生成模型读书笔记八

2021年1月11日 21:53

GAN II----WGAN与WGAN-GP

既然基于原始GAN的各种loss function形式都有各种各样的问题，那他们的根本问题到底出在哪里，到底有没有一个比较好的GAN模型可以避免这些问题呢?

我们来回忆一下前文提到的三种loss都在优化什么，第一种，优化JS距离，第二种，优化一个矛盾的KL，JS距离，第三种最大似然，而本文在讲到最大似然的时候就证明了最大似然也是等价于最小化KL，所以说，三种形式始终逃不出KL距离/JS距离，那么是不是因为这个距离不合理呢？于是有人去研究了一下这个问题，并且指出它确实不合理。下面简单解释一下(复杂的数学公式我也不懂。。)

JS距离的定义是

$$JS(p\|q) = \frac{1}{2}KL(p\|\frac{p+q}{2}) + \frac{1}{2}KL(q\|\frac{p+q}{2}).$$
$$= \frac{1}{2}E_p\left(logp - log\frac{p+q}{2}\right) \frac{1}{2}E_q\left(logq - log\frac{p+q}{2}\right)$$

设想一下，刚开始训练时，生成分布就相当于是一个随机初始化的分布，它与真实分布很有可能相差很远，这就意味着他们重叠的部分几乎可以忽略不计。那么假设从p分布里采样一个x，因为p和q分布之间的重叠部分可以忽略，所以q(x)≈0，第一个期望里的第一项就等于log2，第二项就等于0，相对地，如果是从q分布里采样一个x，那么第二个期望里的第一项就会等于0，第二项等于log2。也就是说，只要两者重叠部分很少，可忽略，那么损失函数的值就会一直是一个常数，模型根本不知道要往哪里走，即使两个分布变得更近了(但是重叠部分还是很少)，也没有有效的指示。

在WGAN提出之前，针对这个问题，有一个简单的解决方案，就是给真实样本和生成样本加噪声，强行地扩大分布的范围，使得两个分布之间存在重叠部分。但是这个解决办法还是没法提供一个可以衡量训练进程的指标。

既然之前的距离度量都不好用，那要不干脆搞个网络让它自己学一个吧，想想看能不能把判别器D改改，让它学学怎么衡量两个分布的距离？终于轮到WGAN出场了。WGAN提出用Wasserstein距离代替JS距离，可以更好地指示训练进程，稳定训练。

i. 什么是Wasserstein距离

wasserstein距离又叫做earth-mover距离(EM距离)，定义如下：

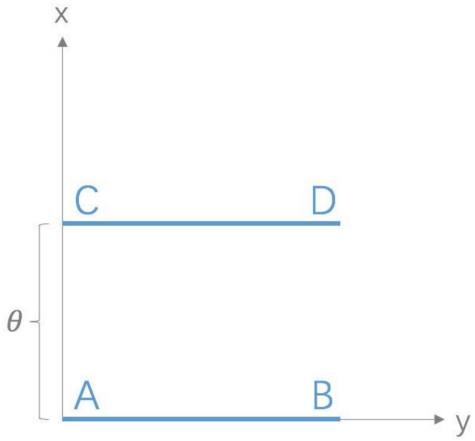
$$W(P_{data}, P_g) = \inf_{\gamma\sim\Pi(P_{data}, P_g)}\mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|]$$

$\Pi(P_{data}, P_g)$是联合分布的集合，这个集合里的联合分布是真实分布和生成分布之间所有可能的组合，也就是说这个集合里的每一个分布的边缘分布都是P_{data}和P_{g}。我们从所有可能的组合中采样一个联合分布γ出来，计算γ分布中所有(x,y)点对的距离的期望，这个期望的下界就wasserstein距离。一句话说就是两个分布中点对距离期望的最小值。

来看一个例子，假设我们有两个在二维空间中的分布p和q，如下图所示，p是AB线段上的均匀分布，q是CD线段上的均匀分布，两个线段之间的距离是θ。p和q点对距离期望的最小值显然就是θ(因为均匀分布，每个点对的概率都一样)。这就像是要把分布p推到q的位置，因此又称为推土机距离。

假如用JS或者KL来计算p和q分布的距离，因为p(x) q(x)中的x取值都不重叠，所以无论是JS还是KL都是一个突变的距离，随着θ变小，KL会从无穷突变到0，JS从log2突变到0，而EM距离则是平滑的，就等于θ。因此。它可以更好地量度两个不重叠分布之间的距离，反应分布之间的远近。

EM距离什么时候会最小？当两个分布中所有样本都重合的时候，EM距离就最小。比如这个例子就是个最简单的情况，均匀分布，所以只要θ=0，所有样本都会重合，EM距离=0



ii. WGAN的损失函数

Wasserstein距离的计算是没法直接求解的，所以我们用一个等价的式子

$$W(P_{\text{data}}, P_g)=\frac{1}{K}SUP_{\|f\|<K}E_{x\sim p_{data}}[f(x)]-E_{x\sim p_g}[f(x)]$$

上式中的$\|f\|<K$来自Lipschitz连续。Lipschitz连续要求存在一个K≥0，使得对于任意两点x_1, x_2函数f(x)都满足

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

简而言之就是对函数f(x)梯度的绝对值的限制，不能超过某个值K。对于所有满足条件的f(x)，f(x)在两个分布上的期望值的差的上界就是Wasserstein距离。

在实现中怎么表示"所有满足Lipschitz限制的f(x)"呢？不知道怎么做的时候，我们就用神经网络去模拟它。用带参数w的神经网络$f_w(x)$来表示f(x)，因为神经网络强大的拟合能力，虽然不能保证表示所有满足条件的f(x)，但也是一个很好的近似了。"所有f(x)"解决了，那么Lipschitz限制又怎么实现呢？WGAN用了一个简单粗暴的办法，就是直接做weight clipping，限制网络的所有参数θ^d不超过某个值[-c, c]，这样f(x)对于x的梯度$\frac{\partial f(x)}{\partial x}$就不会超过某个K(虽然我们不知道这个K是什么，但是只要限制了c，就肯定存在一个不是∞的K)。最后式子中的求上界SUP就用max来实现。

判别器被用来拟合上述距离，也对应我们前面说的干脆用一个网络来学距离。于是我们就得到了判别器D的损失函数

$$J(D) = max_{\theta^d}\ E_{x\sim p_{data}}[D(x)]-E_{x\sim p_g}[D(x)]$$

在训练好了D之后，理论上来讲它现在就是能够准确指示当前P_g与P_{data}距离的一个函数了，然后我们再去优化G。G希望生成分布P_g与真实分布P_{data}尽量接近，也即上述距离尽量小，所以他的目标函数是D的反，式子的第一项与G无关，因此最终G的目标函数写成

$$J(G) = min_{\theta^g}\ -E_{x\sim p_g}[D(x)]$$

原始的GAN里，D是一个二分类器，现在的D是两个分布距离的量度，所以在WGAN中又把它叫做critic网络而不是discriminator。在原始的GAN中，因为要做二分类，所以

D的最后一层用sigmoid，在WGAN中，D是要拟合Wasserstein距离，相当于一个回归问题而不是分类问题，需要把最后一层sigmoid拿掉。

iii. 实现

原始GAN的代码只要改几个地方就能够实现WGAN：

去掉最后一层sigmoid层；

生成器和判别器的loss计算去掉log；

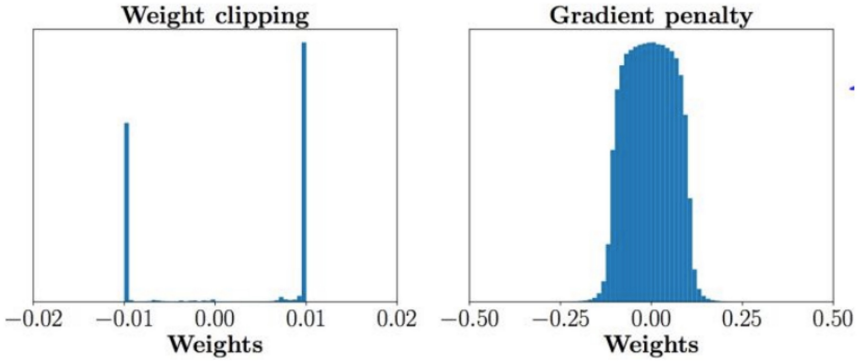
每次更新判别器的参数后需要做weight clipping；

不推荐使用基于动量的算法，因为判别器的loss不稳定，可以用SGD或者RMSProp

iv. WGAN-GP

1) Weight clipping的问题

上文说到对于实现Lipschitz限制使用了简单粗暴的weight-clipping，这个方法有很明显的问题，因为判别器是要学习距离的，那么对于不同输入，不同分布的样本，得到的输出必须有较大的不同，这时候网络的weight肯定不能太小，而同时weight clipping又限制了网络的weight又不能太大，就会导致大量的weight都取clipping区间的两个端点。比如clip在[-0.01,0.01]区间，经过实验，会发现网络weights的分布真的就是集中在两个端点，如下面左图所示



这样子网络跟二值输出都没有太大区别了，它本来该有的拟合能力大大浪费了。

所以WGAN作者后来又提出了一个更好的实现Lipschitz限制的办法，称为gradient penalty，应用这个方法的模型就叫做WGAN-GP。上方右图就是应用了GP方法之后的weights分布，明显合理很多。

除了weights的分布极端以外，weight clipping还存在容易导致梯度消失或者爆炸的问题。因为设想对一个多层网络做backpropagation(自己复习一下BP)，计算过程相当于中间每一层的weight/weight矩阵相乘，如果这个weight clipping的阈值设置得比较小，经过多次相乘后，就会是一个很小的数，容易出现梯度消失，假如设得比较大，经过多次相乘之后，就会是一个很大的数，容易出现梯度爆炸。这个阈值的选取就非常困难，实际应用中，这个从消失到爆炸的区域可能很小，调参就非常困难了。

2) Gradient penalty

那有没有比较合理的办法呢？首先来回忆一下Lipschitz限制，是说函数f(x)任意两点间的梯度的绝对值不超过某个数K。那么我们干脆直接把它写到loss里面好了。f(x)梯度的绝对值不超过K，设计成一个loss就可以写成下式，对于不超过K的，loss=0，对于超过K的，超过部分就算入loss：

ReLU$(\|\nabla_x(D(x))\|_p - K)$ 或者 $\max(\|\nabla_x(D(x))\|_p, K)$ (两个式子一样意思的，因为max(y, 0)在神经网络实现中就是用ReLU)

上式中的$\|\|_p$是指p norm，一般会用L1 norm或者L2 norm.

结合前面说的，为了尽量把真假样本分开，判别器希望学习到的距离大一些，所以一般会落在K附近，比K大一点小一点影响不大，所以又可以写成一个好看一点的loss来表示希望离K越近越好(用上面的式子还是下面的式子结果影响不大)

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

将这个loss加到上一节的D的loss里面去，就有

$$J(D) = max_{\theta^d} E_{x\sim p_{data}}[D(x)] - E_{x\sim p_g}[D(x)] - \lambda E_{x\sim X}[\|\nabla_x(D(x))\|_p - K]^2$$

或者min的形式，比较常用

$$J(D) = min_{\theta^d} E_{x\sim p_g}[D(x)] - E_{x\sim p_{data}}[D(x)] + \lambda E_{x\sim X}[\|\nabla_x(D(x))\|_p - K]^2$$

到这一步，GP算是完成了一半。

来看看上式的实现，前面两项和之前的一样，没什么特别的。新加的第三项，问题来了，因为Lipschitz限制是要求f(x)任意两点x的梯度，所以这一项需要对全体X进行采样，这怎么可能做得到嘛。f(x)是用来算真实分布和生成分布之间的距离的，所以把真实分布和生成分布之外的部分截掉，也丝毫不会影响结果。所以这里的实现就可以改成只对生成分布，真实分布，和夹在两个分布中间的区域(生成分布会向真实分布靠近，经过中间区域)采样就可以了，其他区域是否满足限制我们不关心。因此，具体实现如下：

先采样一对真假样本，以及从均匀分布[0,1]中采样一个值

$$x_1\sim p_{data},\ x_2\sim p_g,\ \varepsilon\sim U(0,1)$$

利用ε在x1和x2的连线上随机插值，得到一个在两个分布中间区域的采样

$$\hat{x} = \varepsilon x_1 + (1-\varepsilon)x_2$$

将这样采样得到的样本分布记为$p_{\hat{x}}$，就可以得到最终的loss function，在论文中K取1

$$J(D) = min_{\theta^d} E_{x\sim p_g}[D(x)] - E_{x\sim p_{data}}[D(x)] + \lambda E_{x\sim p_{\hat{x}}}[\|\nabla_x(D(x))\|_p - K]^2$$

论文实验证明，GP可以改善上述问题，提高训练速度，加快收敛。

v. 一个比较简单有趣的从零到WGAN-GP的推导

https://kexue.fm/archives/4439 有兴趣可以看看，比较简单，没有上述严格的推导和分析。文章直接没有提Wasserstein距离，而是从判别器需要学习一个距离衡量的角度出发，推导出WGAN-GP，感觉可以用同样的思路来理解loss sensitive GAN