Statistic

for machine learning

Tran Trong Khiem

AI lab tranning

2024/05/29

Kullback–Leibler

KL (Kullback–Leibler) divergence measures how one probability distribution p diverges from a second expected probability distribution q:

$$D_{KL}(p||q) = \int_x p(x) ln(\frac{p(x)}{q(x)})$$

D_{KL} is min when q(x) = p(x) have : $D_{KL} = 0$

- KL divergence is asymmetric: $D_{KL}(q||p) <> D_{KL}(p||q)$
- if $p(x)$ is close to 0, $q(x)$ is not close to 0, $D_{KL}(p||q)$ almost not depend on $q(x)$ => **Bug**.

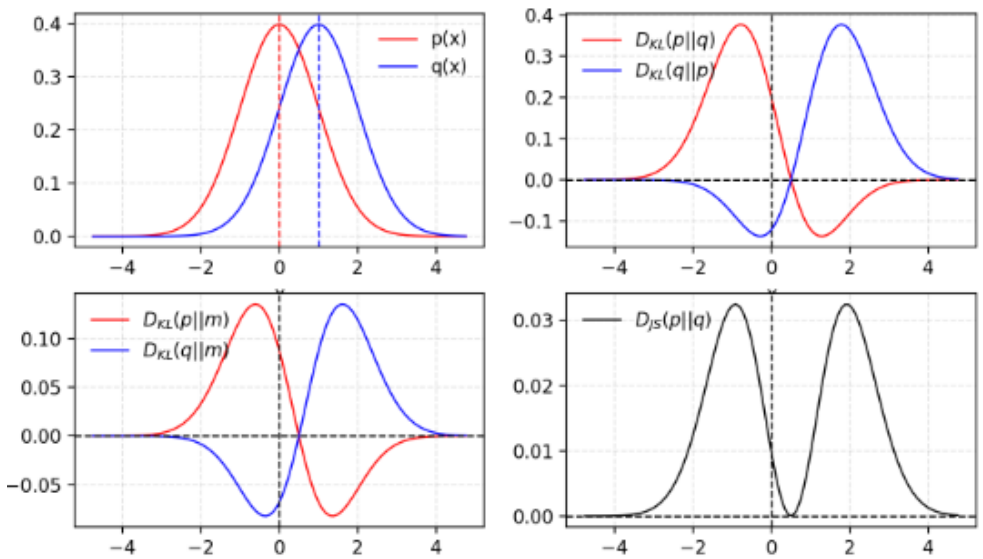
Jensen–Shannon Divergence

Another measure of similarity between two probability distribution :

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

- JS divergence is symmetric: $D_{JS}(q||p) = D_{JS}(p||q)$
- JS divergence value is in [0,1]

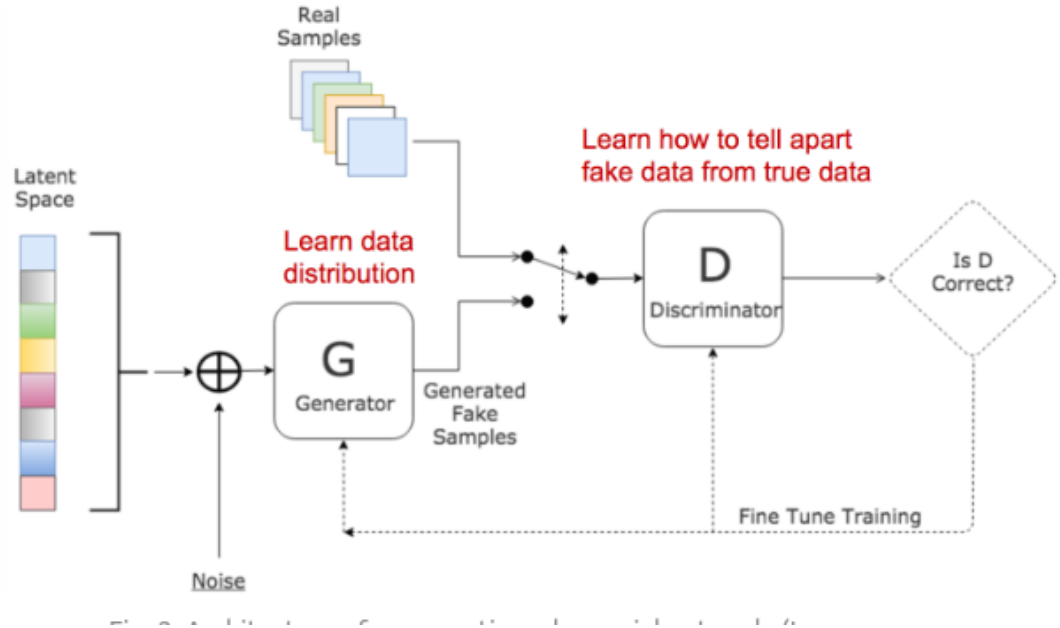
Set $m(x) = \frac{p(x)+q(x)}{2}$



Define

GAN consists of two models:

- A discriminator $\mathcal{D}$ estimates the probability of a given sample coming from the real dataset.It optimized to tell the fake samples from the real ones(0 for fake, 1 for real).
- A generator G, trained to capture the real data distribution and gen samples close as posible with real distribution.



Optimization

Given :

- p_z data distribution over noise z (usually uniform)
- p_g generator's distribution over data x
- p_r data distribution over real sample x

Model D is trying hard not to be cheated :

- Decisions over real data are accurate : maximizing $\mathbb{E}_{x\sim p_r(x)}[log(D(x))]$
- Detect fake sample gen by G : maximizing $\mathbb{E}_{x\sim p_g(x)}[log(1-D(x))]$

Model G is trying to fool D :

- Trained to increase the chances of D producing a high probability for a fake example : minimizing $\mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$

Combining together, we have **loss function**:

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

What is the optimal value for D?

Goal: examine the best value for D
We have **loss function** :

$$\mathcal{L}(G,D) = \int_x [p_r(x)log(D(x)) + p_g(x)log(1-D(x))]dx$$

We need to find $D_{op} = argmax_D \mathcal{L}(G,D)$
Set $\frac{\partial \mathcal{L}(G,D)}{\partial D} = 0$, we have :

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)} \in [0,1]$$

- generator is trained to its optimal, $p_r(x) \approx p_g(x) => D^*(x) \approx 0.5$ (Global optimal)
- Global optimal : $p_r(x) = p_g(x)$ and $D^*(x) = 1/2$, we have $\mathcal{L}(G^*,D^*) = -2log(2)$

Generator optimal

We have :

$$D_{JS}(p_r||p_g) = \frac{1}{2}D_{KL}(p_r||\frac{p_r + p_z}{2}) + \frac{1}{2}D_{KL}(p_g||\frac{p_g + p_r}{2})$$

$$= \frac{1}{2}(log(4) + \mathcal{L}(G, D^*))$$

Thus, $\mathcal{L}(G, D^*) = 2D_{JS}(p_r||p_g) - log(4)$

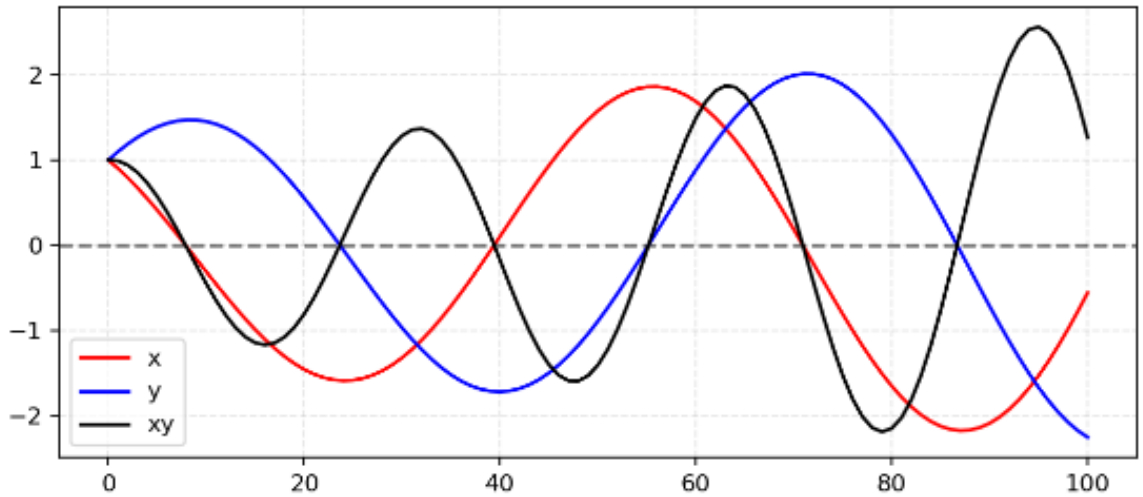
- GAN using D_{JS} to quantifies the similarity between p_r(real data distribution) and p_g(gen data distibution).
- There are many variations of GANs in different contexts or designed for different tasks.
- Depend on task, only need to save Generator to gen data.

Hard to achieve Nash equilibrium

Nash equilibrium : situation where no player could gain by changing their own strategy (**global optimal**)
Goal: Two model are trained to achive global opimal
Hard because :

- Each model updates its weight independently
- Updating the gradient of both models at the same time => cannot coveragence (need freeze D when train G)

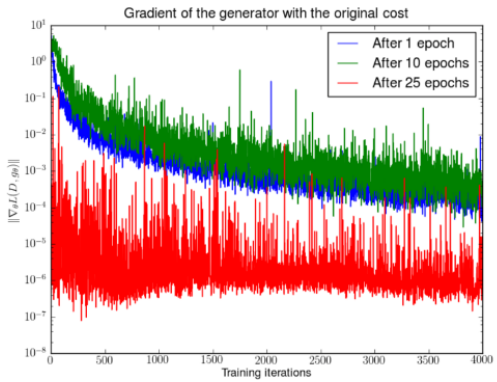
Example : player 1 minimize $f_1(x) = xy$, player 2 minimize $f_2(x) = -xy$(maximize f_1)



Vanishing gradient

When D is perfect :

- If $x \sim p_r$ then $D(x) = 1$
- If $x \sim p_g$ then $D(x) = 0$
- $=> \mathcal{L}(D, G) = 0$ then **vanishing gradient**



G is also in problem :

- **if D is bad** then G does'n have accuracy feedback $- >$ **cannot gen reality**
- **if D is very good** then gradient of loss close to zezo $- >$ **traning very slow**

Low dimensional supports

Define:

- **manifold**: n-manifold, is a topological space with the property that each point has a neighborhood that is same shape to an open subset of n-dimensional Euclidean space.
  - One-dimensional manifolds include lines and circles, but not self-crossing curves.
  - Two-dimensional manifolds are called surfaces
- p_r in **low dimention manifolds**
  - fundamental assumption for Manifold Learning
  - E.g:The images have a lot of restrictions to follow
- p_g lies in a **low dimensional manifolds**, too
- p_g and p_r are **almost gonna be disjoint**
  - bacause, p_g and p_r in low dimensional manifolds.
  - capable of finding a perfect D that separates real and fake samples 100%correctly.

Others

- **Mode collapse**
  - G only gen the same output
  - Can fool D, but fail to to learn to represent the complex real-world data distribution.

- **Lack of a proper evaluation metric**
  - No good sign to stop learning
  - No good indicator to compare the performance of multiple models

Improved GAN Training

- **Feature Matching**
  - Optimize D to check if G's output matches expected statistics of the real samples.
  - $\mathcal{L} = \left|\mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]\right|_2^2$, f(x) can be mod or mean.
- **Minibatch Discrimination**
  - D able uncover the relationship between training data points in one batch, instead of each point independently.
- **Historical Averaging**
  - For boths models, add to loss function $\left|\Theta - \frac{1}{t}\sum_t \Theta_t\right|^2$, where Θ_i is model parameter at i.
  - Help Θ changing smoothly.
- **One-sided Label Smoothing**
  - When feeding the discriminator, instead of providing 1 and 0 labels, use soften values such as 0.9 and 0.1.

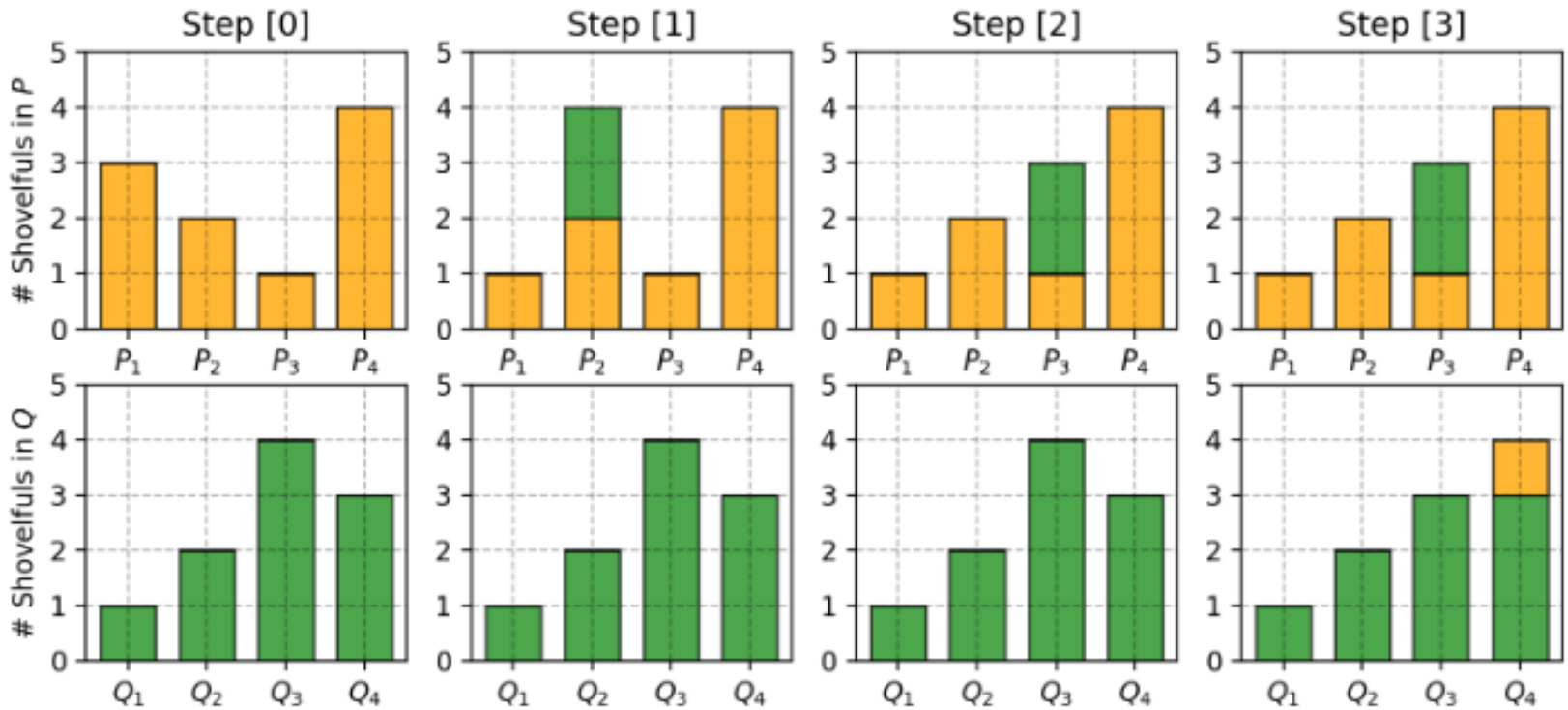
Improved GAN Training

- **Virtual Batch Normalization (VBN)**
  - Each data sample is normalized based on a fixed batch **("reference batch")**
  - Reference batch is chosen from begin and remain to the end of traing.
- **Adding Noises**
  - Add noise in input of D
  - Create higher chances for p_g and p_r to have overlaps.
- **Use Better Metric of Distribution Similarity**
  - use **Wasserstein** matrix

What is Wasserstein distance?

Wasserstein Distance is a measure of the distance between two probability distributions.

- **Discrete case**
  - δ_i is the cost make $P_i = Q_i$, we have : $\delta_i = \delta_{i-1} + P_i - Q_i$
  - Wasserstein distance : $W = \sum_i |\delta_i|$



What is Wasserstein distance?

- **Continous case**
  - We have
    - $II(p_r, p_g)$ is set of all possible joint probability distributions between p_r and p_g
    - $\gamma \in II(p_r, p_g)$ describes one dirt transport plan, (as discrete case)
    - $\gamma(x,y)$ is probability of dirt should move from x to y $->$ **make x follow same distribution as y**
    - $\sum_x \gamma(x,y) = p_g(y)$ and $\sum_y \gamma(x,y) = p_r(x)$
  - When moving from distribution of piont x to distribution of point y, total cost is given by :

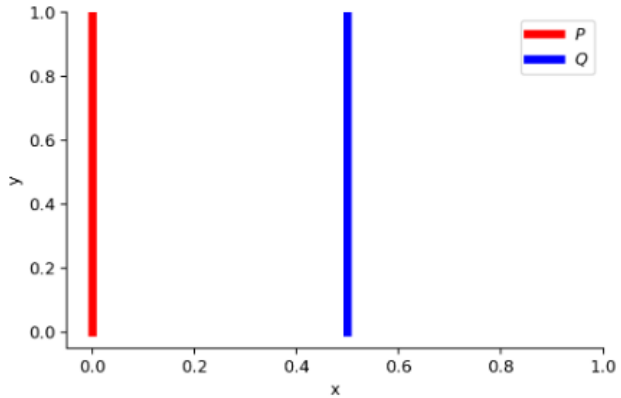
$$\sum_{x,y} \gamma(x,y)||x-y|| = \mathbb{E}_{x,y\sim\gamma}[||x-y||]$$

  - we take the **minimum one among the costs** of all dirt moving solutions as **Wasserstein distance:**

$$W(p_r, p_g) = \inf_{\gamma\sim II(p_r,p_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

Why Wasserstein(W) is better than JS or KL divergence?

- When **two distribution** are located in lower dimensional manifolds **not overlaps**
  - **W** still provide a meaningful and smooth representation
- Example : $(x,y) \in P, x = 0, y \sim U(0,1)$
  $(x,y) \in Q, x = \theta, y \sim U(0,1)$
- We have
  - W still smooth and meaningful
  - KL inifity when two distributions are disjoint
  - JS not depend at θ, and not differentiable at $\theta = 0$



Use Wasserstein distance as GAN loss function

- **Problem**:It is **intractable** to compute **all the possible joint distributions** in $II(p_r, p_g)$ to compute $\inf_{\gamma \sim II(p_r, p_g)}$
- **Lipschitz continuity?**
  - Denote $|f|_L \leq K$ meaning f is K-Lipschiz continous.
  - f :$\mathbb{R} \to \mathbb{R}$ is K-Lipschiz countinous if :

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

    where K is const, $K \geq 0$, $(x_1, x_2) \in \mathbb{R}$
  - Functions that are **everywhere continuously differentiable** is **Lipschitz continuous**,**otherwise not**
- We have: $W(p_r, p_g) = \frac{1}{K} \sup_{|f|_L \leq K} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$
  - Why have this transfomation ?

Use Wasserstein distance as GAN loss function

We have : $\mathcal{L}(p_r, p_g) = W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$

- f come from K-Lipschitz continous family
  - f_w parameterized by w
  - D learn to find w for good f_w
- D not direct tell what sample is fake, trained to learn f_w
- **Problem**: Keep f_w remain K-Lipschitz continous during traning.
  - After every gradient update, clamp the weights w in to small window.
- **WGan is not perfect**
  - Suffers from **unstable training**
  - If **clipping window is too large** then **slow convergence**
  - If **clipping window is too small** then **vanishing gradients**

Use Wasserstein distance as GAN loss function

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