterations. Note the logarithmic scale.

Train Generator

Loss for the generator (version 2) L2: $\mathbb{E}_{x\sim P_g}[-\log D(x)]$

Optimal D: $D^*(x) = \dfrac{P_r(x)}{P_r(x)+P_g(x)}$

And:

$$\begin{aligned}
KL(P_g||P_r) &= \mathbb{E}_{x\sim P_g}[\log \frac{P_g(x)}{P_r(x)}] \\
&= \mathbb{E}_{x\sim P_g}[\log \frac{P_g(x)/(P_r(x)+P_g(x))}{P_r(x)/(P_r(x)+P_g(x))}] \\
&= \mathbb{E}_{x\sim P_g}[\log \frac{1-D^*(x)}{D^*(x)}] \\
&= \mathbb{E}_{x\sim P_g} \log[1-D^*(x)] - \mathbb{E}_{x\sim P_g} \log D^*(x)
\end{aligned}$$

And we already got: $\mathbb{E}_{x\sim P_r}[\log D^*(x)] + \mathbb{E}_{x\sim P_g}[\log(1-D^*(x))] = 2JS(P_r||P_g) - 2\log 2$

then we got: min $\mathbb{E}_{x\sim P_g}[-\log D(x)] \propto$ min $KL(P_g||P_r) - 2JS(P_r||P_g)$

Training Generator

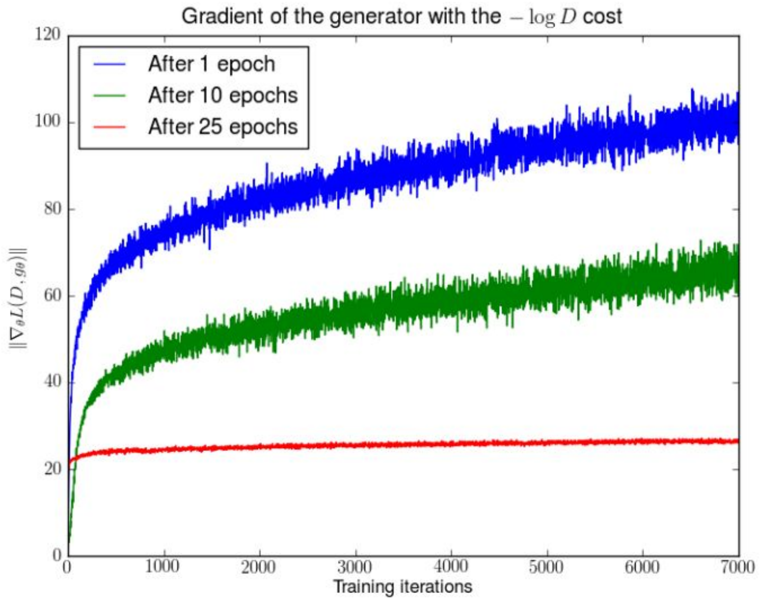
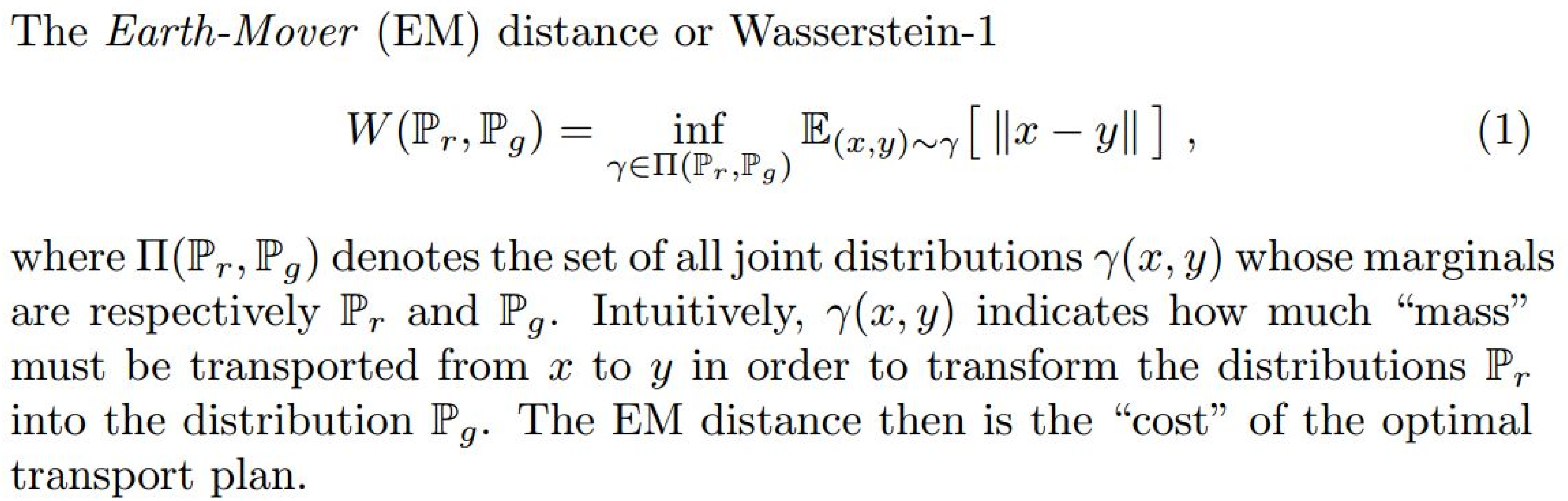


Figure 3: First, we trained a DCGAN for 1, 10 and 25 epochs. Then, with the generator fixed we train a discriminator from scratch and measure the gradients with the $-\log D$ cost function. We see the gradient norms grow quickly. Furthermore, the noise in the curves shows that the variance of the gradients is also increasing. All these gradients lead to updates that lower sample quality notoriously.

Agenda

- ❖ Variational Inference
- ❖ Generative Adversarial Networks (GAN)
- ❖ What's wrong with training GANs?
- ❖ Wasserstein GANs
- ❖ Improved Training of Wasserstein GANs

Earth-Mover Distance



The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \big[\, \|x - y\| \, \big] \, , \tag{1}$$

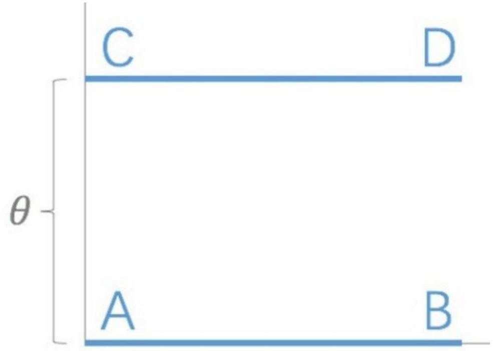
where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

inf: Infimum - *lower bound*

Earth-Mover Distance

Example 1 (Learning parallel lines)**.** Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$



Earth-Mover Distance

Lipschitz continuous: In particular, a real-valued function $f : R \to R$ is called Lipschitz continuous if there exists a positive real constant K such that, for all real x_1 and x_2

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

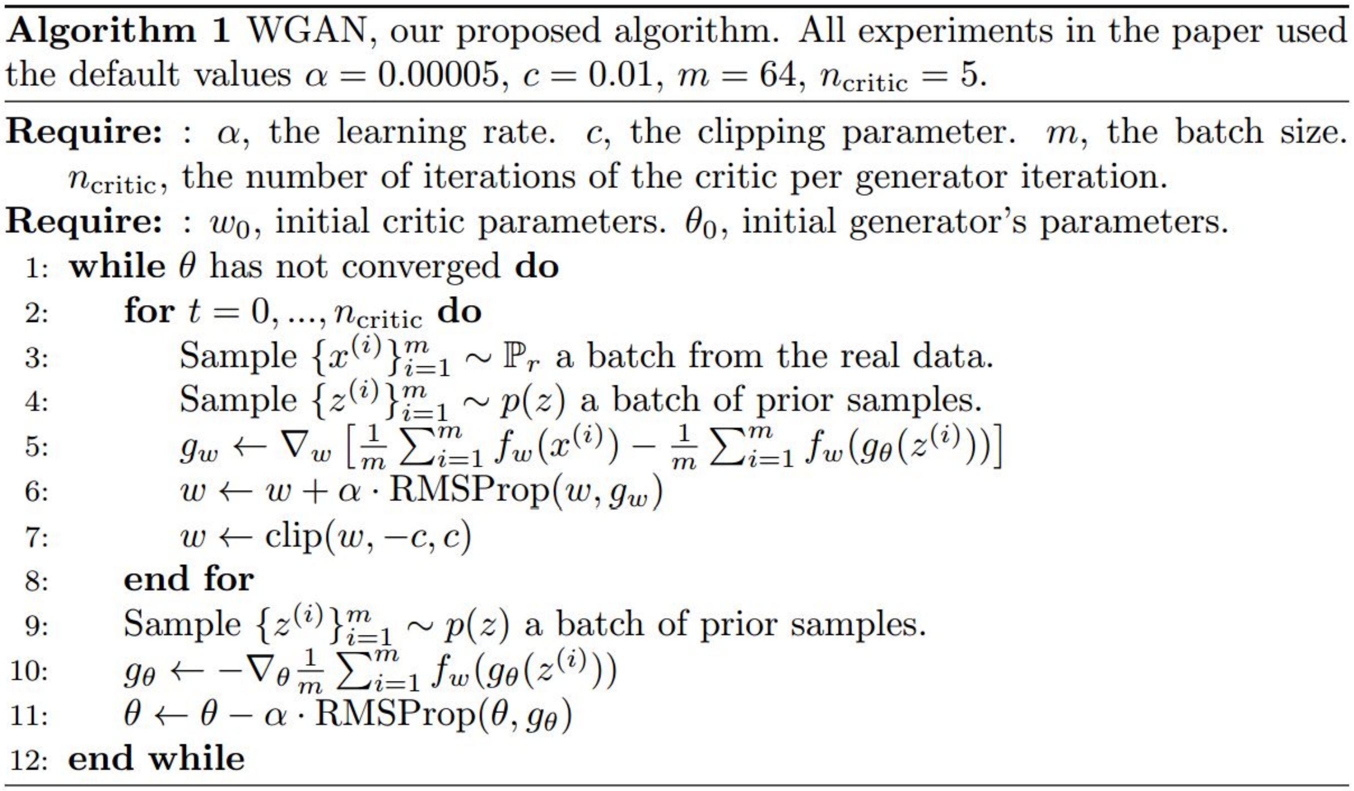
$$W(P_r, P_g) = \frac{1}{K} \sup_{||f||_L \leq K} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]$$

$$\approx \frac{1}{K} \max_{w : |f_w|_L \leq K} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{x \sim P_g}[f_w(x)]$$

For $||f_w||_L \leq K$, we simply set a range for w_i, e.g. $w_i \in [-0.01, 0.01]$, then $\frac{\partial f_w}{\partial x}$ < some constant. So every training step, we need to do weight clipping.

WGAN

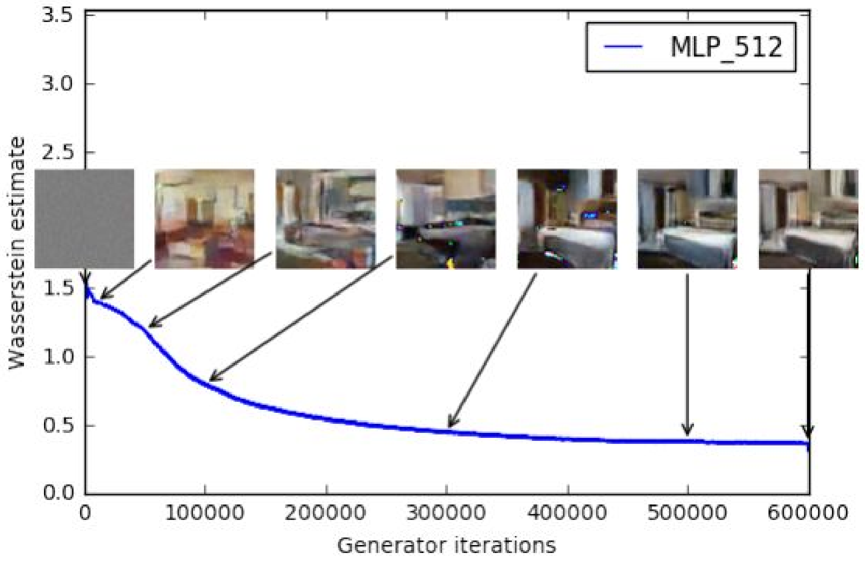
New loss function: $L = \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{x \sim P_g}[f_w(x)]$

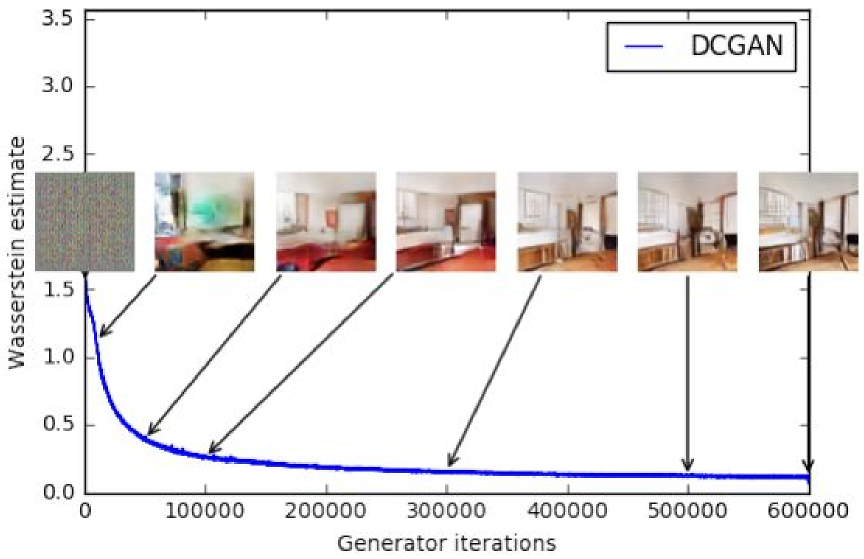


Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

```
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
 1: while $\theta$ has not converged do
 2:     for $t = 0, ..., n_{\text{critic}}$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
 5:         $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
 6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
 7:         $w \leftarrow \text{clip}(w, -c, c)$
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:     $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:     $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

WGAN


MLP_512
Wasserstein estimate
Generator iterations
0.0
0.5
1.0
1.5
2.5
3.0
3.5
0
100000
200000
300000
400000
500000
600000


DCGAN
Wasserstein estimate
Generator iterations
0.0
0.5
1.0
1.5
2.5
3.0
3.5
0
100000
200000
300000
400000
500000
600000

Agenda

- ❖ Variational Inference
- ❖ Generative Adversarial Networks (GANs)
- ❖ What’s wrong with training GANs?
- ❖ Wasserstein GANs
- ❖ Improved Training of Wasserstein GANs

UMASS BOSTON

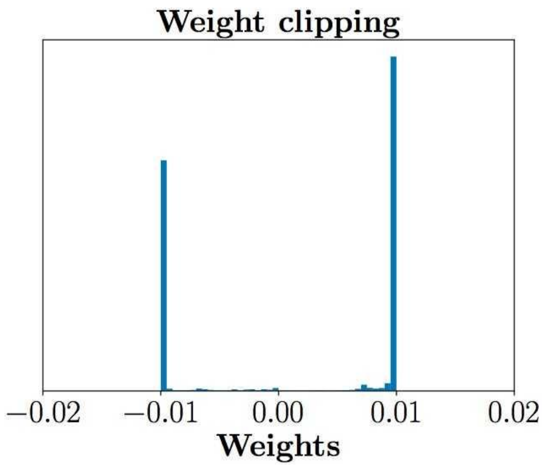
What's wrong in WGAN?

Loss function: $L(D) = -\mathbb{E}_{x\sim P_r}[D(x)] + \mathbb{E}_{x\sim P_g}[D(x)]$

$$L(G) = -\mathbb{E}_{x\sim P_g}[D(x)]$$

subject to $||\nabla_x D(x)||_p \leq K, \forall x \in \mathcal{X}$

So every training step, we need to do weight clipping to let $w_i \in [-0.01, 0.01]$



Gradient Penalty

Loss function: $L(D) = -\mathbb{E}_{x\sim P_r}[D(x)] + \mathbb{E}_{x\sim P_g}[D(x)]$

$L(G) = -\mathbb{E}_{x\sim P_g}[D(x)]$

subject to $||\nabla_x D(x)||_p \leq K, \forall x \in \mathcal{X}$

~~So every training step, we need to do weight clipping to let $w_i \in [-0.01, 0.01]$~~

Gradient Penalty: $[||\nabla_x D(x)||_p - K]^2$

New Loss function: $L(D) = -\mathbb{E}_{x\sim P_r}[D(x)] + \mathbb{E}_{x\sim P_g}[D(x)] + \lambda \mathbb{E}_{x\sim \mathcal{X}}[||\nabla_x D(x)||_p - 1]^2$

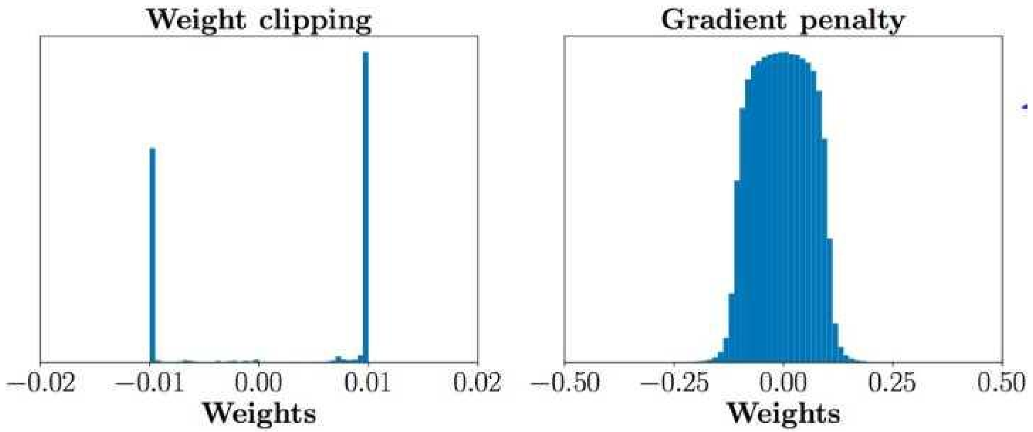
Gradient Penalty

New Loss function: $L(D) = -\mathbb{E}_{x\sim P_r}[D(x)] + \mathbb{E}_{x\sim P_g}[D(x)] + \lambda\mathbb{E}_{x\sim\mathcal{X}}[||\nabla_x D(x)||_p - 1]^2$

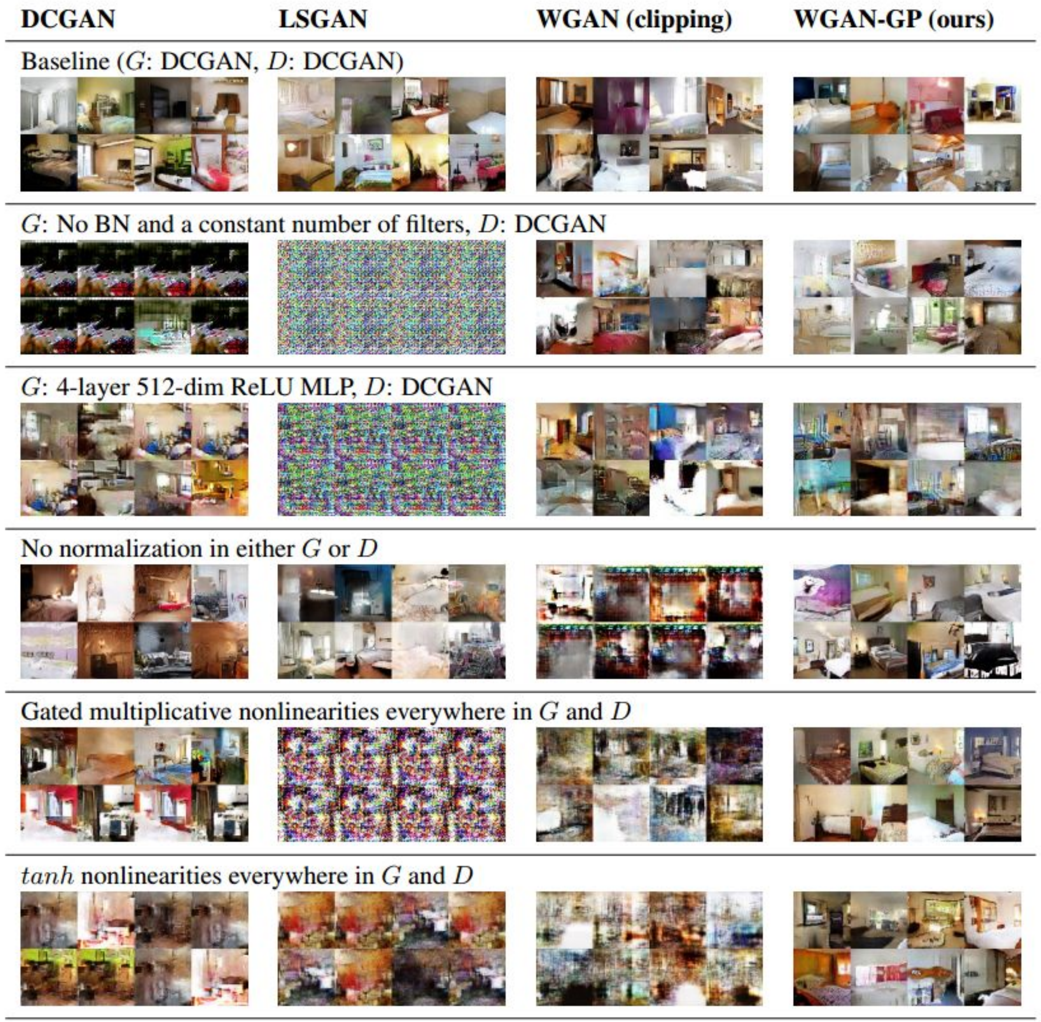
How to calculate expected value $\lambda\mathbb{E}_{x\sim\mathcal{X}}[||\nabla_x D(x)||_p - 1]^2$ on $\mathcal{X}$?

$$x_r \sim P_r, x_g \sim P_g, \epsilon \sim Uniform[0,1]$$
$$\hat{x} = \epsilon x_r + (1-\epsilon)x_g$$

Updated Loss function: $L(D) = -\mathbb{E}_{x\sim P_r}[D(x)] + \mathbb{E}_{x\sim P_g}[D(x)] + \lambda\mathbb{E}_{x\sim\mathcal{P}_{\hat{x}}}[||\nabla_x D(x)||_p - 1]^2$



WGAN - GP


DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN
No normalization in either G or D
Gated multiplicative nonlinearities everywhere in G and D
$tanh$ nonlinearities everywhere in G and D

References

Generative Adversarial Networks: https://arxiv.org/abs/1406.2661

Towards principled methods for training generative adversarial networks: https://arxiv.org/pdf/1701.04862.pdf

Wasserstein GAN: https://arxiv.org/pdf/1701.07875.pdf

Improved Training of Wasserstein GANs : https://arxiv.org/pdf/1704.00028.pdf

Thank You!

UMASS BOSTON