

8 互怼的艺术：从零直达WGAN-GP

Jun By 苏剑林 | 2017-06-08 | 133877位读者

前言

GAN, 全称Generative Adversarial Nets, 中文名是生成对抗式网络。对于GAN来说, 最通俗的解释就是“伪造者-鉴别者”的解释, 如艺术画的伪造者和鉴别者。一开始伪造者和鉴别者的水平都不高, 但是鉴别者还是比较容易鉴别出伪造者伪造出来的艺术画。但随着伪造者对伪造技术的学习后, 其伪造的艺术画会让鉴别者识别错误; 或者随着鉴别者对鉴别技术的学习后, 能够很简单的鉴别出伪造者伪造的艺术画。这是一个双方不断学习技术, 以达到最高的伪造和鉴别水平的过程。然而, 稍微深入了解的读者就会发现, **跟现实中的造假者不同, 造假者会与时俱进地使用新材料新技术来造假, 而GAN最神奇而又让人困惑的地方是它能够将随机噪声映射为我们所希望的正样本, 有噪声就有正样本, 这不是无本生意吗, 多划算~**

另一个情况是, 自从WGAN提出以来, 基本上GAN的主流研究都已经变成了WGAN上去了, 但WGAN的形式事实上已经跟“伪造者-鉴别者”差得比较远了。而且WGAN虽然最后的形式并不复杂, 但是推导过程却用到了诸多复杂的数学, 使得我无心研读原始论文。这迫使我要找从一条简明直观的线索来理解GAN。幸好, 经过一段时间的思考, 有点收获。

在正文之前, 先声明: **笔者所有的GAN的知识, 仅仅从网上的科普文所读而来, 我并没有直接读过任何关于GAN的论文, 因此, 文中的结果可能跟主流的结果有雷同, 也可能有很大出入, 而且本文的讲述方法并不符合GAN的历史发展进程。严谨治学者慎入~**

注: 如无指明, 本文所谈到的GAN都是广义的, 即包括原始GAN、WGAN等等, 对它们不作区分~ 文中出现的正样本、真实样本, 都是指预先指定的一批样本, 而生成样本则指的是随机噪声通过生成模型 G 变换所得的结果。

一道面试题

一道经典的面试题是: **如果有一个伪随机数程序能够生成 $[0, 1]$ 之间的均匀随机数, 那么如何由它来生成服从正态分布的伪随机数? 比如怎么将 $U[0, 1]$ 映射成 $N(0, 1)$?**

这道题不同的角度有不同的做法, 工程上的做法有: 同时运行 n 个这样的伪随机数程序, 每步产生 n 个随机数, 那么这 n 个数的和就近似服从正态分布了。不过, 这里不关心工程做法, 而关心理论上的做法。理论上的做法是: 将 $X \sim U[0, 1]$ 经过函数 $Y = f(X)$ 映射之后, 就有 $Y \sim N(0, 1)$ 了。设 $\rho(x)$ 是 $U[0, 1]$ 是概率密度函数, 那么 $[x, x + dx]$ 和 $[y, y + dy]$ 这两个区间的概率应该相等, 而根据概率密度定义, $\rho(x)$ 不是概率, $\rho(x)dx$ 才是概率, 因此有

$$\rho(x)dx = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right)dy$$

那么

$$\int_0^x \rho(t)dt = \int_{-\infty}^y \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right)dt = \Phi(y)$$

这里 $0 \leq x \leq 1, y \in (-\infty, +\infty)$, 而 $\Phi(y)$ 是标准正态分布的累积分布函数, 所以

$$y = \Phi^{-1} \left(\int_0^x \rho(t) dt \right)$$

注意到累积分布函数是无法用初等函数显式表示出来的，更不用说它的逆函数了。说白了， $Y = f(X)$ 的 f 的确是存在的，但很复杂，以上解只是一个记号，该算的还是要用计算机算。

正态分布是常见的、相对简单的分布，但这个映射已经这么复杂了。如果换了任意分布，甚至概率密度函数都不能显式写出来，那么复杂度可想而知～

神经大法好

现在我们将问题一般化：如何找到映射 $Y = f(X)$ ，把服从均匀分布 X 映射到指定的分布？在一般情形下，这个指定的分布是通过给出一批具体的分布样本 $Z = (z_1, z_2, \dots, z_N)$ 来描述的（比如，给出一批服从正态分布的随机数，而不是给出它的概率密度 $\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ 。）。

这个问题相当一般化，跟GAN所做的事情也是一样的。也就是说，GAN也是希望把均匀的随机噪声映射成特定分布，这个特定分布由一组“正样本”描述。这样的理解就可以回答我们开头的一个小问题了：为什么GAN可以将噪声变换成正样本？事实上GAN并不是学习噪声到正样本的变换，而是学习均匀分布到指定分布的变换。假如学习成功了，那么输入一个随机噪声，那么就变换成指定分布的数据，而通常来说我们指定的分布是一个比较“窄”的分布（比如指定的正样本是某一类图片的集合，但事实上图片无穷无尽，某一类的图片是相当窄的），所以都会映射到我们眼中的“正样本”去。

前面正态分布的例子已经表明，这个映射 f 通常都是很复杂的，因此没必要求它的解析解。这时候“神经大法”就登场了：熟悉神经网络的读者都知道，我们总可以用一个神经网络来拟合任意函数，因此，不妨用一个带有多个参数的神经网络 $G(X, \theta)$ 去拟合它？只要把参数 θ 训练好，就可以认为 $Y = G(X, \theta)$ 了。

可是，问题又来了：拟合什么目标呢？我们怎么知道 $Y = G(X, \theta)$ 跟指定的分布是很接近的呢？

KL距离？JS距离？

让我们把问题再理清楚一下：我们现在有一批服从某个指定分布的数据 $Z = (z_1, z_2, \dots, z_N)$ ，我们希望找到一个神经网络 $Y = G(X, \theta)$ ，将均匀随机数 X 映射到这个指定分布中来。

需要特别指出，我们是要比较两个分布的接近程度，而不是比较样本之间的差距。通常来说，我们会用KL距离来描述两个分布的差异：设 $p_1(x), p_2(x)$ 是两个分布的概率密度（当然，还有其他距离可以选择，比如Wassers tein距离，但这不改变下面要讨论的内容的实质），那么

$$KL(p_1(x) \| p_2(x)) = \int p_1(x) \log \frac{p_1(x)}{p_2(x)} dx$$

如果是离散概率，则将积分换成求和即可。KL距离并非真正的度量距离，但是它能够描述两个分布之间的差异，当它是0时，表明两个分布一致。因为它不是对称的。有时候将它对称化，得到JS距离：

$$JS(p_1(x), p_2(x)) = \frac{1}{2} KL(p_1(x) \| p_2(x)) + \frac{1}{2} KL(p_2(x) \| p_1(x))$$

咦？怎么又回到概率密度了？不是说没给出概率密度吗？没办法，公式就是这样，只好估算一下咯。假设我们可以将实数域分若干个不相交的区间 I_1, I_2, \dots, I_K ，那么就可以估算一下给定分布 Z 的概率分布

$$p_z(I_i) = \frac{1}{N} \sum_{j=1}^N \#(z_j \in I_i)$$

其中 $\#(z_j \in I_i)$ 表示如果 $z_j \in I_i$ ，那么取值为1，否则为0，也就是说大家不要被公式唬住了，上式就是一个简单的计数函数，用频率估计概率罢了。

接着我们生成 M 个均匀随机数 x_1, x_2, \dots, x_M （这里不一定要 $M = N$ ，还是那句话，我们比较的是分布，不是样本本身，因此多一个少一个样本，对分布的估算也差不了多少。），根据 $Y = G(X, \theta)$ 计算对应的 y_1, y_2, \dots, y_M ，然后根据公式可以计算

$$p_y(I_i) = \frac{1}{M} \sum_{j=1}^M \#(y_j \in I_i)$$

现在有了 $p_z(I_i)$ 和 $p_y(I_i)$ ，那么我们就可以算它们的差距了，比如可以选择JS距离

$$\text{Loss} = JS(p_y(I_i), p_z(I_i))$$

注意 y_i 是由 $G(X, \theta)$ 生成的，所以 $p_y(I_i)$ 是带有参数 θ 的，因此可以通过最小化Loss来得到参数 θ 的最优值，从而决定网络 $Y = G(X, \theta)$ 。

神经距离！

假如我们只研究单变量概率分布之间的变换，那上述过程完全够了。然而，很多真正有意义的事情都是多元的，比如在MNIST上做实验，想要将随机噪声变换成手写数字图像。要注意MNIST的图像是 $28 \times 28 = 784$ 像素的，假如每个像素都是随机的，那么这就是一个784元的概率分布。按照我们前面分区间来计算KL距离或者JS距离，哪怕每个像素只分两个区间，那么就有 $2^{784} \approx 10^{236}$ 个区间，这是何其巨大的计算量！

终于，有人怒了：“老子干嘛要用你那逗比的JS距离，老子自己用神经网络造一个距离！”于是他写出带参数 Θ 的神经网络：

$$L(\{y_i\}_{i=1}^M, \{z_i\}_{i=1}^N, \Theta)$$

也就是说，直接将造出来的 y_i 和真实的 z_i 都放进去这个神经网络一算，自动出来距离，多方便。**这个思想是里程碑式的，它连距离的定义都直接用神经网络学了，还有什么不可能学的呢？**

我们来看看，要是真有这么个 L 存在，它应该是怎么样的？首先，对于特定的任务来说， $\{z_i\}_{i=1}^N$ 是给定的，因此它并非变量，我们可以把它当做模型本身的一部分，因此简写成

$$L(\{y_i\}_{i=1}^M, \Theta)$$

接着，别忘记我们是**描述分布之间的距离而不是样本的距离**，而分布本身跟各个 y_i 出现的顺序是没有关系的，因此分布之间的距离跟各个 y_i 出现的顺序是无关的，也就是说，尽管 L 是各个 y_i 的函数，但它必须全对称的！这是个很强的约束，当然，尽管如此，我们的选择也有很多，比如

$$L = \frac{1}{M!} \sum_{\text{对 } y_1, \dots, y_M \text{ 所有的排列求和}} D(y_1, y_2, \dots, y_M, \Theta)$$

也就是说，我们先找一个有序的函数 D ，然后对所有可能的序求平均，那么就得到无序的函数了。当然，这样

的计算量是 $\mathcal{O}(M!)$ ，显然也不靠谱，那么我们就选择最简单的一种：

$$L = \frac{1}{M} \sum_{i=1}^M D(y_i, \Theta)$$

这便是**无序的最简单实现**，可以简单的理解为：分布之间的距离，等于单个样本的距离的平均。

对抗来了 ~

“等等，你的标题是GAN，你讲了那么一大通，我怎么没感觉到半点GAN的味道呀？对抗在哪里？”这位看官您别急，马上就有了~（这位看官，看来你就是等着看干架的呀^_^）

前面说到，用神经网络来学习一个距离 L ，最终简化版的形式，应该是这样的

$$L = \frac{1}{M} \sum_{i=1}^M D(y_i, \Theta)$$

问题是： $D(Y, \Theta)$ 怎么训练？别忘了，之前的 $G(X, \theta)$ 还没有训练好，现在又弄个 $D(Y, \Theta)$ 出来，越搞越复杂，小心跳到坑里出不来了~（GAN还真的是个大坑）

对抗终于来了...

因为 $D(Y, \Theta)$ 的均值，也就是 L ，是度量两个分布的差异程度，这就意味着， L 要能够将两个分布区分开来，即 L 越大越好；但是我们最终的目的，是希望通过均匀分布而生成我们指定的分布，所以 $G(X, \theta)$ 则希望两个分布越来越接近，即 L 越小越好。这时候，一个天才的想法出现了：**互怼！**不要怂，gan！

首先我们随机初始化 $G(X, \theta)$ ，固定它，然后生成一批 Y ，这时候我们要训练 $D(Y, \Theta)$ ，既然 L 代表的是“与指定样本 Z 的差异”，那么，如果将指定样本 Z 代入 L ，结果应该是越小越好，而将 Y 代入 L ，结果应该是越大越好，所以

$$\begin{aligned}\Theta &= \arg \min_{\Theta} L = \arg \min_{\Theta} \frac{1}{N} \sum_{i=1}^N D(z_i, \Theta) \\ \Theta &= \arg \max_{\Theta} L = \arg \max_{\Theta} \frac{1}{M} \sum_{i=1}^M D(y_i, \Theta)\end{aligned}$$

然而有两个目标并不容易平衡，所以干脆都取同样的样本数 B （一个batch），然后一起训练就好：

$$\begin{aligned}\Theta &= \arg \max_{\Theta} L_1 \\ &= \arg \max_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(y_i, \Theta) - D(z_i, \Theta)]\end{aligned}$$

很自然， $G(X, \theta)$ 希望它生成的样本越接近真实样本越好，因此这时候把 Θ 固定，只训练 θ 让 L 越来越小：

$$\begin{aligned}\theta &= \arg \min_{\theta} L_2 \\ &= \arg \min_{\theta} \frac{1}{B} \sum_{i=1}^B [D(G(x_i, \theta), \Theta)]\end{aligned}$$

这就是天才的对抗网络！

需要指出的是：

- 1、这里的Loss写法跟传统的GAN相反，习惯性的做法是让真实样本的 L 越大越好，但这只不过跟本文差了个负号而已，不是本质的问题；
- 2、从GAN开始， D 这个神经网络就被赋予了“判别器”的意义，但在这里 D 本身是没有意义的（正如我们不能说某个数是不是正态分布的），只有 D 的平均值 L 才代表着与真实分布的差距（我们只能根据一批数据来估计它是否服从正态分布），所以从这里也可以看到，GAN不能单个样本地训练，至少成批训练，因为有一批样本才能看出统计特征；
- 3、乍看上去 D 只是个二分类问题，而 G 则要把噪声映射为正样本，貌似 D 应该比 G 要简单得多？事实并非如此，它们两者的复杂度至少是相当的。我们可以直观考虑一下它们的工作原理：因为 D 的均值 L 直接就给出了输入的数据与指定分布的差异，而要真的做到这一点，那么 D 要把所有的“正样本”（在某种程度上）都“记住”了才行；而 G 要生成良好的正样本，基本上也是“记住”了所有的正样本，并通过随机数来插值输出。因此两个网络的复杂度应该是相当的（当然这里的“记住”是形象理解，不是真的强行记住了，不然就是过拟合了）；
- 4、既然 L_1 是真伪样本的分布差（的最大值），那么 L_1 越小，意味着“伪造”的样本质量越好，所以 L_1 同时也指示着GAN训练的进程， L_1 越小，训练得越好。

别走～还没完

稍微思考一下，我们就发现，问题还没完。我们目前还没有对 D 做约束，不难发现，无约束的话Loss基本上会直接跑到负无穷去了～

因此，有必要给 D 加点条件，一个比较容易想到的方案是约束 D 的范围，比如能不能给 D 最后的输出加个Sigmoid激活函数，让它取值在0到1之间？事实上这个方案在理论上是没有任何问题的，然而这会造成训练的困难。因为Sigmoid函数具有饱和区，一旦 D 进入了饱和区，就很难传回梯度来更新 G 了。

最好加什么约束呢？**我们应该尽可能从基本原理出发来寻找约束，尽量避免加入人工因素。**我们回到距离的作用上来看：距离是为了表明两个对象的差距，而如果对象产生的微小的变化，那么距离的波动也不能太大，这应该是**对距离基本的稳定性要求**，“失之毫厘，谬以千里”是会产生浑沌的，数学模型不应该是这样。从这个角度来看，那个所谓的“JS距离”，根据就不是距离了，因为就算对于伯努利分布 $\{0:0.1, 1:0.9\}$ 和 $\{0:0, 1:1\}$ ，这两个相似分布算出来的“距离”居然是无穷大（因为出现了 $0.1/0$ 这一项）。

放到我们的 D 中，这个约束我们该怎么体现呢？假如某个样本不是 y_i 而是 y'_i ，假设 $\|y_i - y'_i\|$ （用两竖表示欧式距离，因为 y 可能是个多元向量）并不是十分大，那么会对分布造成一定的影响。这个影响有多大呢？显然不会大，因为分布是一批样本的统计特征，如果只是稍微改变了一个样本，那么分布的变化显然不能大的。而我们知道，分布的距离用 D 的均值 L 来描述，只改变一个 y_i ，所造成的分布差正比于

$$\|D(y_i, \theta) - D(y'_i, \theta)\|$$

我们希望 $y'_i \rightarrow y_i$ 时，自然地就有 $\|D(y_i, \theta) - D(y'_i, \theta)\| \rightarrow 0$ ，怎么实现这一点呢？一个简单的方案是 D 满足以下约束：

$$\|D(y, \theta) - D(y', \theta)\| \leq C\|y - y'\|^\alpha$$

这里 $\alpha > 0$ ，而最简单的方案就是

$$\|D(y, \theta) - D(y', \theta)\| \leq C\|y - y'\|$$

这就是数学中常见的**Lipschitz约束**。如果能够满足这个约束，那么距离就能满足稳定性要求。注意这是个充分条件，不是必要条件，也可以使用其他方案。但不得不说，这是个简单明了的方案。而使得函数 D 满足Lipschitz约束的一个充分条件就是：

$$\left\| \frac{\partial D(y, \Theta)}{\partial y} \right\| \leq C$$

“罚”出来的成果

怎么把这个约束加入到模型中去呢？在

$$\Theta = \arg \max_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(y_i, \Theta) - D(z_i, \Theta)] = \arg \min_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(z_i, \Theta) - D(y_i, \Theta)]$$

的基础上，加入个惩罚项就好：

$$\Theta = \arg \min_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(z_i, \Theta) - D(y_i, \Theta)] + \lambda \max \left(\left\| \frac{\partial D(y, \Theta)}{\partial y} \right\|, 1 \right)$$

当然惩罚是“软约束”，最终的结果不一定满足这个约束，但却会在约束上下波动。也就是说虽然我们指定了 $C = 1$ ，但最终的 C 却不一定等于1，不过会在1上下波动，而这也不过是一个更宽松的Lipschitz约束而已，我们不在乎 C 的具体大小，只要 C 有上界就好。另外，约束的加法不是唯一的，WGAN的作者Martin Arjovsky在他的论文中提出的加法为：

$$\Theta = \arg \min_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(z_i, \Theta) - D(y_i, \Theta)] + \lambda \left(\left\| \frac{\partial D(y, \Theta)}{\partial y} \right\| - 1 \right)^2$$

哪个好？实验结果好像都差不多。

不过，上面的惩罚项都是形式而已，我们还没给出具体的计算方法。理论上最好能够对所有的 y （全空间）都算一遍 $\left\| \frac{\partial D(y, \Theta)}{\partial y} \right\|$ 然后取平均，显然这是做不到的。那么只好用一个退而求其次的方案：只对真实样本 z_i 和生成样本 y_i 算。但这样约束范围貌似也太小了，所以干脆在真实样本和生成样本之间随机插值，希望这个约束可以“布满”真实样本和生成样本之间的空间，即

$$\Theta = \arg \min_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(z_i, \Theta) - D(y_i, \Theta)] + \frac{\lambda}{B} \sum_{i=1}^B \max \left(\left\| \frac{\partial D(y, \Theta)}{\partial y} \right\|_{y=\varepsilon_i y_i + (1-\varepsilon_i) z_i}, 1 \right)$$

以及

$$\Theta = \arg \min_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(z_i, \Theta) - D(y_i, \Theta)] + \frac{\lambda}{B} \sum_{i=1}^B \left(\left\| \frac{\partial D(y, \Theta)}{\partial y} \right\|_{y=\varepsilon_i y_i + (1-\varepsilon_i) z_i} - 1 \right)^2$$

这里的 ε_i 是 $U[0, 1]$ 的随机数，这应该已经是我们“力所能及”的最优的方案了。后面这个就是Martin Arjovsky提出的最新的Lipschitz约束的方案，而实验结果表明前一个方案效果也不错。目前它们的大名叫“WGAN-GP”，全称Wasserstein Generative Adversarial Nets - Gradient Penalty。

最后，有人会反驳，梯度有上界，只不过是Lipschitz约束的充分条件，**为啥不直接将Lipschitz约束以差分形式加入到惩罚中去呢？**（其实有这个疑问的最主要的原因，是很多深度学习框架并没有提供梯度函数；另外，尽

管tensorflow提供了梯度函数，但如果判别器用的是RNN，那么梯度函数也是不可用的。）事实上，这样做某种意义上更加合理，我觉得Martin Arjovsky直接用梯度，不过是想写得简单一点~这时候惩罚是

$$\Theta = \arg \min_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(z_i, \Theta) - D(y_i, \Theta)] + \frac{\lambda}{B} \sum_{i=1}^B \max \left(\frac{|D(y_{i,1}, \Theta) - D(y_{i,2}, \Theta)|}{\|y_{i,1} - y_{i,2}\|}, 1 \right)$$

以及

$$\Theta = \arg \min_{\Theta} \frac{1}{B} \sum_{i=1}^B [D(z_i, \Theta) - D(y_i, \Theta)] + \frac{\lambda}{B} \sum_{i=1}^B \left(\frac{|D(y_{i,1}, \Theta) - D(y_{i,2}, \Theta)|}{\|y_{i,1} - y_{i,2}\|} - 1 \right)^2$$

这里 $y_{i,j} = \varepsilon_{i,j} y_i + (1 - \varepsilon_{i,j}) z_i$ ，也就是每步插值两次，然后用插值的结果算差分。

然后呢？

暂时没有然后了~终于写完了。这便是我理解的GAN。

通过本文，我们可以一气呵成地直达WGAN-GP，而不需要很多的历史知识和数学知识。有趣的是，**我们的推导过程表明，WGAN-GP其实跟Wasserstein距离没有直接的联系**，尽管当初WGAN的作者是从Wasserstein距离将它们推导出来的。也就是说，**WGAN跟W没啥关系，这就尴尬了，还能好好叫WGAN(Wasserstein GAN)么~**另外，有人提问“WGAN相比原始的GAN有什么优势？”，如果根据本文的理论推导，那么原始的GAN根本就不是GAN，因为它不能改写为本文的某个特例！（原因在于，本文的推导基于分布的拟合，而原始GAN的推导基于博弈论，出发点不同。）


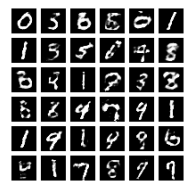
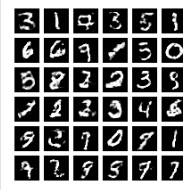
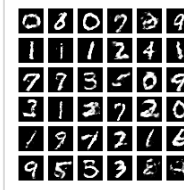

这个Loss还有一定的改进空间，比如Loss Sensitive GAN (LS-GAN)，还有更广义的CLS-GAN（将LS-GAN和WGAN统一起来了），这些推广我们就不讨论了。不过这些推广都建立在Lipschitz约束之上，只不过微调了Loss，也许未来会有人发现比Lipschitz约束更好的对D的约束~

WGAN-GP的例子

最后，分享一个WGAN-GP的实现，以MNIST为数据集，读者可以自己改着玩~

<https://github.com/bojone/gan/>

训练进度显示~

				
0	100	200	300	400

转载到请包括本文地址: <https://kexue.fm/archives/4439>

更详细的转载事宜请参考: 《科学空间FAQ》

如果您需要引用本文，请参考：

苏剑林. (Jun. 08, 2017). 《互怼的艺术：从零直达WGAN-GP 》 [Blog post]. Retrieved from <https://kexue.fm/archives/4439>