目录

Reference

- TOWARDS PRINCIPLED METHODS FOR TRAINING GENERATIVE ADVERSARIAL NETWORKS
- Wasserstein Distance Guided Representation Learning for Domain Adaptation
- Wasserstein GAN
- Improved Training of Wasserstein GANs
- CML-Collaborative metric learning
- 令人拍案叫绝的Wasserstein GAN
- WGAN理论推导WGAN推导
- WDGRL阅读总结
- Collaborative Metric Learning

NOTE

GAN - Problems

GAN 判别器$D(x)$损失函数:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \tag{1}$$

其中P_r是真实样本分布，P_g是生成器产生的样本分布。

GAN 生成器的Loss Function中，Goodfellow提出两种损失函数：

$$\mathcal{L}_G = \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \tag{2}$$

$$\mathcal{L}_G = \mathbb{E}_{x\sim P_g}[-\log D(x)] \tag{3}$$

下面分别证明两种损失函数缺陷：

Problems with G(x) First Loss Function

我们假设生成器G固定，观察判别器D的最优下降情况

我们改写(1)式得：

$$\mathcal{L} = -P_r(x)\log D(x) - P_g(x)\log[1-D(x)] \tag{4}$$

令(4)关于$D(x)$的导数为0得：

$$-\frac{P_r(x)}{D(x)} + \frac{P_g(x)}{1-D(x)} = 0 \tag{5}$$

得条件下最优判别器：

$$D^*(x) = \frac{P_r(x)}{P_r(x)+P_g(x)} \tag{6}$$

直观理解下，如果只有真样本，则$P_g = 0$，判别器给出结果1，如果真假各占一半$P_g = P_r$，则判别器给出概率为0.5

假设我们拥有理想下的判别器，回过头观察关于生成器的第一种损失函数(2)，同时为(2)加上一项不依赖$G(x)$的项

$$\mathcal{L}_G = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \tag{7}$$

带入(6)并变化得：

$$\mathcal{L}_G = \mathbb{E}_{x\sim P_r}[\log \frac{P_r(x)}{\frac{1}{2}[P_r(x)+P_g(x)]}] + \mathbb{E}_{x\sim P_g}[\log \frac{P_g(x)}{\frac{1}{2}[P_r(x)+P_g(x)]}] - 2\log 2 \tag{8}$$

我们知道$Kullback–Leiblerdivergence(KL散度)$：

$$KL(P_1 \parallel P_2) = \mathbb{E}_{x\sim P_1}[\log \frac{P_1}{P_2}] \tag{9}$$

我们也知道$Jensen-Shannondivergence(JS散度)$

$$JS(P_1 \parallel P_2) = \frac{1}{2}KL(P_1 \parallel \frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2 \parallel \frac{P_1+P_2}{2}) \tag{10}$$

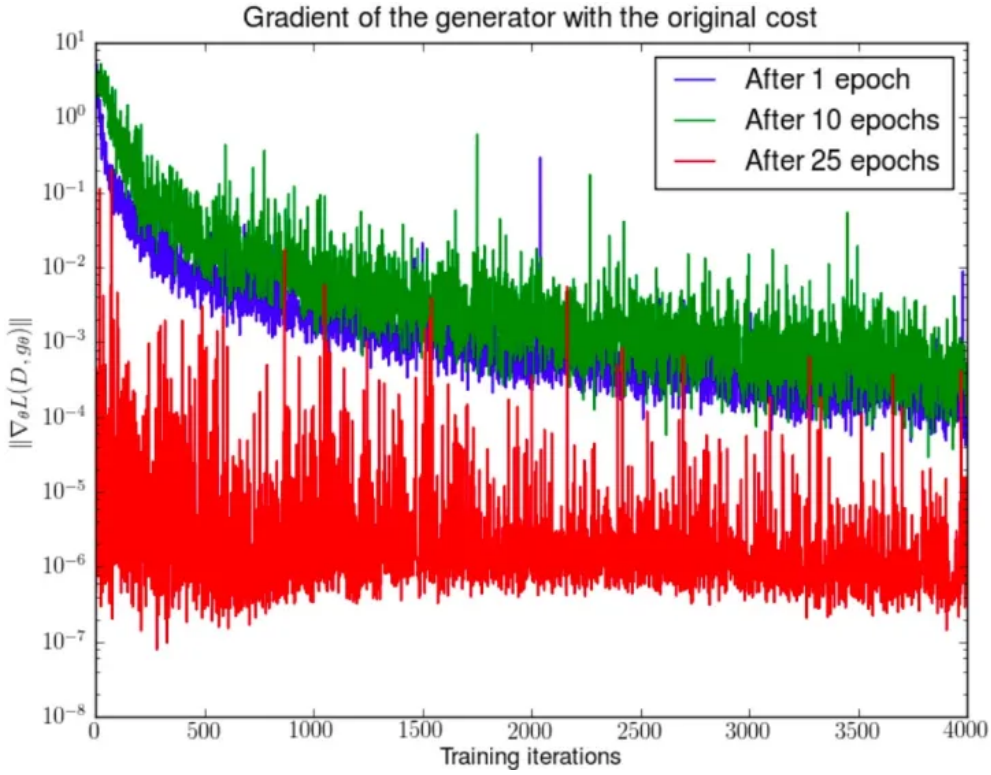
于是将$(9)(10)$带入(8)得

$$\mathcal{L}_G = 2JS(P_r \parallel P_g) - 2\log 2 \tag{11}$$

因此当真实数据分布与生成器分布在完全没有重合或**极少重合**的情况下，损失函数降为常数，意味着梯度为0.

而极少重合的概率是非常大的。准确描述为：因为当P_r与P_g的支撑集是高维空间的低维流形时，二者重叠部分测度为0的概率近似为1.

上句换言之：GAN的$G(x)$一般由低维向量信息向高维向量变换，当生成器固定时，所生成的高维向量很可能由低维向量的分布来固定，本质还是低维向量(同时也得考虑映射降维)，综上生成器难以撑起高维空间信息，最终仍在低维展示。同时试想三维空间中，若两平面相交为直线，在维度坍缩下线在三维中体积可以忽略。因此我们可以理解成$\mathcal{L}_G$极可能变成常数导致训练过程梯度消失



> 实验验证：作者在正常训练1、10、25轮后固定生成器不动，判别器随机从头训练，结果是生成器的$loss$迅速衰减(y轴是对数坐标轴)

Problems with G(x) Second Loss Function

下面将对(3)的合理性验证

由KL的定义(9)得

$$
\begin{aligned}
KL\left(P_g \| P_r\right) &= \mathbb{E}_{x \sim P_g}\left[\log \frac{P_g(x)}{P_r(x)}\right] \\
&= \mathbb{E}_{x \sim P_g}\left[\log \frac{P_g(x) / \left(P_r(x)+P_g(x)\right)}{P_r(x) / \left(P_r(x)+P_g(x)\right)}\right] \\
&= \mathbb{E}_{x \sim P_g}\left[\log \frac{1-D^*(x)}{D^*(x)}\right] \\
&= \mathbb{E}_{x \sim P_g} \log \left[1-D^*(x)\right] - \mathbb{E}_{x \sim P_g} \log D^*(x)
\end{aligned} \tag{12}
$$

由$(3)(11)(12)$得

$$\begin{aligned}\mathbb{E}_{x\sim P_g}\left[-\log D^*(x)\right] &= KL\left(P_g\|P_r\right) - \mathbb{E}_{x\sim P_g}\log\left[1-D^*(x)\right] \\ &= KL\left(P_g\|P_r\right) - 2JS\left(P_r\|P_g\right) + 2\log 2 + \mathbb{E}_{x\sim P_r}\left[\log D^*(x)\right]\end{aligned}$$

观察(13)发现后两项与生成器$G(x)$无关，因此损失函数进一步等价为：

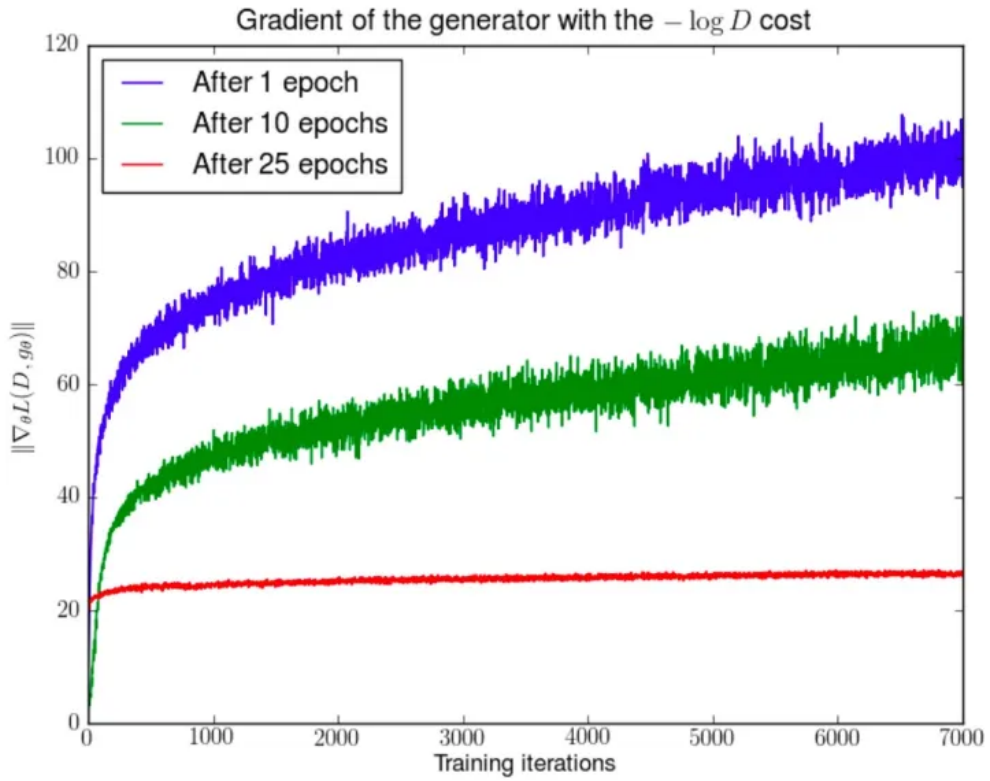
$$\mathbb{E}_{x\sim P_g}[-\log D^*(x)] = KL(P_g \parallel P_r) - 2JS(P_g \parallel P_r) \tag{14}$$

至此可以发现如果最小化损失函数，既要最小化KL散度，又要最大化JS散度，既要拉近又要推远，是直觉上非常不合理的。

同时KL散度是一个非对称函数，即$KL(P_g \parallel P_r) \neq KL(P_r \parallel P_g)$，则如下情况：

- 当$P_g(x) \to 0$而$P_r(x) \to 1$时，$KL(P_g \parallel P_r) = P_g(x)\log\frac{P_g(x)}{P_r(x)} \to 0$，意味着$loss$更低
- 当$P_g(x) \to 1$而$P_r(x) \to 0$时，$KL(P_g \parallel P_r) = P_g(x)\log\frac{P_g(x)}{P_r(x)} \to +\infty$，意味着$loss$极大

通俗解释上述影响为：生成器将更愿意生成绝对正确的“安全”样本，导致多样性不足($collapse-mode$)



> 实验验证：同样将预训练1、10、25轮的模型将判别器重新随机分布再进行训练，观察得到梯度震荡，红线为DCGAN相对稳定的收敛状态。

Wasserstein Distance

What's Wasserstein Distance

$Wasserstein$距离又称推土机距离，公式如下：

$$\mathcal{W}_{[P,Q]} = \inf_{\gamma\in\prod_{[P,Q]}} \iint \gamma(x,y)d(x,y)dxdy \tag{15}$$

$\gamma \sim \Pi[P,Q]$ 表示

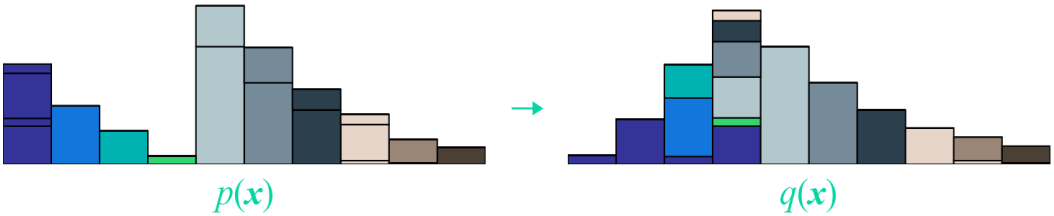
其中$\gamma \in \prod_{[p,q]}$表示γ服从$[P,Q]$的联合分布，P,Q是γ的边缘分布，因此有如下关系

$$\int \gamma(x,y)dy = P(x) \tag{16}$$

$$\int \gamma(x,y)dx = Q(y) \tag{17}$$

下面对(15)进行简单解释

- $\inf$：表示下确界，简单理解为可以得到的最小值
- $d(x,y)$： 表示分布p和q的距离，最常用有各种范数，余弦距离等
- 以推土机来比喻，就是将一个分布推至另一个分布所做的工(下图)，其中$\int \gamma(x,y)dx = Q(y)$理解为从$x$处搬运$\gamma(x,y)dx$到$y$处所需要的成本，可以理解成做工多少



> 图源于WGAN理论推导

因此将$d(x,y)$理解为做工距离，$\gamma(x,y)$理解为做工方式，最终$\mathcal{W}_{[p,q]}$就成为搬运最小成本

How to calculate Wasserstein Distance

重新明确一下我们目标：

$$\begin{aligned} &argmin \iint \gamma(x,y)d(x,y)dxdy \\ s.t. &\int \gamma(x,y)dy = P(x), \int \gamma(x,y)dx = Q(y), \gamma(x,y) \geqslant 0 \end{aligned} \tag{18}$$

我们将$\gamma(x,y)$与$d(x,y)$离散化得：

$$\boldsymbol{\Gamma} = \begin{pmatrix} \gamma\left(\boldsymbol{x}_1, \boldsymbol{y}_1\right) \\ \gamma\left(\boldsymbol{x}_1, \boldsymbol{y}_2\right) \\ \vdots \\ \hline \gamma\left(\boldsymbol{x}_2, \boldsymbol{y}_1\right) \\ \gamma\left(\boldsymbol{x}_2, \boldsymbol{y}_2\right) \\ \vdots \\ \hline \vdots \\ \hline \gamma\left(\boldsymbol{x}_n, \boldsymbol{y}_1\right) \\ \gamma\left(\boldsymbol{x}_n, \boldsymbol{y}_2\right) \\ \vdots \end{pmatrix}, \quad \boldsymbol{D} = \begin{pmatrix} d\left(\boldsymbol{x}_1, \boldsymbol{y}_1\right) \\ d\left(\boldsymbol{x}_1, \boldsymbol{y}_2\right) \\ \vdots \\ \hline d\left(\boldsymbol{x}_2, \boldsymbol{y}_1\right) \\ d\left(\boldsymbol{x}_2, \boldsymbol{y}_2\right) \\ \vdots \\ \hline \vdots \\ \hline d\left(\boldsymbol{x}_n, \boldsymbol{y}_1\right) \\ d\left(\boldsymbol{x}_n, \boldsymbol{y}_2\right) \\ \vdots \end{pmatrix} \tag{19}$$

$$\begin{pmatrix} \vdots \\ \gamma(x_n, y_n) \end{pmatrix} \qquad \begin{pmatrix} \vdots \\ d(x_n, y_n) \end{pmatrix}$$

用(19)改写(18)中目标函数内积形式得

$$argmin\langle \Gamma, D \rangle \tag{20}$$

同时改写(18)中的约束条件

$$\underbrace{\left(\begin{array}{ccc|ccc|c|ccc|c} 1 & 1 & \dots & 0 & 0 & \dots & \dots & 0 & 0 & \dots & \dots \\ 0 & 0 & \dots & 1 & 1 & \dots & \dots & 0 & 0 & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots \\ 0 & 0 & \dots & 0 & 0 & \dots & \dots & 1 & 1 & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots \\ \hline 1 & 0 & \dots & 1 & 0 & \dots & \dots & 1 & 0 & \dots & \dots \\ 0 & 1 & \dots & 0 & 1 & \dots & \dots & 0 & 1 & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots \\ 0 & 0 & \dots & 0 & 0 & \dots & \dots & 0 & 0 & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots \end{array}\right)}_{A} \underbrace{\left(\begin{array}{c} \gamma(\boldsymbol{x}_1, \boldsymbol{y}_1) \\ \gamma(\boldsymbol{x}_1, \boldsymbol{y}_2) \\ \vdots \\ \hline \gamma(\boldsymbol{x}_2, \boldsymbol{y}_1) \\ \gamma(\boldsymbol{x}_2, \boldsymbol{y}_2) \\ \vdots \\ \hline \vdots \\ \hline \gamma(\boldsymbol{x}_n, \boldsymbol{y}_1) \\ \gamma(\boldsymbol{x}_n, \boldsymbol{y}_2) \\ \vdots \\ \gamma(x_n, y_n) \end{array}\right)}_{\Gamma} = \underbrace{\left(\begin{array}{c} P(\boldsymbol{x}_1) \\ P(\boldsymbol{x}_2) \\ \vdots \\ \hline P(\boldsymbol{x}_n) \\ \hline Q(\boldsymbol{y}_1) \\ Q(\boldsymbol{y}_2) \\ \vdots \\ Q(\boldsymbol{y}_n) \end{array}\right)}_{b}$$

即

$$A\Gamma = b \tag{22}$$

由$(18)(20)(22)$得目标函数最终为:

$$\begin{aligned} &\underbrace{argmin}_{\Gamma}\langle \Gamma, D \rangle \\ &s.t.\ A\Gamma = b, \Gamma \geqslant 0 \end{aligned} \tag{23}$$

强对偶形式转换:

$$\underset{y}{max}\{b^T y \mid A^T y \leqslant c\} = \underset{x}{min}\{c^T x \mid Ax = b, x \geqslant 0\} \tag{24}$$

用(24)转换(23)得

$$\underset{\Gamma}{min}\{\langle \Gamma, D \rangle \mid A\Gamma = b, \Gamma \geqslant 0\} = \underset{F}{max}\{\langle b, F \rangle \mid A^T F \leqslant D\} \tag{25}$$

其中F作为新变量可以写作如下形式，因为上文b中为上下分块矩阵，因此拿f和g两个自变量表示:

$$F = \begin{pmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_n) \\ \hline g(\boldsymbol{y}_1) \\ g(\boldsymbol{y}_2) \end{pmatrix} \tag{26}$$

$$\begin{pmatrix} g(\boldsymbol{y}_2) \\ \vdots \\ g(\boldsymbol{y}_n) \end{pmatrix}$$

即（25）中max形式进一步写成：

注：由于 F 是由上下拼接组成，因此长度为 $2n$

$$\begin{aligned} \langle b, F\rangle &\Longleftrightarrow \sum_{2n}^{i} P(x_i)f(x_i) + \sum_{2n}^{i} Q(x_i)g(x_i) \\ A^T F \leqslant D &\Longleftrightarrow \forall x, y, f(x) + g(y) \leqslant d(x, y) \end{aligned} \tag{27}$$

Wasserstein Distance即可写成：

$$\mathcal{W}[P, Q] = \max_{f,g} \left\{ \int [P(\boldsymbol{x})f(\boldsymbol{x}) + Q(\boldsymbol{x})g(\boldsymbol{x})]d\boldsymbol{x} \mid f(\boldsymbol{x}) + g(\boldsymbol{y}) \leqslant d(\boldsymbol{x}, \boldsymbol{y}) \right\} \tag{28}$$

由于：

$$f(x) + g(y) \leqslant d(x, y) \Rightarrow f(x) + g(x) \leqslant d(x, x) \Rightarrow g(x) \leqslant -f(x) \tag{29}$$

得：

$$\begin{aligned} p(\boldsymbol{x})f(\boldsymbol{x}) + q(\boldsymbol{x})g(\boldsymbol{x}) &\leqslant p(\boldsymbol{x})f(\boldsymbol{x}) + q(\boldsymbol{x})[-f(\boldsymbol{x})] \\ &= p(\boldsymbol{x})f(\boldsymbol{x}) - q(\boldsymbol{x})f(\boldsymbol{x}) \end{aligned} \tag{30}$$

因此从(30)中如果$g = -f$，Wasserstein Distance条件仍然成立

由$(28)(30)$得

$$\mathcal{W}[P, Q] = \max_{f,g} \left\{ \int [P(\boldsymbol{x})f(\boldsymbol{x}) - Q(\boldsymbol{x})f(\boldsymbol{x})]d\boldsymbol{x} \mid f(\boldsymbol{x}) + g(\boldsymbol{y}) \leqslant d(\boldsymbol{x}, \boldsymbol{y}) \right\} \tag{31}$$

同时也能写成期望形式，利用极大似然估计拟合真实分布

$$\mathcal{W}[P, Q] = \max_{f, \|f\|_L \leqslant K} \mathbb{E}_{x \sim P(x)}[f(x)] - \mathbb{E}_{x \sim Q(x)}[f(x)] \tag{32}$$

至此结束Wasserstein Distance推导

WGAN

即将(32)用在GAN的损失函数中

考虑到(31)中存在约束条件，我们入Lipschitz常数，如下：

> Lipschitz连续，即连续函数f要求在定义域上满足任意两个元素x_1, x_2满足：
>
> $$| f(x_1) - f(x_2) | \leqslant K | x_1 - x_2 | \tag{33}$$
>
> 此时称K为Lipschitz常数

更进一步的，我们可以理解成，只要函数f的导数存在，即满足(31)式子约束条件

在深度学习中我们可以将f看成带参数w的神经层，通过不断学习拟合函数

考虑到梯度爆炸的可能会破坏Lipschitz连续，因此训练过程需要每次更新完w后$clip$回某个阈值即可，原文中设置为：$w_i \in [-0.01, 0.01]$

最后损失函数改进成

判别器

$$\mathcal{L}_D = \mathbb{E}_{x\sim P_g}[f_w(x)] - \mathbb{E}_{x\sim P_r}[f_w(x)] \tag{34}$$

生成器

$$\mathcal{L}_G = -\mathbb{E}_{x\sim P_g}[f_w(x)] \tag{35}$$

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

> 原论文算法流程

WGAN-GP

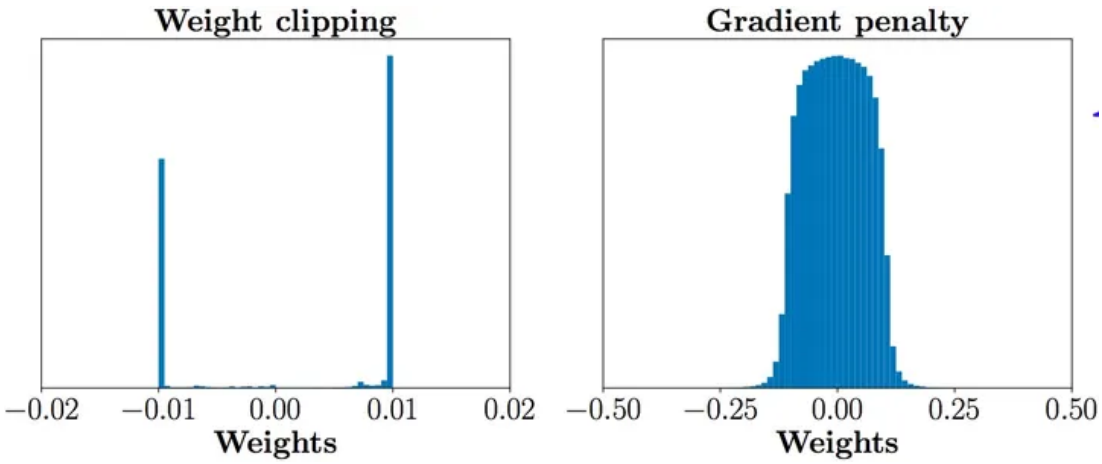
我们以(33)强硬的约束条件，不出意外的出意外了

为了区分原方法与新方法，我们称原方法为Weight clipping，改进方案为Gradient penalty

WGAN's Problem

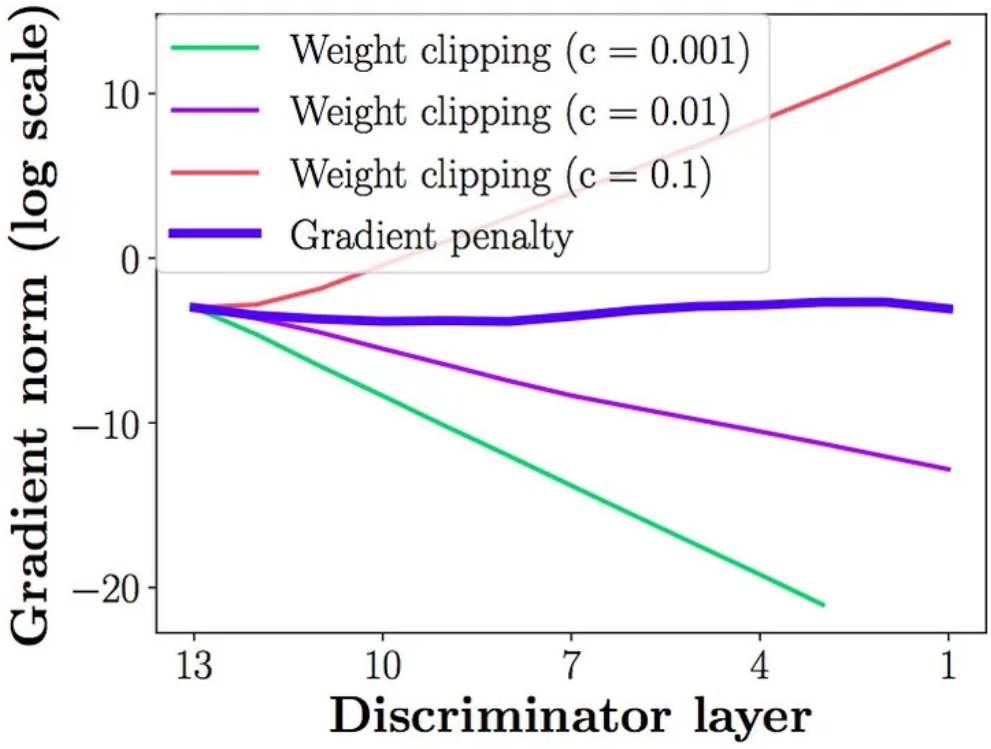
问题一：WGAN独立限制f的参数取值(clipping回某范围内)，导致参数取值极端，即要么取最大值，要么取最小值

> 对此我个人理解是
>
> $f(x)$的本质还是二分类判别器，为了将正负样本拉开，最优决策还是采取二值化
>
> 为此作者实验出如下图结果，展示f的分布



因此大大浪费神经网络参数拟合

问题二：weight clipping很容易导致梯度消失、梯度爆炸，f试一个多层网络，如果clipping threshold设置较低，每经过一层就会被夹一次，变得指数衰减，如果设置得较大，又会变成指数爆炸，为此调参工作繁杂



> 作者实验指出随着层数增加梯度消失，c表示clipping值

Solution - Gradient penalty

我们不在强硬clipping，而是将条件放回Loss函数中，毕竟(31)中的条件本质上是避免出现不存在梯度的软限制。

$$\mathcal{L}_f = [\| \nabla f(x) \|_p - K]^2 \tag{36}$$

但是作者补充证明(33)中最优梯度为1

> 简单证明

> **Proposition 1.** *Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distributions in $\mathcal{X}$, a compact metric space. Then, there is a 1-Lipschitz function f^* which is the optimal solution of $\max_{\|f\|_L \leq 1} \mathbb{E}_{y \sim \mathbb{P}_r}[f(y)] - \mathbb{E}_{x \sim \mathbb{P}_g}[f(x)]$. Let π be the optimal coupling between $\mathbb{P}_r$ and $\mathbb{P}_g$, defined as the minimizer of: $W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\pi \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\pi}[\|x - y\|]$ where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of joint distributions $\pi(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively. Then, if f^* is differentiable[‡], $\pi(x = y) = 0$[§], and $x_t = tx + (1-t)y$ with $0 \leq t \leq 1$, it holds that $\mathbb{P}_{(x,y)\sim\pi}\left[\nabla f^*(x_t) = \frac{y - x_t}{\|y - x_t\|}\right] = 1$.*
> **Corollary 1.** *f^* has gradient norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$.*

Improved Training of Wasserstein GANs, page 3

得总损失函数为

$$\mathcal{L}_D = \mathbb{E}_{x\sim P_g}[f_w(x)] - \mathbb{E}_{x\sim P_r}[f_w(x)] + \lambda \mathbb{E}_{x\sim\chi}[\| \nabla f(x) \|_p - 1]^2 \quad (37)$$

观察损失项，前二者可以通过采样简单获取，但第三个要求在整个样本空间中χ采样，很显然无法实现。

但回过头思考什么位置的梯度可能最大，答：

$$由A分布 \rightarrow B分布直线路径上产生$$

因此我们只需要调整关键位置即可，更具体的说：我们随机采取真假样本如下

$$x_r \sim P_r, x_g \sim P_g, \epsilon \sim Uniform[0, 1] \quad (38)$$

所两分布连线上随机插值

$$\hat{x} = \epsilon x_r + (1 - \epsilon) x_g \quad (39)$$

联立$(37)(38)(39)$得

$$\mathcal{L}_D = \mathbb{E}_{x\sim P_g}[f_w(x)] - \mathbb{E}_{x\sim P_r}[f_w(x)] + \lambda \mathbb{E}_{x\sim P_{\hat{x}}}[\| \nabla f(x) \|_p - 1]^2 \quad (40)$$

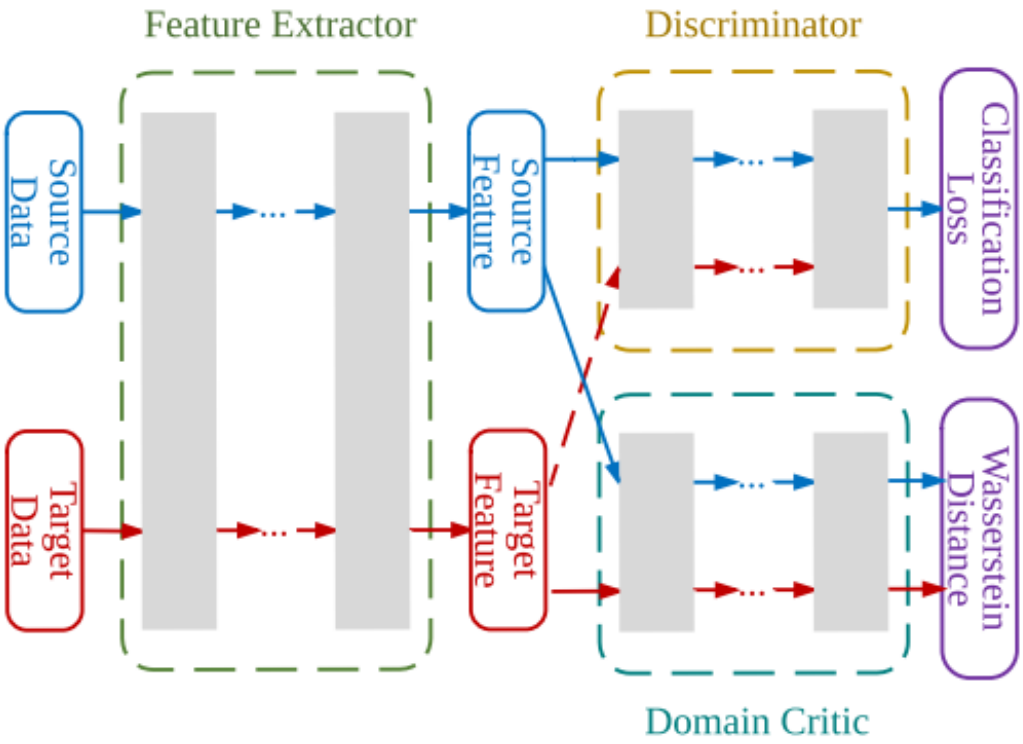
至此为全部的WGAN-GP

[WDGRL] Wasserstein Distance Guided Representation Learning for Domain Adaptation

What's WDGRL

分为源域与目标域，目标是做分类任务，通过学习变换分布的方式将目标域的分布转换成源域相符特征分布，以此套用源域方式进行分类任务

本质还是套了上文WGAN-GP



> WDGRL模型框架

WDGRL Loss Function

我们尽量以论文中定义符号：

$$\begin{gathered} x_s, x_t\text{分别表示源域与目标域的特征} \\ f_g(x)\text{表示框架图中}FeatureExtractor, \\ f_w(x)\text{表示}WGAN\text{所提及的神经层用于计算}WassersteinDistance \end{gathered} \tag{41}$$

第一项损失函数为Domain Critic中的Wasserstein Distance，由$(32)(41)$得

$$\mathcal{L}_{wd} = \frac{1}{n^s}\sum_{x^s \in \chi^s} f_w(f_g(x^s)) - \frac{1}{n^t}\sum_{x^t \in \chi^t} f_w(f_g(x^t)) \tag{42}$$

第二项损失函数为WGAN-GP提出来关于f_w的损失函数，由(36)得

$$\begin{gathered} \mathcal{L}_{grad}(\hat{h}) = [\| \nabla f(\hat{h}) \|_p - 1]^2 \\ \hat{y} \leftarrow sample\_in\_line\_between\{h^s, h^t\} \end{gathered} \tag{43}$$

本质上框架还是个分类器，故加上第三项关于Discriminator的损失函数

$$\mathcal{L}_c(x^s, y^s) = -\frac{1}{n^s}\sum_{i=1}^{n^s}\sum_{k=1}^{l} 1(y_i^s == k) * \log f_c(f_g(x_i^s))_k \tag{44}$$

最后联立$(42)(43)(44)$，汇总得到的目标函数为

$$\min_{\theta_g, \theta_c}\{\mathcal{L}_c + \lambda \max_{\theta_w}[\mathcal{L}_{wd} - \gamma \mathcal{L}_{grad}]\} \tag{45}$$

综上为WDGRL

[CML]Collaborative Metric Learning

前言：基于矩阵分解的模型使用product dot度量user vector和item vector的方式不满足三角不等式，导致MF模型无法捕获用户的细粒度偏好，作者希望通过Large-Margin Nearest Neighbor（LMNN）中提到的"异类远离"的思想，即一个用户所喜欢的商品要远离这个用户不喜欢的商品，并且该距离也会被一个与Rank（物品排序）有关的权重控制。最终实现提升

BACKGROUND

- 度量学习

比较两个对象之间的相似程度，通常记为$d(x,y)$，尽量满足以下条件

1. $d(x,x)=0$
2. $d(x,y)>=0$
3. $d(x,y)=d(y,x)$
4. $d(x,k)+d(k,y)>=d(x,y)$

常见相似度度量方法有欧氏距离、曼哈顿距离、切比雪夫距离、闵可夫斯基距离、**马氏距离**、夹角余弦、信息熵等

马氏距离计算如下：

$$d_M(x,y)=\sqrt{(x-y)^T\sum^{-1}(x-y)} \tag{46}$$
$$其中\sum为多维随机变量的协方差矩阵$$

- 常用的度量学习损失函数
  - 对比损失

$$\mathcal{L}=yd^2(x,y)+(1-y)(\alpha-d(x,y))_+^2 \tag{47}$$
$$其中y=1当且x,y为正样本对(同类)，反之为0$$
$$\alpha表示负样本对被推开的安全距离$$

  - 三元组损失（Triplet loss）

$$\mathcal{L}=(d(a,p)-d(a,n)+\alpha)_+ \tag{48}$$
$$选定三元组，其中固定点a(Anchor)、正样本p(Positive)和负样本n(Negative)$$

  - 改进三元组损失（Improved triplet loss）

    原三元组损失中未考虑正样本之间绝对距离，使得正样本没有汇聚趋向，因此改进方程如下

$$\mathcal{L}=\underline{d(a,p)}+(d(a,p)-d(a,n)+\alpha)_+ \tag{49}$$

LMNN

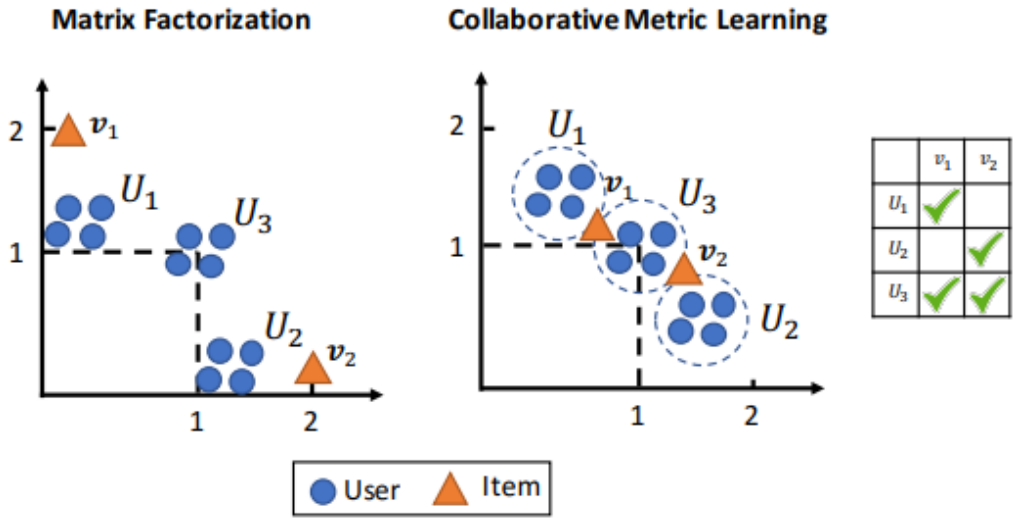
思想源于(48)得到以下损失函数

$$\mathcal{L}_{pull}(d) = \sum_{j \rightsquigarrow i} d(a,p)^2$$
$$\mathcal{L}_{push}(d) = \sum_{j \rightsquigarrow i} \sum_{k} (1 - y_{ik})[1 + d(a,p)^2 - d(a,n)^2]_+ \tag{50}$$
$$其中 j \rightsquigarrow i 表示 j 为 i 的 target - neighbor$$

即前者负责拉近相似用户，后者推开互斥用户

CML

- 简单解释为什么不满足三角不等式



> 上图左边可以观察到，user1与user3的特征相近，同理user3与user2相近，因此理论上user1也与user2相似，user1喜欢的物品v1应该距离user2近，可实际上却较远，正确的关系应为上图右边

- CML损失函数

我们先规定user向量、item向量、距离表示，以及集合属性如下:

$$\begin{aligned} u_i \in \mathcal{R}^r, v_i \in \mathcal{R}^r \\ d(i,j) = \| u_i - v_i \| \\ \mathcal{S} = \{(x_i, x_j) \mid x_i 与 x_j 相似\} \\ \mathcal{D} = \{(x_i, x_j) \mid x_i 与 x_j 不相似\} \end{aligned} \tag{51}$$

结合$LMNN$作者提出损失函数

Metric Loss Function

$$\mathcal{L}_m(d) = \sum_{(i,j) \in \mathcal{S}} \sum_{(i,k) \notin \mathcal{S}} w_{ij}[m + d(i,j)^2 - d(i,k)^2]_+ \tag{52}$$
$$m > 0 (安全距离)$$

补充：w_{ij}表示排名加权损失，原文称为Weighted Approximate-Rank Pairwise (WARP) loss

$$w_{ij} = \log(rank_d(i,j)+1) \\ \text{其中物品}j\text{被用户}i\text{所喜欢} \tag{53}$$

针对物品的特征嵌入函数需要通过损失函数微调，使得映射对应的正确位置

$$\mathcal{L}_f(\theta, v_*) = \sum_j \parallel f(x_j,\theta) - v_j \parallel^2 \tag{54}$$

以及正则化损失函数

$$\mathcal{L}_c = \frac{1}{N}(\parallel C \parallel_f - \parallel diag(C) \parallel_2^2)$$

$$C_{ij} = \frac{1}{N}\sum_n (y_i^n - \mu_i)(y_j^n - \mu_j)$$

$$\mu_i = \frac{1}{N}\sum n(y_i^n)$$

其中，y_i^n为$user$或$item$的向量，n为$batch$中的引索，i为向量维度下标，N为$batch$大小

> 协方差可以看作向量之间的冗余，正则化损失引入目的是接触各维度见的相关性，使得整个空间能有更充分的信息表达

综上$(52)(54)(55)$得到最终损失函数为：

$$\min_{\theta,u_*,v_*} \mathcal{L}_m + \lambda_f \mathcal{L}_f + \lambda_c \mathcal{L}_c \\ s.t. \parallel u_* \parallel^2 \leqslant 1, \parallel v_* \parallel^2 \leqslant 1 \tag{56}$$

至此为CML