Generative Adversarial Networks

Benno Geißelmann

12. August 2018

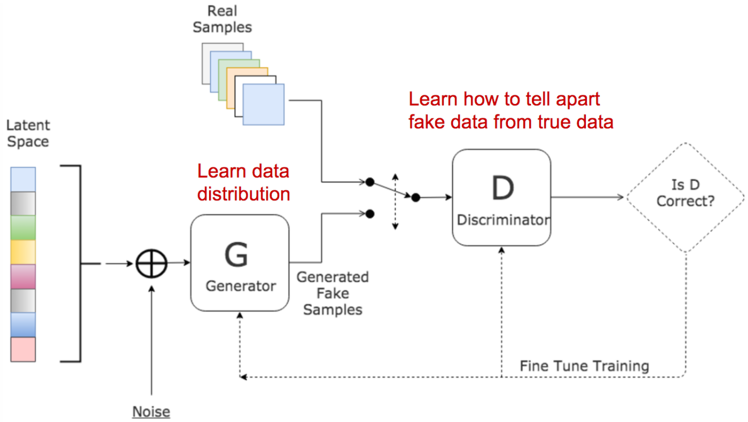
Contents

Example

What does this look like?

Example

General Architecture of GANs



The minimax problem

To retrieve a suitable generator network and a suitable discriminator network, the following minimax problem needs to be solved:

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

with
$D(x)$ = The discrimiator network
$G(x)$ = The generator network
$p_r(x)$ = The distribution of the real data
$p_g(x)$ = The distribution of the generated data
$p_z(z)$ = The distribution of a random noise variable
$\mathbb{E}_{x\sim P}[f(x)] = \sum_x P(x)f(x)$

Approximating a solution for GANs

Algorithm 1: Gradient Descent for GAN

for *it in interations* **do**

$nd \leftarrow \{z^{(1)}, \cdots, z^{(m)}\} \sim p_z(z)$

$rd \leftarrow \{x^{(1)}, \cdots, x^{(m)}\} \sim p_r(x)$

$g_{w_d} \leftarrow \nabla w_d \frac{1}{m} \sum_{i=1}^{m} [\log D(rd^{(i)}) + \log(1 - D(G(nd^{(i)})))]$

$w_d \leftarrow w_d + \eta g_{w_d}$

$nd \leftarrow \{z^{(1)}, \cdots, z^{(m)}\} \sim p_z(z)$

$g_{w_g} \leftarrow \nabla w_g \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(nd^{(i)})))]$

$w_g \leftarrow w_g - \eta g_{w_g}$

end

Convergence is not guaranteed

- In this non cooperative game, convergence of the two networks is not guaranteed
- It is non cooperative because the gradients are calculated independently
- Oscillation and instability during learning are common
- Possible Solution: Add Penalty term to loss function (historical averaging) which penalizes a high fluctuation of the networks parameters θ

Low dimensionality problem

- In reality $p_r(x)$ is concentrating on a small subset of a possible high dimensional event space
- At the same time $p_g(x)$ is initialized using some low dimensional noise data and is therefore also small
- There can always be found a suitable discriminator $D(x)$

Vanishing gradient problem

- If we have a very good discriminator $D(x)$ this means $D(G(z)) = 0 \quad \forall z \sim p_z(z)$
- At the same time $D(x) = 1 \quad \forall x \sim p_r(r)$
- As we then have no gradient which we can minimize for the generator, we cannot learn
- Possible solution: Add Noise to input of discriminator to artificially enlarge its "known"distribution

Mode collapse

- It can happen that the generator outputs always the same sample from $p_g(z)$
- We then end up in a small subset of the desired dirstibution $p_r(x)$
- Variety of the created samples is very low
- Possible solution: Show the discriminator a batch of outputs from the generator (Minibatch discrimination)

Wasserstein GANs

- Wasserstein GANs introduce a new way for measuring the distance (and therefore also a new loss function) between two distributions
- In words the Wasserstein-1 Metric defines how costly it is to transform a distribution $P_r(x)$ into another distribution $P_g(y)$ using an optimal transport plan
- Assuming that γ is this optimal transport plan where $\gamma(x, y)$ is the amount to transport from x to y, we can define the total cost as:

$$Cost = \sum_{x,y} \gamma(x, y)|x - y|$$

Wasserstein GANs

- So the Wasserstein-1 metric is defined as:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$$

- Also called the earth moovers distance
- $\Pi(P_r, P_g)$ can be seen as the set of all possible transport plans from P_r to P_g
- Wasserstein metric needs the optimal transport plan (greatest lower bound of these transport plans - the infimum)

Wasserstein GANs

- The Wasserstein-1 is hard to be used within the GAN learning process
- Therefore there is used an equivalent definition derived from Kantorovich-Rubinstein duality

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

- where f must be a 1-Lipschitz function.
- $f(x)$ can be seen as an instance of a parameterized family of functions $\{f_w(x)\}_{w \in W}$

Wasserstein GANs

- The discriminator now has the task to learn this function again as a neural network
- Actually the discriminator (or now called critic) has the aim to learn the Wasserstein-1 distance:

$$W(P_r, P_g) = \max_{w \in W} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{z \sim P_z}[f_w(G(z))]$$

- At the same time for a fixed f at time t the generator wants to miniminize $W(P_r, P_g)$ and does this by descending on $W(P_r, P_g)$

Wasserstein GANs

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