Computing the GAN Equilibrium

- Theory of GAN convergence
- Alternating minimization ⇒ Mode collapse
- Generator loss gradient descent
- Practical GAN convergence

GAN Equilibrium Conditions
(Non-Saturating)

- We would like to find the solution to:

$$\theta_g^* = \arg\min_{\theta_g} g\left(\theta_g, \theta_d^*\right)$$
$$= \arg\min_{\theta_g} E\left[-\log\left(f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right]$$

$$\theta_d^* = \arg\min_{\theta_d} d\left(\theta_g^*, \theta_d\right)$$
$$= \arg\min_{\theta_d}\left\{E\left[-\log f_{\theta_d}(Y)\right] + E\left[-\log\left(1 - f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right]\right\}$$

- Observations:
  - Known as a Nash Equilibrium
  - At convergence:

$$R_{\theta_g}(y) = \frac{p_{\theta_g}(y)}{p_r(y)} = 1 \quad \Rightarrow \text{(generated distribution = reference distributions)}$$

$$f_{\theta_d}(y) = \frac{1}{1+R_{\theta_g}(y)} = \frac{1}{2} \quad \Rightarrow \text{(discriminator can't distinguish between distributions)}$$

Reparameterize Loss Functions

- Goal: Replace θ_g and θ_d with R and f
- Generator parameter R:
  - Generated samples are

$$\tilde{Y} \sim R(y)\, p_r(y) = p_R(y)$$

where $R(y) = \frac{p_{\theta_g}(y)}{p_r(y)}$

- Discriminator parameter f:
  - Discriminator is

$$f(y) = P\{Class = R|y\}$$

- Important facts:
  - $E[R(Y)] = 1$
  - $\Omega_g = \{R: \Re^N \rightarrow (0, \infty) \text{ such that } E[R(Y)] = 1\}$
  - $\Omega_d = \{f: \Re^N \rightarrow (0,1)\}$
  - For any function $h(y)$, $E\left[h\left(\tilde{Y}\right)\right] = E[h(Y)R(Y)]$

GAN Equilibrium Conditions

- We would like to find an algorithm that gives a solution to:

$$R^* = \arg\min_{R \in \Omega_g} g(R, f^*)$$

$$f^* = \arg\min_{f \in \Omega_d} d(R^*, f)$$

  - This is known as a Nash Equilibrium
  - We would like it to converge to $R^* = 1 \Rightarrow$ (generated = reference distributions)
  - At convergence:

    $R^*(y) = 1$ $\Rightarrow$ (generated distribution = reference distributions)

    $f^*(y) = \frac{1}{1+R^*(y)} = \frac{1}{2}$ $\Rightarrow$ (discriminator can't distinguish between distributions)

- How do we do this?
- Will it converge?

Reparameterized Loss Functions

$Y \sim$ from real distribution
$\tilde{Y} \sim$ from fake distribution

- Generator loss function:

$$g(R, f) = E\left[-\log f\left(\tilde{Y}\right)\right]$$
$$= E[-R(Y)\log f(Y)]$$

- Discriminator loss function:

$$d(R, f) = E[-\log f(Y)] + E\left[-\log\left(1 - f\left(\tilde{Y}\right)\right)\right]$$
$$= E[-\log f(Y)] + E\left[-R(Y)\log\left(1 - f(Y)\right)\right]$$

- Nash equilibrium:

$$R^* = \arg\min_{R \in \Omega_g} g(R, f^*)$$

$$f^* = \arg\min_{f \in \Omega_d} d(R^*, f)$$

Algorithm 1: Alternating Minimization

- Algorithm

Repeat {

$$\hat{f} \leftarrow \arg\min_{f \in \Omega_d} d(\hat{R}, f)$$

$$\hat{R} \leftarrow \arg\min_{R \in \Omega_g} g(R, \hat{f})$$

}

- Discriminator update

$$\hat{f}(y) \leftarrow \frac{1}{1 + R(y)}$$ (probability its real)

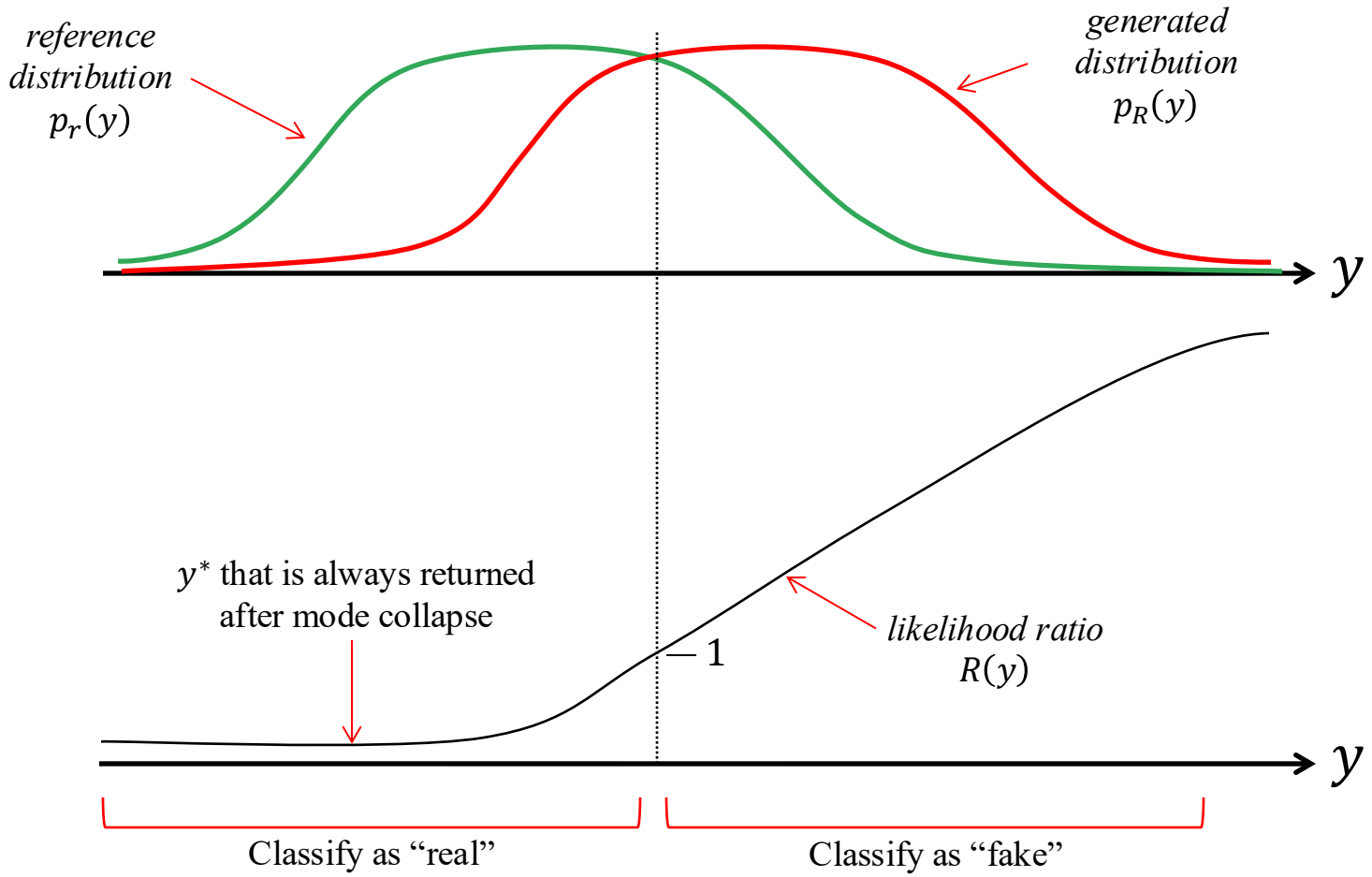
- Generator update

$$\hat{R}(y) \leftarrow \frac{1}{p_r(y_o)} \delta(y - y_o) \text{ where } y_o = \max_y f(y) = \min_y R(y)$$

Problem:

- This is called "Mode collapse"
- Only generates samples that the discriminator likes best
- Intuition:
  "We come from France." "I like cheese steaks."
  "Too good to be true."
  "Too creepy to be real."

Bayes Discriminator and the Likelihood Ratio



- Bayesian Discriminator:

$$f^*(y) \approx P\{Class = R|y\} = \frac{p_r(y)}{p_r(y) + p_R(y)} = \frac{1}{1 + R_R(y)}$$

Alternative Approach

- Define

$$f^*(R) = \arg\min_{f \in \Omega_d} d(R, f)$$

  - Intuition: $f^*(R)$ gives you the best discriminator for each generator.

- Then we can define a single loss function

$$L(R) = g(R, f^*(R))$$

  - Intuition: $L(R)$ is the loss you get with each generator

- Then directly minimize the new loss

$$R^* = \arg\min_{R \in \Omega_g} L(R)$$

- Intuition:
  - Minimize the discriminator a lot for each generator step.

Algorithm 2: Generator Loss Gradient Descent (GLGD)

- Algorithm

Repeat {

$$f^* \leftarrow \arg\min_{f \in \Omega_d} d(R, f)$$

$$R \leftarrow R - \alpha P_o \nabla_R g(R, f^*)$$

}

My term

gradient descent step

Projection onto valid parameter space

- Discriminator update

$$f^*(y) \leftarrow \frac{1}{1 + R(y)}$$

- Generator update
  - Take a step in the negative direction of the generator loss gradient.
  - P_o - project into the allow parameter space. (This is not an issue in practice.)

$$P_o d(y) = d(y) - \frac{E[d(Y)]}{E[p_r(Y)]} p_r(y)$$

- Questions/Comments:
  - Can be applied with a wide variety of generator/discriminator loss functions
  - Does this converge?
  - If so, then what (if anything) is being minimized?

*Martin Arjovsky and Leon Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", ICLR 2017.

GLGD Convergence for Non-Saturating GAN

Repeat {

$$f^* \leftarrow \arg\min_{f \in \Omega_d} d(R, f)$$

$$R \leftarrow R - \alpha P_o \nabla_R g(R, f^*)$$

}

- For non-saturating GAN when $f_R^*(y) = \arg\min_{f \in \Omega_d} d(R, f)$, then it can be shown that

$$\nabla_R \, g(R, f^*) = \nabla_R C(R)$$

where*

$$C(R) = E\left[\left(1 + R(Y)\right)\log\left(1 + R(Y)\right)\right]$$

- Conclusions:
  - GLGD is really a gradient descent algorithm for the cost function $C(R)$.
  - $h(x) = (1 + x)\log(1 + x)$ is a strictly convex function on $x \in \Re^{\mathrm{N}}$.
  - Therefore, it can be shown that $C(R)$ has a unique global minimum on Ω_g at $R(Y) = 1$.
  - However, convergence tends to be slow.

*This is an equivalent expression to Theorem 2.5 of [1].

[1] Martin Arjovsky and Leon Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", ICLR 2017.

Extra Stuff

- Some important functions
- Some useful relationships

More Details on GLGD

Repeat {

$$f^* \leftarrow \arg\min_{f \in \Omega_d} d(R, f)$$

$$R \leftarrow R - \alpha P_o \nabla_R g(R, f^*)$$

}

- Arjovsky and Bottou showed that for the non-saturating GAN*

$$P_o \nabla_R g(R, f^*) = \nabla_R \{KL(p_R || p_r) - 2\, JSD(p_R || p_r)\}$$

- So from previous identities, we have that:

$$P_o \nabla_R g(R, f^*) = \nabla_R \left\{2\, KL\big((p_R + p_r)/2 || p_r\big)\right\}$$

$$= P_o \nabla_R E\big[\big(1 + R(Y)\big) \log\big(1 + R(Y)\big)\big]$$

- Conclusions:
  - GLGD is really a gradient descent algorithm for the cost function

$$C(R) = 2\, KL\big((P_R + P_r)/2 || P_r\big)$$

  - $C(R)$ has a unique global minimum at $P_R = P_r$
  - However, convergence tends to be slow.

*Martin Arjovsky and Leon Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", ICLR 2017.

Some Important Functions

- Kullback-Leibler Divergence

$$KL(p_a||p_b) = \int_{\Re^N} -p_a(x) \log \frac{p_b(x)}{p_a(x)} dx$$
$$= \mathrm{E_a}\left[-\log \frac{p_b(x)}{p_a(x)}\right]$$
$$= \mathrm{E_a}[-\log p_b(x) + \log p_a(x)]$$

- Jensen-Shannon Divergence

$$JSD(p_a||p_b) = \frac{1}{2} KL(p_a||\bar{p}) + \frac{1}{2} KL(p_b||\bar{p})$$

where

$$\bar{p}(y) = \frac{1}{2}\left(p_a(y) + p_b(y)\right)$$

A Useful Relationships

- For $p_R(y) = R(y)p_r(y)$, with $Y \sim p_r(y)$ and $\tilde{Y}_R \sim p_R(y)$, then

$$KL(p_R||p_r) = E\left[-\log\left(R\left(\tilde{Y}_R\right)\right)\right] = E\left[-R(Y)\log\left(R(Y)\right)\right]$$

and

$$2\,KL\left((p_R + p_r)/2||p_r\right) = E\left[\left(1 + R(Y)\right)\log\left(1 + R(Y)\right)\right]$$

- For p_1 absolutely continuous with respect to p_2, then

$$KL(\bar{p}||p_2) = KL(p_1||p_2) - 2\,JSD(p_1||p_2)$$

where $\bar{p} = \frac{1}{2}(p_1 + p_2)$.

- Proof:

$$\begin{aligned}
KL(p_1||p_2) - 2\,JSD(p_1||p_2) &= KL(p_1||p_2) - KL(p_1||\bar{p}) - KL(p_2||\bar{p}) \\
&= \int p_1 \log\left(\frac{p_1}{p_2}\right) - p_1 \log\left(\frac{p_1}{\bar{p}}\right) - p_2 \log\left(\frac{p_2}{\bar{p}}\right) dy \\
&= \int p_1 \log\left(\frac{\bar{p}}{p_2}\right) - p_2 \log\left(\frac{p_2}{\bar{p}}\right) dy \\
&= \int -(p_1 + p_2) \log\left(\frac{p_2}{\bar{p}}\right) dy \\
&= 2\int -\bar{p} \log\left(\frac{p_2}{\bar{p}}\right) dy \\
&= 2KL(\bar{p}||p_2)
\end{aligned}$$

Convergence of GANs

- Generator and discriminator at convergence:
  - Reference and generated distribution are the same $\Rightarrow R^*(Y) = 1$
  - Discriminator can not distinguish distributions $\Rightarrow f^*(y) = \frac{1}{1+R(y)} = \frac{1}{2}$

- At convergence the generated and reference distributions are identical.
  - Therefore, the likelihood ratio is $R(y) = 1$;
  - The generated (fake) and reference (real) distributions are identical;
  - The discriminator assigns a 50/50 probability to either case because they are indistinguishable.
  - Then the generator are discriminator cross-entropy losses are

$$g(\theta_g^*, \theta_d^*) = E\left[-\log\left(f_{\theta_d}(\tilde{Y})\right)\right] = -\log\frac{1}{2} = \log 2 \approx 0.693$$

$$d(\theta_g^*, \theta_d) = E\left[-\log f_{\theta_d}(Y)\right] + E\left[-\log\left(1 - f_{\theta_d}(\tilde{Y})\right)\right] = -2\log\frac{1}{2} = 2\log 2 \approx 1.386$$

- In practice, things don't usually work out this well…

Algorithm 2: Practical Algorithm

- Algorithm

Repeat {

For N_d iterations {

$B \leftarrow GetRandomBatch$

$\theta_d \leftarrow \theta_d - \beta \nabla_{\theta_d} d(\theta_g, \theta_d; B)$

}

$B \leftarrow GetRandomBatch$

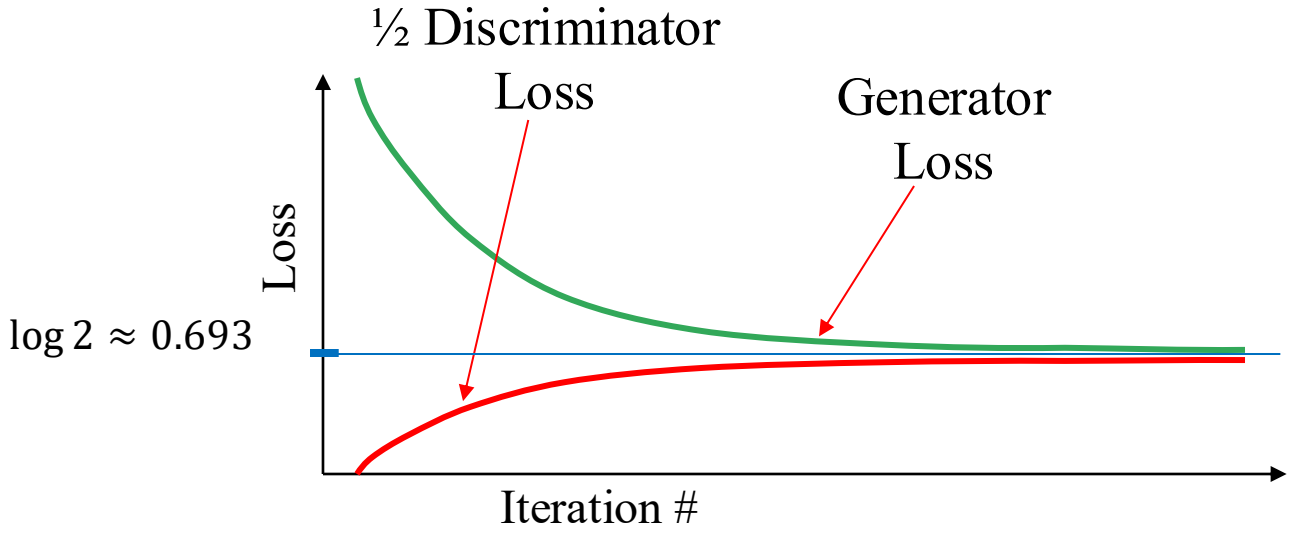
$\theta_g \leftarrow \theta_g - \alpha \nabla_{\theta_g} g(\theta_g, \theta_d; B)$

}

- What you would like to see

Looks good, but…

– Could result from a discriminator with insufficient capacity



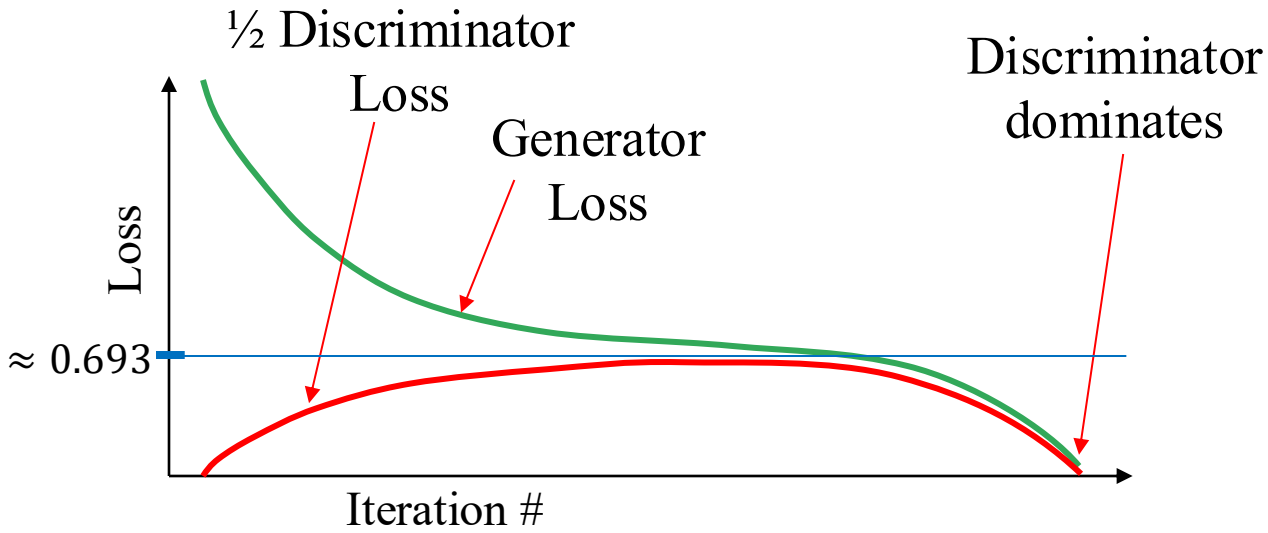
Failure Mode: Mode Collapse

- Algorithm

Repeat {
  For $i = 0$ to $N_d - 1$ {
    $\theta_d \leftarrow \theta_d - \beta \nabla_{\theta_d} d(\theta_g, \theta_d)$
  }
  $\theta_g \leftarrow \theta_g - \alpha \nabla_{\theta_g} g(\theta_g, \theta_d)$
}

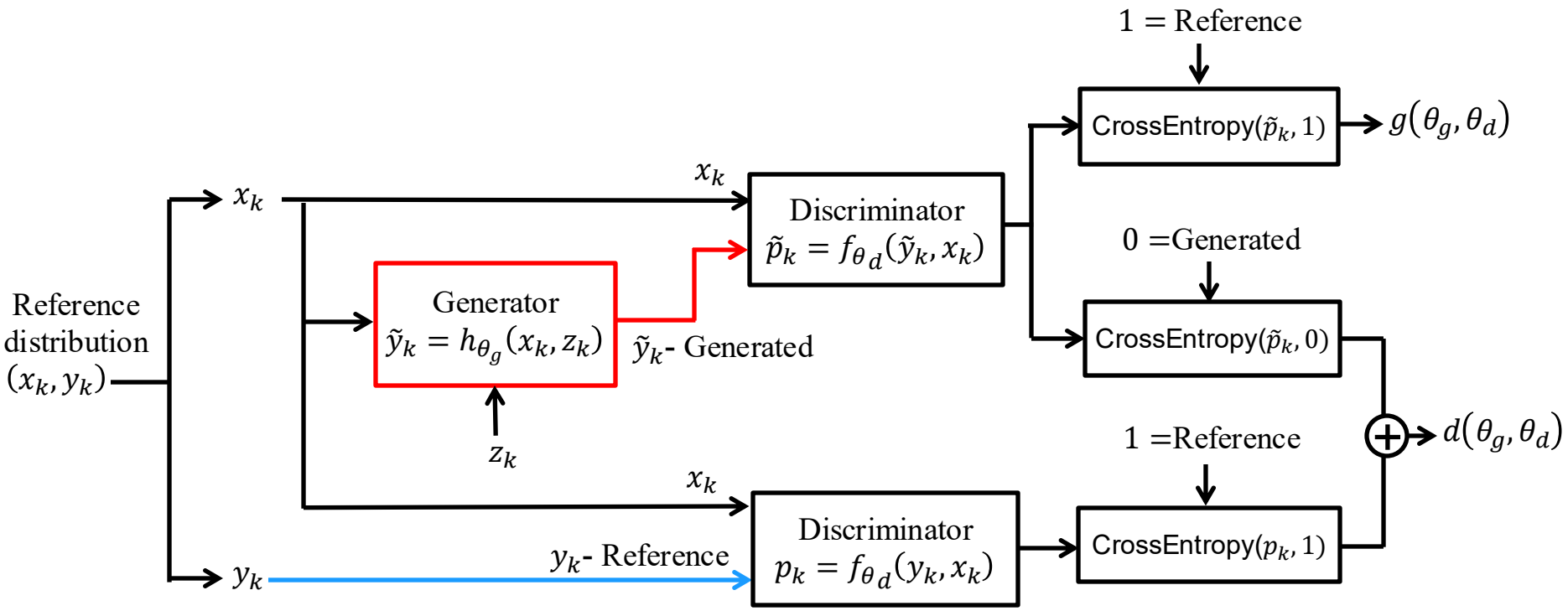
Might be caused by:
- Overfitting by discriminator
- Insufficient number of discriminator updates
- Insufficient generator capacity

- Sometimes you get mode collapse



Conditional Generative Adversarial Network

- Generates samples from the conditional distribution of Y given X.



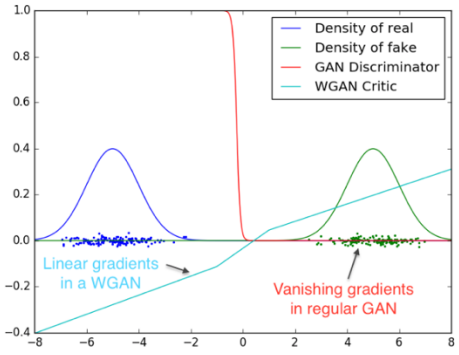
- Discriminator takes (y_k, x_k) input pairs for $k = 0, \cdots, K-1$

Advanced GAN Methods

- Wasserstein GANs
- Conditional GANs

Concept Wasserstein GAN

- Problem with GAN training using " $-\log D$ trick"
  - Slow and sometimes unstable convergence
  - Problems with vanishing gradient
- Conjecture:
  - The problem is caused by the discriminator function.
  - Bayes classifier is
    - Too sensitive and too nonlinear
    - Non-overlapping distributions create vanishing gradients that slow convergence.
- Base discriminator of the Wasserstein distance (i.e., earth mover distance)



*Reproduced from paper

Figure 3: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the traditional GAN discriminator saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

*Martin Arjovsky, Soumith Chintala and Leon Bottou, "Wasserstein Generative Adversarial Networks", ICML 2017.

Wasserstein Metric

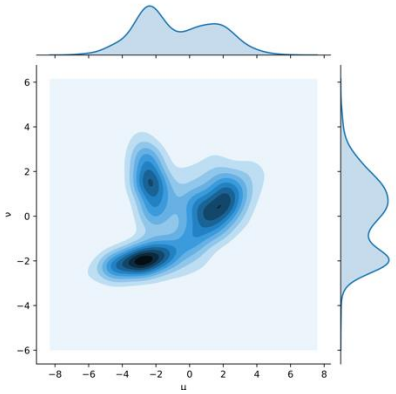
- Defined as

$$W(p_r||p_R) = \inf_{\gamma \in \Gamma} \int \|x - y\| \, \gamma(x, y) \, dxdy$$

where Γ is the set of all joint distributions, $\gamma(x, y)$, such that

$$p_r(x) = \int \gamma(x, y) \, dy$$

$$p_R(x) = \int \gamma(x, y) \, dx$$



*Reproduced from Wikipedia.

Wasserstein Fundamentals

- Based on Kantorovich-Rubinstein duality (Villani, 2009)*

$$W(p_r||p_R) = \sup_{\|f\|_L \leq 1} \left\{ E[f(Y)] - E\left[f\left(\tilde{Y}_R\right)\right] \right\}$$

  - where $\|f\|_L$ is the Lipschitz constant of f
  - $\|f\|_L \leq 1$ is referred to as the 1-Lipschitz condition

*Villani, Cedric. Optimal Transport: Old and New. Grundlehren der mathematischen Wissenschaften. Springer, Berlin, 2009.

Wasserstein GAN*

- Then the fundamental result of the Arjovsky and Bottou
  - Define the sets $R \in \Omega_g$ as usual, but define and $f \in \Omega_g$ so that

$$\Omega_d^W = \{f : \Re^N \to (-\infty, \infty)\ s.t. \|f\|_L \leq 1\}$$

  - Then define Wasserstein generator and discriminator loss functions as

$$g_{\mathrm{w}}(R, f) = E\left[-f\left(\tilde{Y}_R\right)\right]$$

$$d_{\mathrm{w}}(R, f) = E\left[f\left(\tilde{Y}_R\right)\right] - E[f(Y_R)]$$

- Key result from Arjovsky paper*:

$$\nabla_R g_{\mathrm{w}}(R, f^*) = \nabla_R W(p_r || p_R)$$

  - where

$$f^* = \arg\min_{f \in \Omega_d^W} d_w(R, f)$$

*Martin Arjovsky, Soumith Chintala, and Leon Bottou, "Wasserstein Generative Adversarial Networks", ICML 2017.

Method 3: Wasserstein Algorithm

- Algorithm

$$\text{Repeat } \{$$
$$f^* = \arg\min_{f \in \Omega_d^W} d_w(R, f)$$
$$R \leftarrow R - \alpha P_o \nabla_R g_w(R, f^*)$$
$$\}$$

- Discriminator update
  - How do we solve the problem of minimizing the discriminator loss with the Lipschitz constraint?
  - Answer: We clip the discriminator weights during training.
  - Observation: Isn't this just regularization of the discriminator DNN??
- Generator update
  - Take a step in the negative direction of the generator loss gradient descent.

*Martin Arjovsky and Leon Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", ICLR 2017.

Method 3: Wasserstein Practical Algorithm

- 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.

1: **while** θ 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 priors.
5: $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
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**

Iterate discriminator to approximate convergence

Make sure to get new batches

Minimize discriminator loss

Clip discriminator weights to approximate Lipschitz constraint

Take gradient step of generator loss

- Observations:
  - Some people seem to feel the Wasserstein GAN has better convergence.
  - However, is this because of the Wasserstein metric?
  - Or is it because of the other algorithmic improvements?

*Martin Arjovsky and Leon Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", ICLR 2017.