

GAN summary

Chen Gong

21 May 2021

目录

1	Introduction	1
1.1	GAN as structured learning	1
1.2	生成器可以自己学习吗?	1
1.3	判别器 (Discriminate) 可以自己生成图片吗?	4
1.4	小结	6
2	Conditional GAN	7
2.1	Conditional GAN 基本思想	7
2.2	Conditional GAN 基本框架	8
2.3	Conditional GAN 的应用场景	9
3	Unsupervised Conditional GAN	11
3.1	Direct Transformation	11
3.1.1	基本思想	11
3.1.2	Cycle-GAN	12
3.1.3	Issue of Cycle Consistency	13
3.1.4	Star GAN	13
4	投影到公共空间 (Projection to Common Space)	15
4.1	Couple GAN, UNIT	16
4.2	添加 domain 判别器	16
4.3	Cycle Consistency	16
4.4	XGAN	17
5	The Basic theory of GAN	17
5.1	Maximum Likelihood Estimation (MLE)	18
5.2	GAN 得到数据分布	18
5.3	判别器与散度之间的关系	20
5.4	GAN 的实际操作	22

6	f-GAN: General Framework of GAN	23
6.1	Fenchel Conjugate	24
6.2	Connection with GAN	25
6.3	Model Collapse & Model Dropping	25
7	Tips for improving GAN	27
7.1	JS 散度的缺点	27
7.2	Least Square GAN (LSGAN)	28
7.3	Wasserstein GAN (WGAN)	28
7.3.1	Weight Clipping	30
7.3.2	Improved WGAN	31
7.3.3	GAN 改成 WGAN	32
7.4	Energy-based GAN (EBGAN)	32
8	Feature Extraction	33
8.1	Info-GAN	33
8.2	VAE-GAN	34
8.3	BiGAN	35
8.4	Triple GAN	36
9	Improving Sequence Generation by GAN	36
9.1	Conditional Sequence Generation	36
9.1.1	RL improve Sequence-to-Sequence	37
9.1.2	Continuous Input for Discriminator	40
10	Evaluation	40
10.1	Kernel Density Estimation	41
10.2	Likelihood v.s. Quality	41
10.3	Objective Evaluation	42
10.4	Model Dropping	42

1 Introduction

本次 Summary 内容是参考李宏毅老师的 GAN 系列课程。首先一开始学到一个新名词，挺有意思的：**since sliced bread (有史以来)**。

GAN 是想让机器自己生成东西。随便给它一个输入，比方说高斯分布，就会产生一个图片或者一句话等，输入的 vector 将会产生不同的图片或语句。GAN 的目标就是训练出这样的 NN Generator。基础的东西本文中将不做过多的介绍。小编最近要扩大自己的思路，所以决定大致的看看 GAN 的基础思想，GAN 目前作为 ML 中最火的模型之一，非常值得学习。本小节的主要框架是，描述 GAN 作为结构学习的特点；

1.1 GAN as structured learning

ML 很多时候是为了找到函数映射： $f : X \rightarrow Y$ 。如果是 Regression 问题，则输出一个 scalar；如果是 classification 问题，则输出一个“class”(one-hot 向量)；而 structure 学习，则是输出更复杂的东西，比如句子，矩阵，图或者树

而为什么 structured learning 非常具有挑战呢？

1. One-shot/Zero-shot Learning: 如果将所有结果看成是一个“class”，而其输出空间非常的大，很多 class 没有 training data。所以，要让机器输出一个从来没有见过的东西，这对机器的泛化性要求非常高。
2. 机器必须要用大局观，需要学会 planning。在生成图片或者句子的过程中，每一个像素或者每一个字直接要相互联系，最后在一起组成一幅好看的画或者句子，这个大局观非常重要。一个眼睛画的再好看，如果一个人脸上画了三个眼睛也肯定不合适。

1.2 生成器可以自己学习吗？

下面将思考一个重要的问题，Generator 可以自己学习吗？当然可以，Generator 的输入是一个向量，而输出是一个图片，那么只需采用传统的监督学习的方式，用 $\{(vector, image)\}$ 这样的数据集训练就可了。

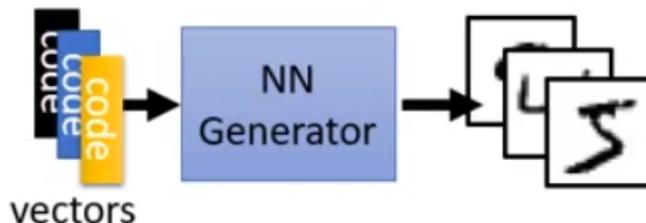


图 1: Generator 示意图

但是，问题马上就来了。当输出类似的时候，输入由于是随机的，有很大差别，这样很难训练出一个网络；**我们怎么获得输入向量？**当然有办法做，这是就搬出来大名鼎鼎的 Auto-Encoder 算法，通过 Encoder 来得到这样一个向量，而后半部分就是对应的 Generator。

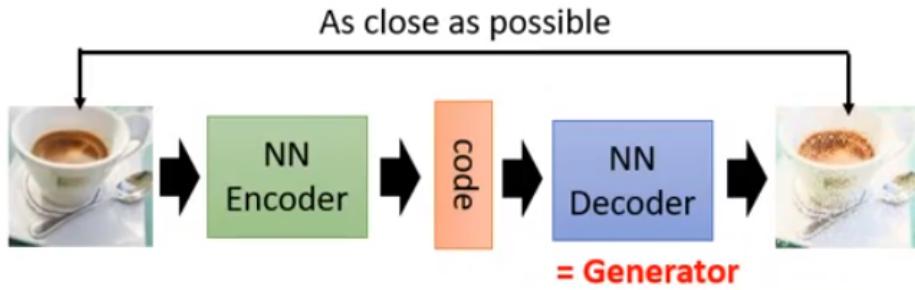


图 2: Auto-Encoder 算法示意图

但是存在一个问题: training data 里面的 image 是有限的, 当生成器在看到 a 的情况下生成偏左的 1, 在看到 b 的情况下也生成偏右 1, 当 a 和 b 的平均, 会产生什么样的图片呢? 事实是由于网络是非线性的, 得到的不一定是正的 1, 也可能是噪声。这样就很难办了, 算法泛化性不够, 当遇到没有见过的 vector 时, 无法生成较好的图片。为了解决这个问题, VAE 诞生了。

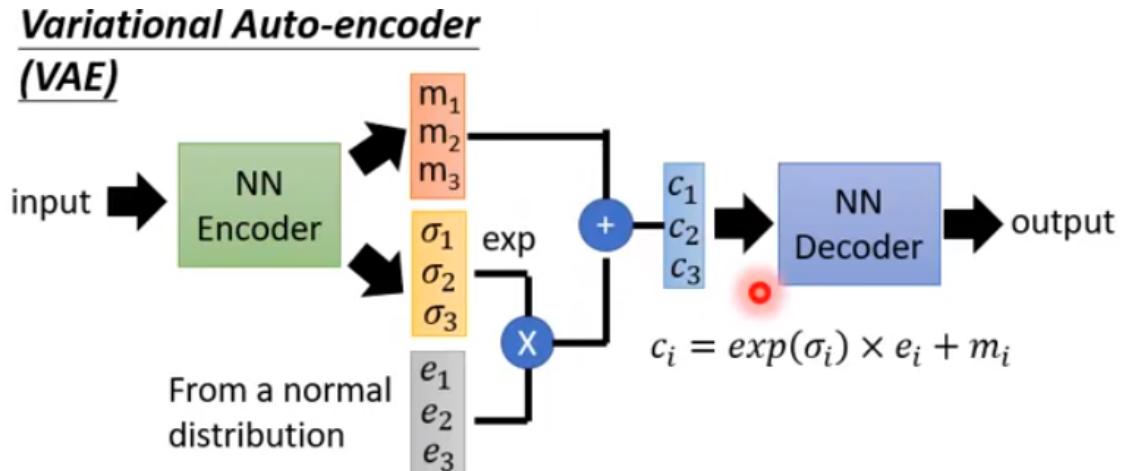


图 3: Variational Auto-Encoder 算法示意图

VAE 中通过在每一个维度都生成一个分布的方法, 从而得到向量的分布, 然后从中采样的方法来使算法更稳定。这样情况下, decoder 学习过程中就不只是看到单独的向量 a 或 b 而产生一些数字, 而且当看到向量 a 或 b 加上一些 noise 也要产生数字; 这样使 decoder 更加鲁棒。所以使得 VAE 比之前的 AE 要强大很多了。看到这似乎一切都很漂亮, 那么, 这样的 VAE 有什么问题那?

Generator 在 training 的时候, 希望的是最终生成器生成的图片和给定的图片在像素级别上越像越好。但是肯定会产生一些 mistakes。这些 mistake 可能会很致命, 就会导致虽然 mistake 算起来很小, 但是出现在了不出现的地方, 就会使图片看起来很奇怪。比如,

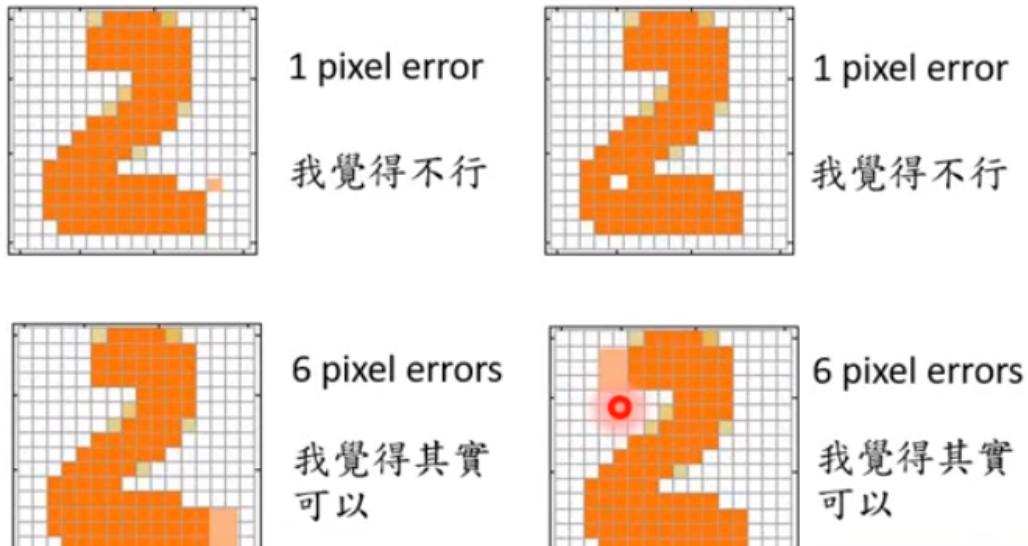


图 4: Variational Auto-Encoder 中面临的重要问题

显然，上两个图的误差更小，但是下两个图更符合人的认知。这是因为 VAE 中，往往会忽略输出点之间的连续，而 component 之间的联系实际上是非常重要的。如果点之间独立输出，而没有相互的影响，就会导致效果不行。

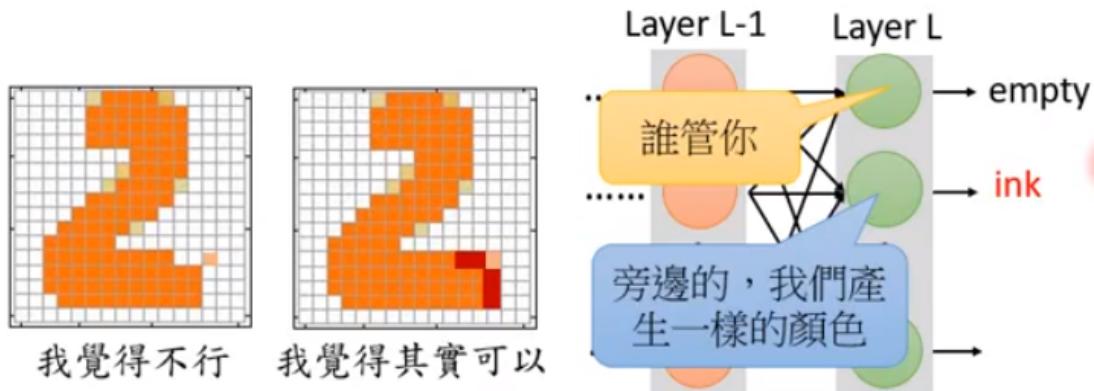


图 5: Variational Auto-Encoder 中面临的重要问题

而目前常见的方法是采用更深的网络，来加强不同的 component 之间的联系。从下面这幅图中，我们可以更加深刻的体会到这一点，绿色的是 VAE 学习的目标，而蓝色的是 generator 产生的目标。VAE 无法考虑到 x_1, x_2 之间的联系，比如它知道， x_1, x_2 都很大的情况下是好的。但是 x_1 很大， x_2 很小的情况下也是好的。但是，当 x_1 很大，而 x_2 不大不小的情况下，机器无法判断，就会觉得它应该也是好的，就会造成很多的错误。生成的即为下图所示的图片。

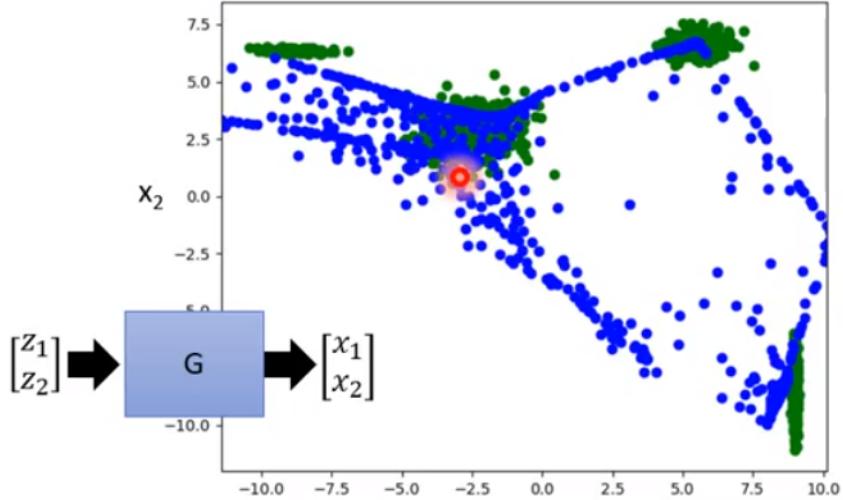


图 6: Variational Auto-Encoder 输出分布示意图

1.3 判别器 (Discriminate) 可以自己生成图片吗？

实际上，判别器可以自己生成图片，只不过过程非常的曲折。所谓，判别器就是给一张图片，从而生成一个 scalar 来判断这张图片好还是不好。Discriminate 表示为函数 $D : \mathcal{X} \rightarrow \mathcal{R}$ 。

对于单纯的生成器，想要去判别不同像素之间的关联是很困难的；对判别器来说，却很容易，因为有卷积核操作，卷积核会去检索有没有独立的像素点，有的话就是低分。

下一个问题是使用 Discriminator 来做生成？判别器可以鉴别一个图片好不好，那么我们的做法是穷举所有的 $x \in \mathcal{X}$ ，选择分数高的图片就行，

$$\hat{x} = \arg \max_{x \in \mathcal{X}} D(x) \quad (1)$$

这看着就很麻烦，，，下一个问题是如何去训练 Discriminate？因为我们目前只有好的样本，没有坏的样本。实际上下一个重要的问题是，如何找到有效的高质量的负样本来让 Discriminator 学习。

一个办法是直接用 noise，让它成为负样本。但是会出现一张稍微有点像图片的 picutre 就会得到很高的分数。这显然是不合适的，如下所示。但是怎样产生这些好的负样本呢？

- Negative examples are critical.

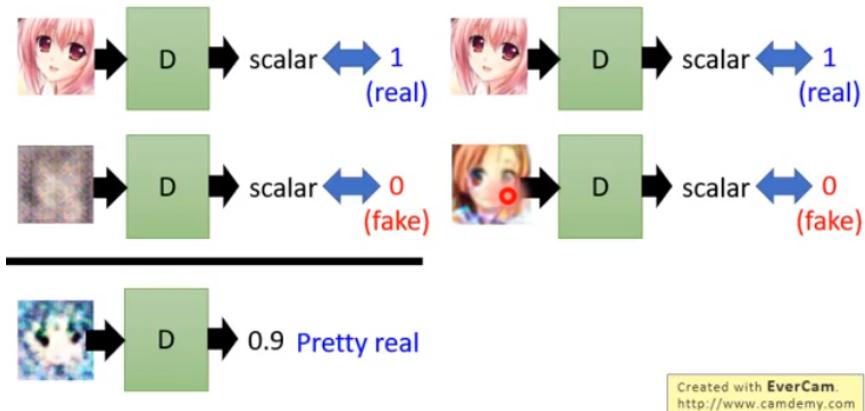


图 7: nice 负样本非常的重要

这时，大家都把目光聚集到了 Discriminator 上。不好的 Discriminator 生成的，它认为好的图片，这不就是绝佳的负样本吗？我们把这些它认为好的负样本不断的优化，就可以得到好的 Discriminator 了。这时候需要一个迭代算法，

- 开始的时候你又一堆正样本和负样本：正样本是真实图片负样本是噪声；
- 开始迭代
 - 判别器需要做的就是去给正样本高的分数，负样本低的分数。
 - 学出一个判别器后，再用判别器去做生成，生成一堆它觉得好的图片，之后作为负样本重复 1 过程。

Discriminator - Training

• General Algorithm



- Given a set of **positive examples**, randomly generate a set of **negative examples**.
- In each iteration
 - Learn a discriminator D that can discriminate positive and negative examples.



- Generate negative examples by discriminator D

$$\tilde{x} = \arg \max_{x \in X} D(x)$$

图 8: Disciminator 的训练过程

从概率分布的角度看会更清晰。其中，蓝色的表示是 Generated 的数据，绿色表示为 real 的数据。

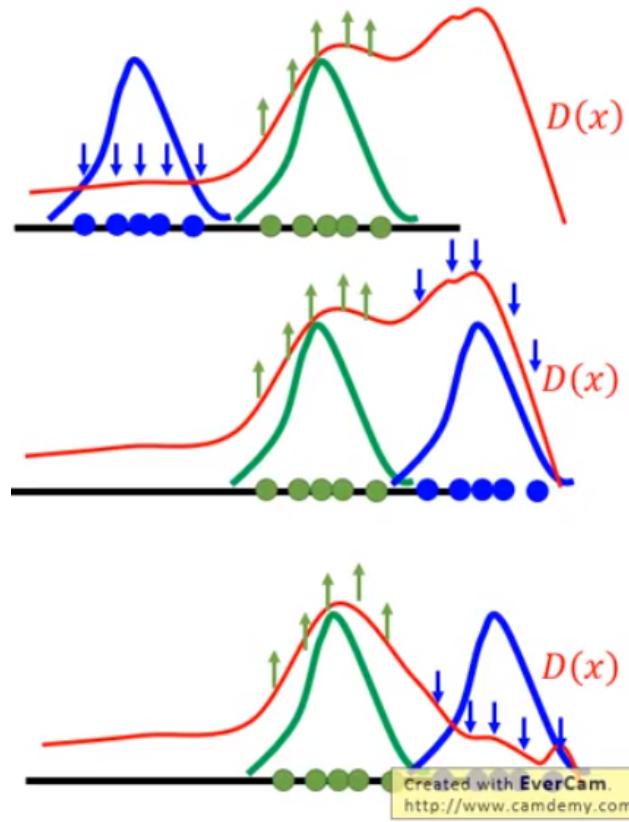


图 9: Disciminator 的训练过程的概率分布表示

第一次随机得到的 Generate 数据和 real 数据生成了一个 $D(x)$ ，但是右边的部分什么信息都没有，可能被赋予较好的分数。然后第二次迭代，根据 $D(x)$ 的得分，选择较好的数据，于是得到了左边蓝色的数据，再次来迭代 $D(x)$ ，最终不断的迭代收敛到 $D(x)$ ， $D(x)$ 分布和真实分布相近。

1.4 小结

单纯的生成器：很容易生成图片，因为就是一个前向输出过程。但是它不考虑像素之间的联系（只学到表象，没学到精神或全局信息）；单纯的判别器：可以学习考虑大局，但是很难生成图片（要去解一个 argmax 的问题，要做一些线性假设，线性则限制了它的能力，非线性无法解）。

而 GAN 就可以解决这个问题，它用 Generator 来代替了 $\arg \max D(x)$ 的过程。

从判别器角度：过去无法求解 argmax 的过程由生成器来取代；

从生成器角度：虽然在生成图片过程中的像素之间依然没有联系，但是它的图片好坏是由有大局观的判别器来判断的。从而能够学到有大局观的生成器。

2 Conditional GAN

2.1 Conditional GAN 基本思想

前面讲到了 GAN 的基本学习方法，大家知道了 GAN 就是生成一张大家都没见过的新图片。这实际上并没有什么用。真正有用的 Conditional GAN。就比如给你一句话，“a dog is running”，要你生成一张图片。

以此图为例，输入是“train”，输出是一张火车的图片，希望得到一个张火车的图片，可火车有正面有侧面，有不同的形态，如果不加约束的话，则可能会产生一张包含各种样式的火车的平均（模糊）图。

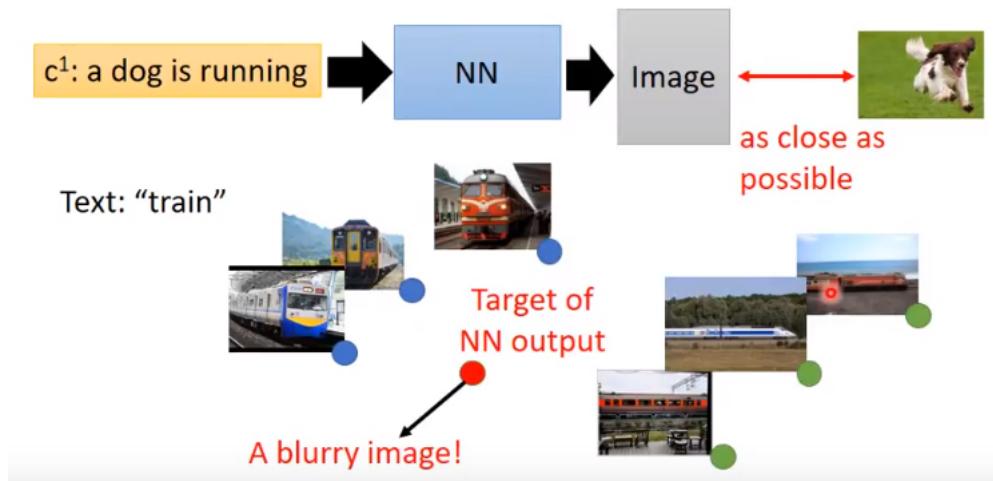


图 10: Traditional supervised approach 训练中的问题

而传统的 GAN 中，又有什么样的问题呢？GAN 的机制是只要识别器能分辨得出一张图是人工生成的还是原图就行，所以不能确保输入火车就会生成火车图片，比如输入火车生成一张很逼真的猫图，生成器也会被认为是一个成功的生成器 (GAN 容易钻空子，即如果某种特定模式可以骗过生成器，生成器会倾向于生成这种模式)。

所以说，Generator 输出一张图片和条件，然后将同样的语句（条件）和生成的图片一起输入识别器，关键是，识别器不仅需要分辨真假，还需要分辨出语句（条件）和图片是否匹配。总的来说有三种情况：1. 生成的 x 图片逼真，且和条件匹配，得高分；2. 生成的 x 图片不逼真，得低分；3. 生成的 x 图片逼真，但是不和条件匹配，得低分。

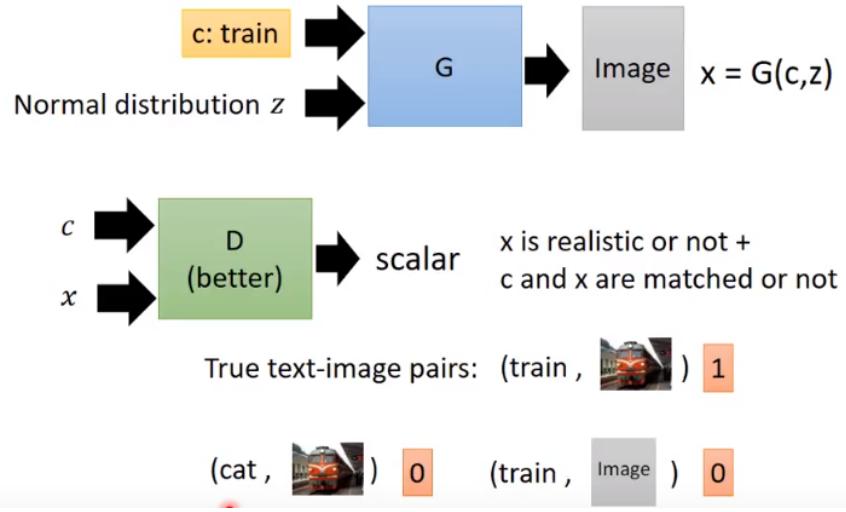


图 11: Conditional GAN 的评分体系

算法流程，如下所示，

- In each training iteration:
 - Sample m positive examples $\{(c^1, x^1), (c^2, x^2), \dots, (c^m, x^m)\}$ from database
 - Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from a distribution
 - Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(c^i, z^i)$
 - Sample m objects $\{\hat{x}^1, \hat{x}^2, \dots, \hat{x}^m\}$ from database
 - Update discriminator parameters θ_d to maximize
 - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(c^i, x^i)$
 - $+ \frac{1}{m} \sum_{i=1}^m \log (1 - D(c^i, \tilde{x}^i)) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(c^i, \hat{x}^i))$
 - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$
 - Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from a distribution
 - Sample m conditions $\{c^1, c^2, \dots, c^m\}$ from a database
 - Update generator parameters θ_g to maximize
 - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log (D(G(c^i, z^i)))$, $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$

图 12: Conditional GAN 算法流程

2.2 Conditional GAN 基本框架

这两种结构的区别主要在生成器的输出部分，实际上下图中下半部分的结构更为合理，因为下半部分的结构中生成器的输出分为两部分：

1. 是否是真实图片。
2. 输出与条件是否配对。

这样做的好处是，直观上说，模型能分别知道两个任务的性能，更有针对性地提高模型能力，如果是输出的图像不够真实，但能比较好地识别出是否匹配，则主要调节生成样本的分支，而第一种结构由于是混合在一起的，就没有这种针对性调节的能力。

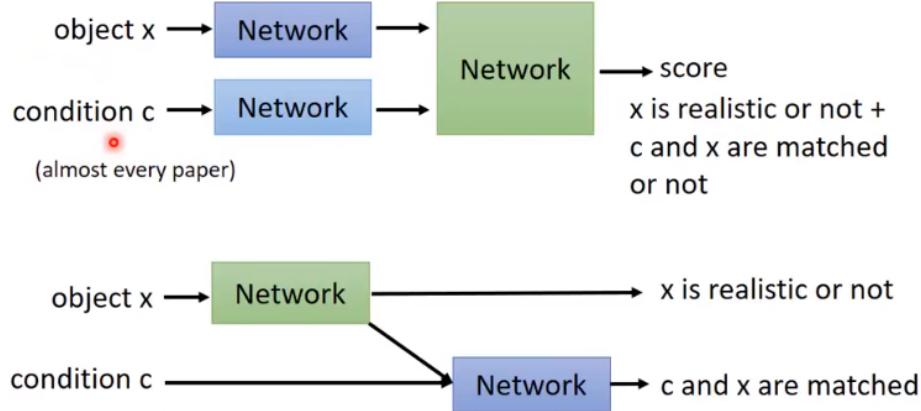


图 13: Conditional GAN 算法设计框架

另外一种极为 Stack GAN，其主要思想是先产生小张的图，然后利用小张的图来生成大图。

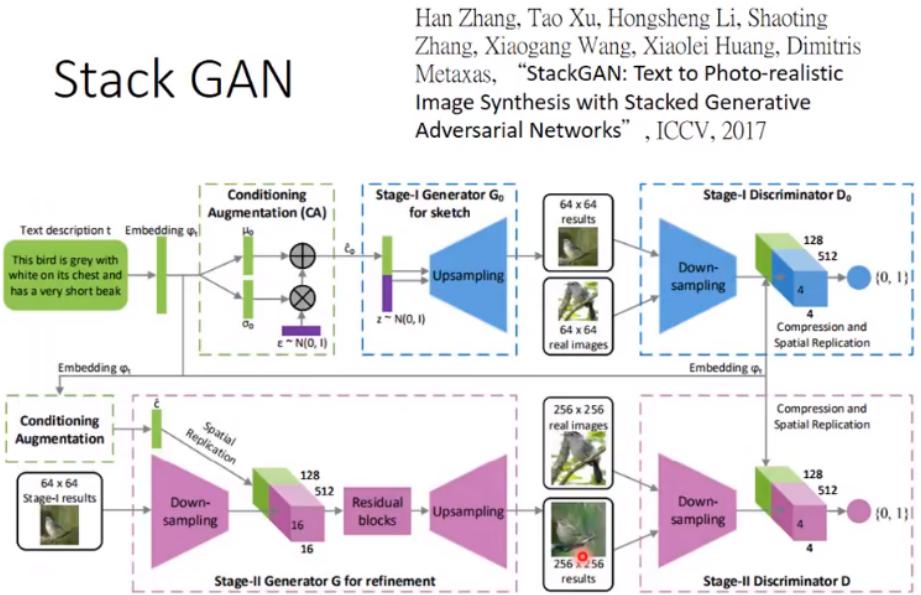


图 14: Stack GAN 算法设计框架

2.3 Conditional GAN 的应用场景

下图是一个简略图生成建组图的例子，主要是对比了三种实验结果。第一种是只使用类似于 AE 的方法，只希望生成样本和目的样本尽量接近，这样做的话会导致输出样本整体显得模糊，因为单纯的像素尽量接近，没把法提供足够的高频信息。第二种是只使用 CGAN，这样的输出相对于只使用像素尽量靠近的方法，画面会精细很多，因为识别器对与模糊和精细的图像有较大的分辨能力，且能从

全局角度考虑问题。但是问题来了 GAN 总喜欢自己创作，觉得输出的图像只要能骗过识别器就行了，所以中间的图像会有一些莫名其妙的地方。第三个实验是混合两种方法，使用 CGAN 的同时，在生成器的末端加入一个 L1 损失（像素尽量靠近），这样的输出既能保持清晰，画面也不会有不合实际的创作部分。这里的例子应该取材于 CGAN 的论文，原文中的模型并没有输入噪声（生成器的输入只有一个箭头），而是靠 dropout 提供噪声，但论文里也说，这种输入噪声的方法效果不强，即输出的多样性不强，这也是接下来可以深入的工作之一。

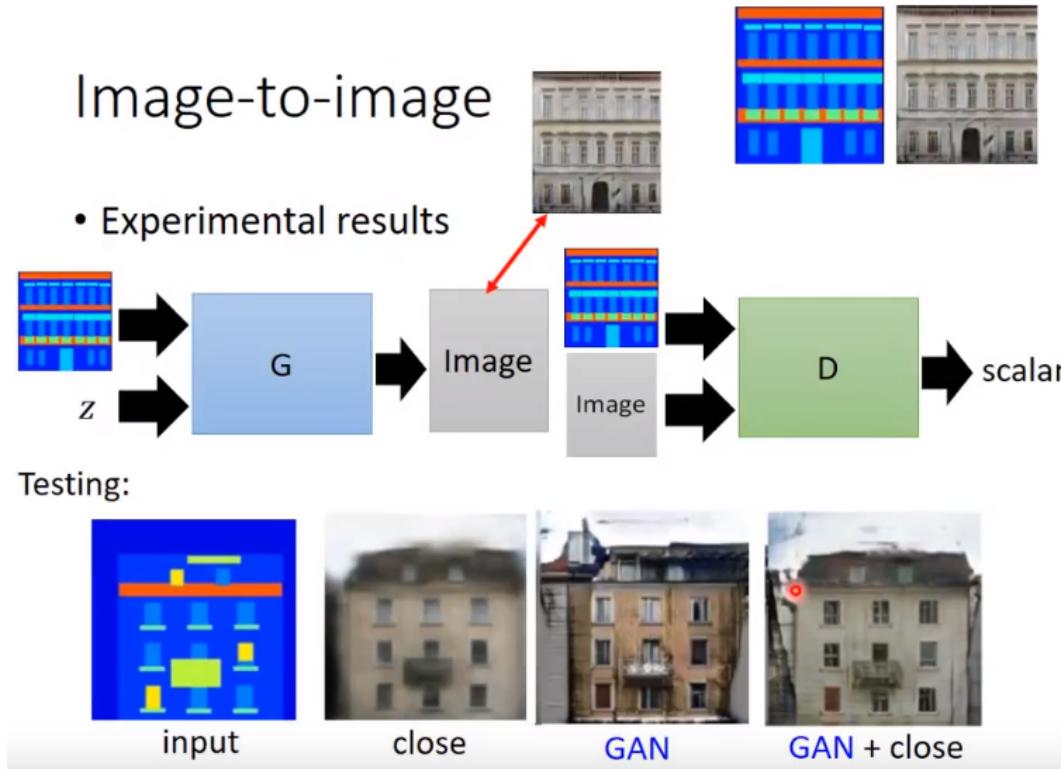


图 15: GAN 实验效果举例

这就是 image to image 的论文，但是其中搞的 Discriminator 经过了稍微的设计，

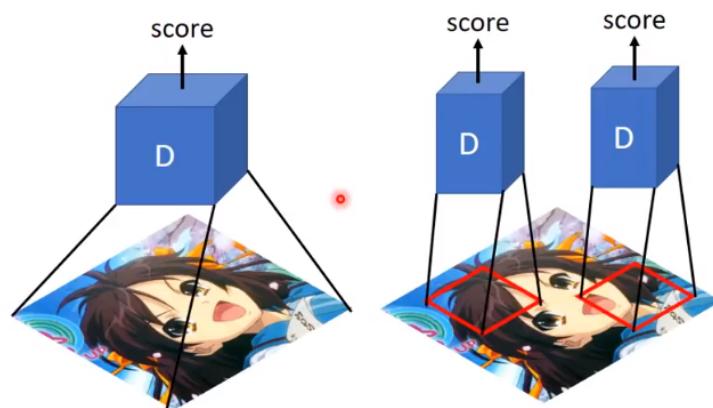


图 16: Patch GAN 基本思想

如果，图片太大了直接判断这张图片好还是不好，太难了，所以基础思路是，把图片切割开，一个 Discriminator 看一个小部分。

其中，Condition GAN 也可以用来做 Speech Enhancement。这里也需要确定，output 和 input 是否是 match 的。

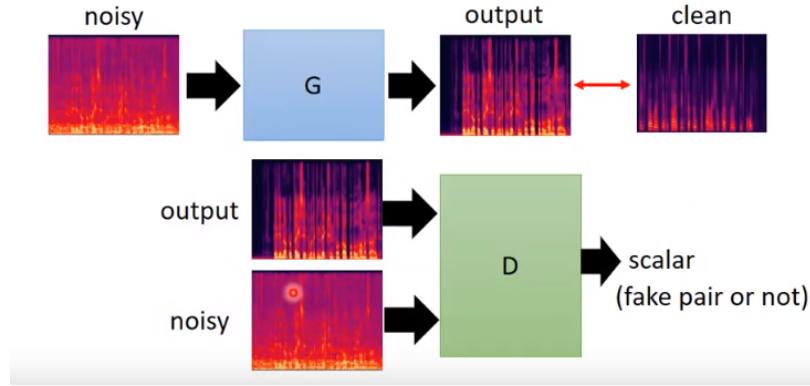


图 17: 用 GAN 实现 Speech Enhancement 基本方法

同样也可以做 Video Generation 来比如知道目前几帧的情况，预测下一帧会是什么。如果采用传统的监督学习会有什么问题呢？还是和前面火车的例子一样，数据集中对于同一个状态，可能会产生多个动作，而这些动作都是对的，那么就可以会导致最终训练出来的结果是这些动作平均之后的结果。

3 Unsupervised Conditional GAN

在 2 小节中讲到的 Conditional GAN 都是 supervised 的，比如由文字生成图片，都是监督学习下的，每一张图片都有对应的 label。而这一节主要想讲的是 Unsupervised 下的 Conditional GAN。

以风格迁移（领域迁移）为例：比如你有一堆 domain X 的图片如风景照，然后你有一堆 domain Y 的图片如复古照；但是它们彼此之间是没有联系的，彼此不是匹配的。你想要将任一一张风景照变成与之相匹配的复古照。大致的解决方法有两种，直接转换 (Direct Transformation) 和投影到公共空间 (Projection to Common Space)。

直接转换：直接学习一个 Generator，可以将 domain X 转换为 domain Y 的东西，主要是用于小改。

投影到公共空间：首先学习一个 domain X 的 encoder，将特征抽出来；之后输入 domain Y 的 decoder，将特征转换为想要的东西；(**适用于输入与输出差距很大**)

3.1 Direct Transformation

3.1.1 基本思想

注意，此时的场景设定是，要 learn 一个 Generator，它可以将 X domain 的图片转换为 Y domain 的图片；现在 X domain 的图片有一堆，Y domain 的图片有一堆，但是 X, Y 并不是一一对应的。因为输入和输出不是对应的，没办法直接使用一些监督学习方法里的损失函数或技巧。而且对于一张风景

图片，我们只是想将它转变成梵高的风格，而不是将它变成梵高的星空或梵高的自画像，所以，我们既要转变图片的风格，又要保持原始信息，防止图片完全变样。那生成器怎么知道给定 X domain 的图片，然后输出 Y domain 的图片呢？基本想法如下所示，

1. 这个时候就需要一个 Y domain 的判别器，这个判别器他看过很多 Y domain 的图片，所以给他一张 image，他可以鉴别这个 image 是不是 Y domain 的 image。
2. 生成器要做的就是想办法去骗过判别器，这样就生成 Y domain 的东西。
3. 但是还要注意生成器生成的图片要与输入有关；因此生成器不仅要骗过判别器，还要保证让输出的图片与输入图片有关。

3.1.2 Cycle-GAN

那么，下一个问题是如何令输入图片和输出图片相关呢？

1. 无视这个问题，直接做下去：因为生成器的输入和输出其实不会差太多；[Tomer Galanti, et al. ICLR, 2018] 中提到过这个问题：如果生成器网络很浅，输入和输出会特别像，这时候就不需要额外的处理；如果你的生成器网络很深，输入和输出会真的很不一样，这时候就需要额外的处理；
2. 用一个已训练好的网络，如 VGG；把生成器的输入和输出 image，统统丢给这个 pre-train 好的网络，就会输出两个 embedding vector；接下来在训练的时候，一方面生成器要骗过判别器，同时希望两个 embedding 的 vector 不要差太多。

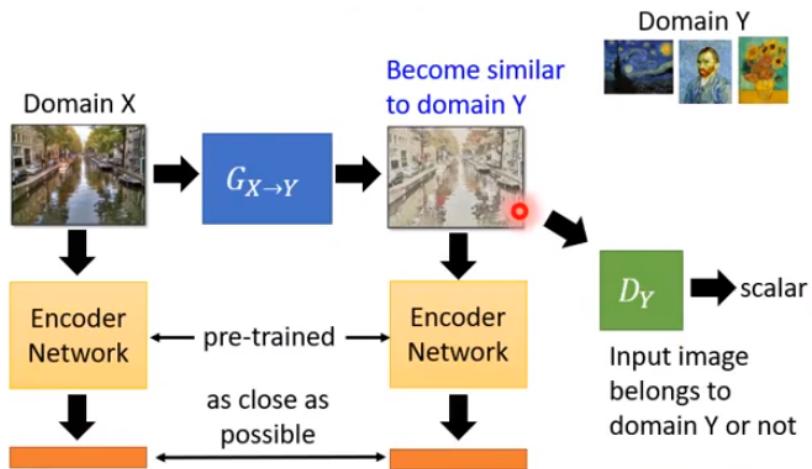


图 18: Direct Transformation 基本方法 [Yannick Taigman, et al., ICLR, 2017]

3. 在处理此问题上比较出名的是 Cycle-GAN。如果想得到一个 X domain 到 Y domain 的生成器，你需要同时 train 一个 Y domain 到 X domain 的生成器；第一个生成器用于将风景画变成复古画，第二个用于将复古画还原为原来一模一样的图；然后目标是两次转换以后的图越接近越好。（这个路径叫做 cycle consistency）同时训练 Y domain 的判别器；重复上面的 X domain 转换为 Y domain；我们再训练一个 Y domain 到 X domain 同样的结构；这就是 cycle-GAN。这样训出来的 GAN 就会比较强大。这样做好处是，如果 Generator 将原图像转换为完全不相关的头像（原始信息都被完全改变了），就没办法将图像再转回到原来的域，这样就保证了原始信息不会丢失。Cycle-GAN 的模型示意图如下所示，

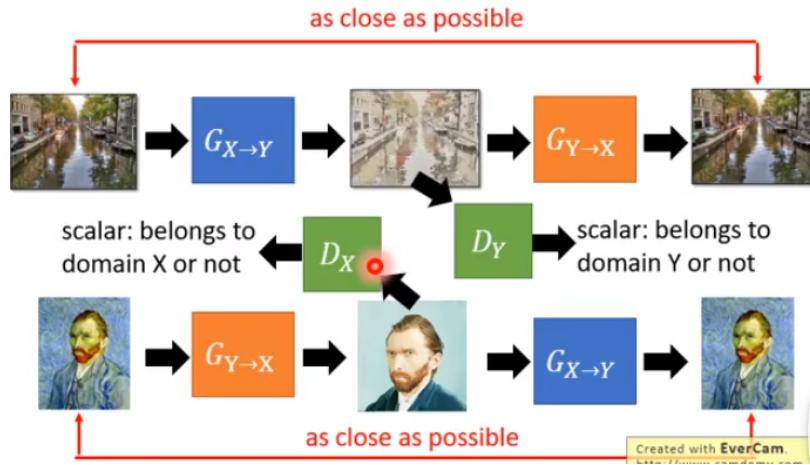


图 19: Cycle-GAN 基本框架

3.1.3 Issue of Cycle Consistency

Cycle-GAN 中的致命缺陷就是 Issue of Cycle Consistency，被李老师称为隐写术问题。cycle-GAN 会把输入的东西藏起来，然后再 output 的时候再呈现出来；例子：输入和输出很像屋顶上都有黑点，但是中间 image 屋顶的黑点却没有。中间产物没有黑点，那么怎么通过中间层 image 得到有黑点的屋顶，一个解释是生成器有很大的能力将信息藏在中间 image 里面，比如数值很小，或者和周围的像素非常的像，让大家分辨不出来。因为就是这种结构就是为了让第一个生成器的输出和输入不要差太多，但是如果 Cycle-GAN 自己藏信息的，就会导致中间层和输出差距很大。这种结果，机器可能自己会学习隐藏东西，结构失去了最开始的意义。但是，个人感觉这个问题也不是非常严重，哈哈哈哈！

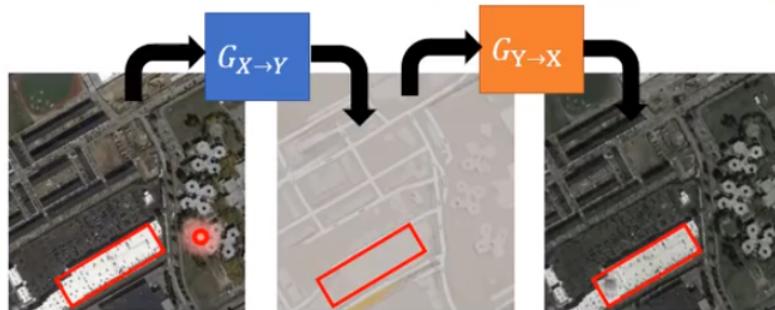


图 20: Issue of Cycle Consistency 问题示意图

3.1.4 Star GAN

Star GAN 是用来做多个 domain 之间的转换，比如 4 个 domain 之间的转换，按照 Cycle-GAN 的思路，应该学会 $2 \times C_4^2 = 12$ 个生成器；而 Star GAN 只需要学会一个生成器就可以在多个 Domain 之间相互转换。

而具体做法怎么做呢？

- 在 star-GAN 中，要学习一个判别器，这个判别器要做两件事：首先对于一张 image，它要鉴别它的真假；还要判断这个 image 来自于哪一个 domain；

- 只需要学习一个生成器，它的输入是一张 image，以及目标 domain；就生成出目标 domain 的 image；
- 然后再将生成的图片输入同一个生成器，输入是原始图片的 domain 以及生成的 image 然后通过这个生成器重建出新的 image，希望 input 的 image 和重建的 image 越接近越好。这实际上就是 Cycle-GAN 的 loss，经过两次转换还原成一样的 image。
- 然后判别器要做两件事：首先对于一张 image，它要鉴别它的真假；还要判断这个 image 来是不是我们要的目标 domain；

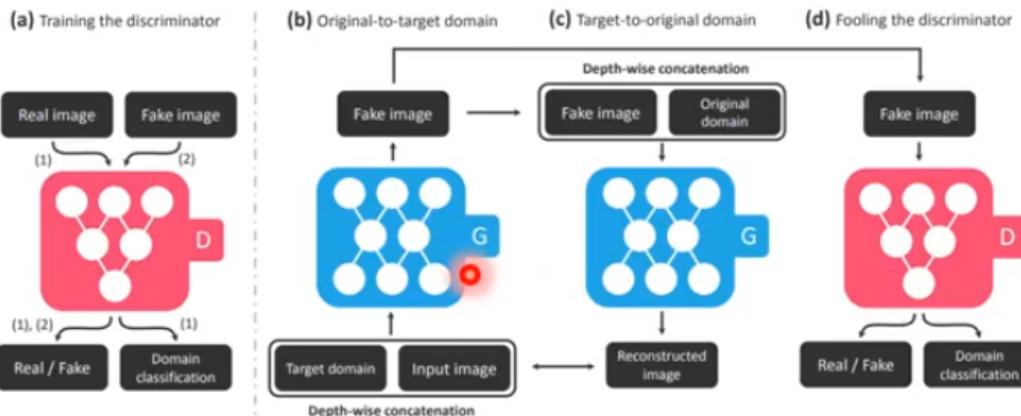


图 21: Star GAN 基本框架

下面描述的是 Star-GAN 的一个实例。下面是实际的例子：判别器做的事是首先判断它是 real 的还是 fake；然后还有判断它是不是目标的 domain；domain 的表示是很多属性编码的，目标 domain 中：第一个 10011 黑色头发男性年轻、00000 表示没有情绪；原始 domain 中：00101 表示棕色女性年轻，00000 表示没有情绪；首先让生成器根据棕色女和目标 domain 将图片转成黑色男，然后黑色男和原始 domain 通过同样的生成器转换为棕色女然后希望输入和输出的棕色女越接近越好。接下来要确保输出的黑色男 image 是 real (Discriminator 看不出来) 的并且是和目标 domain 是一样的。

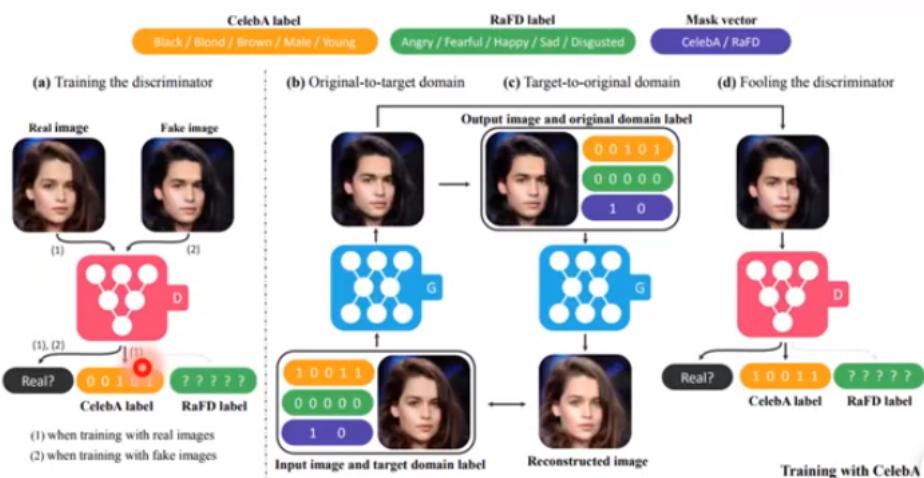


图 22: Star GAN 举例

4 投影到公共空间 (Projection to Common Space)

这种做法的实际想法为生成一个共同的特征空间，不管是从 domain A 到 domain B 还是从 domain B 转换到 domain A，都会经过这个公共层。其主要目的是得到这么一个公用的特征表示层，理想状态是某个维度有特定的特征意义。比如是人脸转换到动漫头像，那么公共空间的某一维或几维的输出就可能表示头发的颜色，经过公共层后，就能生成对应的发色。

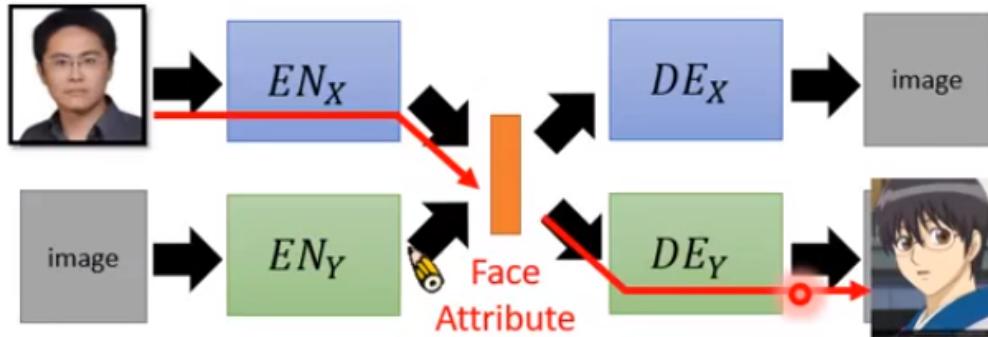


图 23: 投影到公共空间基本架构

那么，训练的方式如下所示：

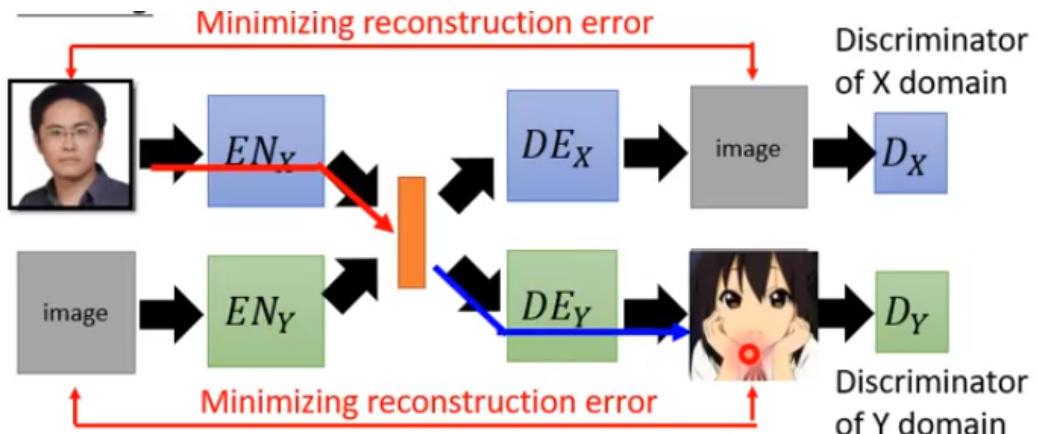


图 24: 投影到公共空间基本实现思路

但是问题马上就来了，因为两个 auto-Encoder 是分开训练的，两者之间并没有什么关系。两个转换的支路没有真正地在公共特征层共享信息。所以，就有可能出现上图所示的尴尬场景。例如当你丢个 X domain 的人脸进去，变成一个 vector，当输入如 Y domain 的 encoder 中时，可能出现一个截然不同的卡通人物。因为是分开训练的，所以在 latent space 中每个维度的表示属性是不一样的，如上面的 auto-encoder 用第一维代表性别，而下面一组则用其他代表性别。那么怎么解决关联性的问题呢？

4.1 Couple GAN, UNIT

第一种改进的方法非常的直观，多设置几层公共特征层，希望公共特征层能学到更多有用的信息，如下图所示，比如 Couple GAN[Ming-Yu Liu,et al; NIPS, 2016]，UNIT[Ming-Yu Liu,et al, NIPS 2017]中是这么考虑的。

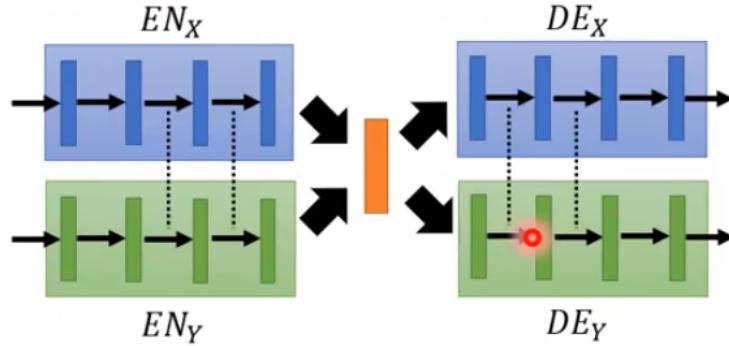


图 25: 部分 Encoder 和 decoder 采用共享的参数

4.2 添加 domain 判别器

原来 X domain 和 Y domain 训练的时候，之间没有关联，现在，用一个 domain discriminator 来判断这个 vector 是来自于 X domain 的 image 还是来自于 Y domain 的 image。两个 encoder 的作用是骗过这个判别器。当无法判断的时候，就是说明两者被 encode 到同一个空间。如下图所示，

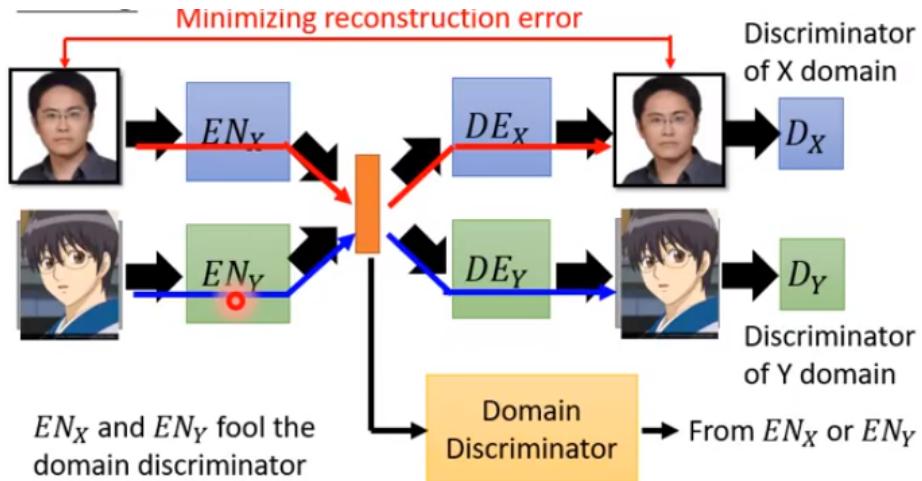


图 26: 添加 domain 判别器

4.3 Cycle Consistency

这个 pipeline 是这样的，对于一张图片 X，经过 X Encoder -> Y Decoder -> Y Encoder -> X Decoder 解回来，让输入和输出越靠近越好。实际上是用 x domain 的图片生成 y domain 的图片，然后用生成的 y domain 的图片，再还原出 x domain 的图片。这个实际上和 Cycle GAN 的思路是一样的，只不过这里没有特别切开，分成 latent space 而已。网络框架如下图所示，

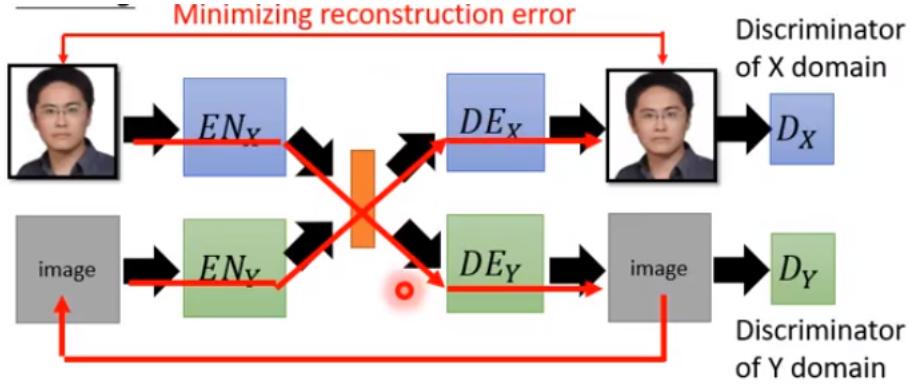


图 27: Cycle Consistency

4.4 XGAN

下面这个网络为 XGAN，与上面的网络不同的地方。是比较 x domain Encoder 生成的特征和经过 y domain Encoder 重构的公共特征的相似度。实际上是直接在高度抽象的 latent space 比较相似度，在这个任务中会更有效，除了对公共特征层每个维度所代表的含义有更严格的要求外，能更有效地抗噪身或者说冗余信息的干扰。

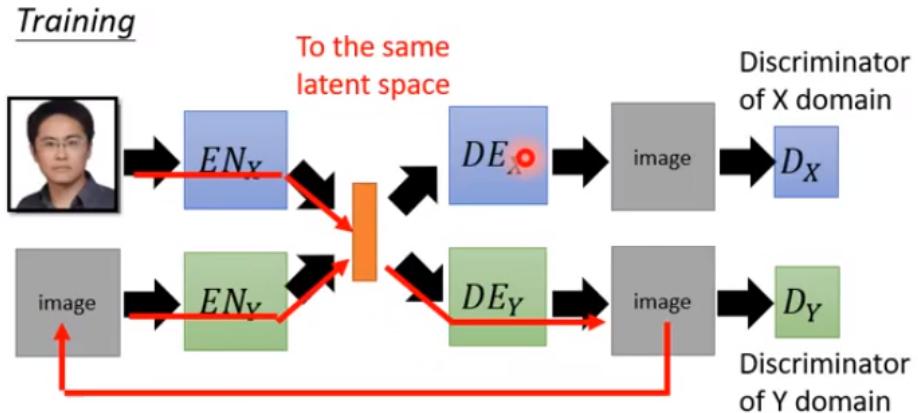


图 28: Semantic Consistency 的 XGAN

此方法也可以在语音领域进行应用，来做语音风格迁移，或者是变声音，变头像等等操作都是一样的。

5 The Basic theory of GAN

GAN 实际上是一种生成模型，其主要目标是寻找关于数据的分布 $P_{\text{data}}(x)$ 。假设目标要生成的是 image，用 X 来表示 image，每个 image 都是高维空间分布中的一个点。我们所需要的人脸只是这个高维空间中一个很小的区域，就是一个固定的分布。机器需要做的就是找出这个分布。

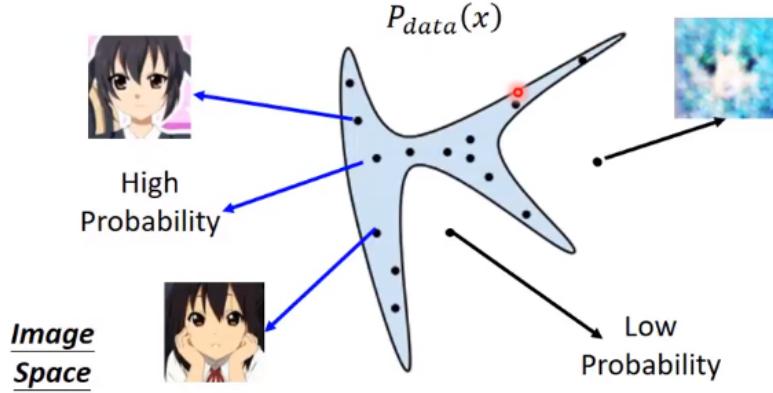


图 29: GAN 的目的是找到数据分布 $P_{\text{data}}(x)$

5.1 Maximum Likelihood Estimation (MLE)

对于一个数据分布 $P_{\text{data}}(x)$, 我们不知道它长什么样, 但是我们可以从中采用出一些样本。虽然, 不知道分布到底长什么样, 可以假设参数化分布 $P_G(x; \theta)$, 让 $P_G(x; \theta)$ 去近似 $P_{\text{data}}(x)$ 即可。比如, 可以假设 $P_G(x; \theta)$ 为 Gaussian Mixture Model, θ 表示 Guassian 的均值和方差。假设从 $P_{\text{data}}(x)$ 中采样得到 $\{x^1, x^2, \dots, x^m\}$, 可以计算出 $P_G(x^i; \theta)$, 生成样本的似然函数为,

$$L = \prod_{i=1}^m P_G(x^i; \theta) \quad (2)$$

目标是找到参数 θ^* , 使似然函数最大。而事实上 MLE 等价于 Minimize KL Divergence,

$$\begin{aligned} \theta^* &= \arg \max_{\theta} \prod_{i=1}^m P_G(x^i; \theta) = \arg \max_{\theta} \log \prod_{i=1}^m P_G(x^i; \theta) \\ &= \arg \max_{\theta} \sum_{i=1}^m \log P_G(x^i; \theta) \\ &\approx \arg \max_{\theta} E_{x \sim P_{\text{data}}} [\log P_G(x; \theta)] \\ &= \arg \max_{\theta} \int_x P_{\text{data}}(x) \log P_G(x; \theta) dx - \int_x P_{\text{data}}(x) \log P_{\text{data}}(x) dx \\ &= \arg \min_{\theta} \text{KL}(P_{\text{data}} \| P_G) \end{aligned} \quad (3)$$

问题马上就来了, 我们常常要先假定一个具体的分布去逼近实际分布; 但是有很多限制, 我们需要一个通用的分布, 如何定义分布 P_G 去逼近这个复杂的分布 P_{data} ; 如何得到通用的分布呢? GAN 中的生成器登场了。

5.2 GAN 得到数据分布

通过一个复杂的网络来表示概率分布 P_G 。然后我们的目的是让生成器生成的通用分布 P_G 和实际图片分布 $P_{\text{data}}(x)$ 之间的 KL 散度最小。如下所示,

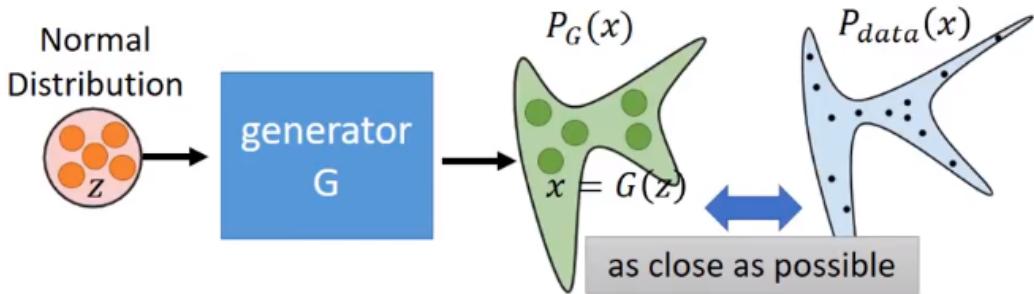


图 30: GAN 的基本思路

其中, $G^* = \arg \min_G \mathbb{D}(P_G, P_{data}(x))$, \mathbb{D} 表示 Divergence。那么, 下一个问题马上来了, 两个分布的 formulation 我们都不知道, 那么如何计算 Divergence? 那么, 先直观的感觉一下, GAN 是如何近似 Divergence 的。我们虽然不知道 P_G 和 $P_{data}(x)$ 是什么样的, 但是我们可以从 P_G 和 $P_{data}(x)$ 中 sample 一些样本出来呀。Discriminator 化身成了衡量 Divergence 的有效工具。

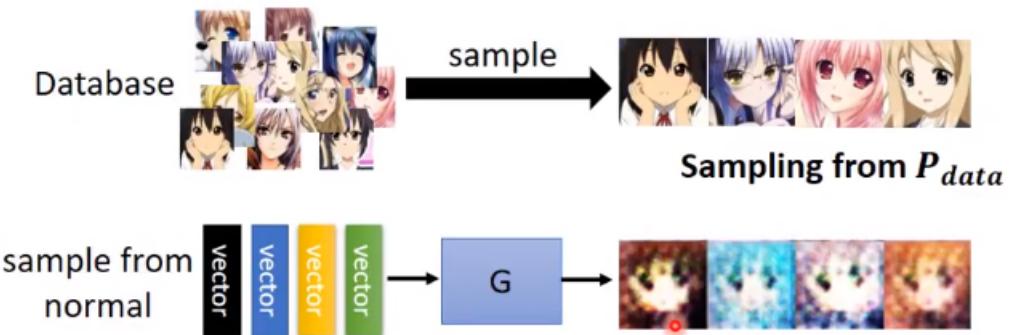


图 31: Sample from P_G and $P_{data}(x)$

然后, 通过判别器可以判断两个分布之间的距离。一般来说, 我们要训练一个判别器, 我们会用一个损失函数: 首先把 m 张真实图片都拿出来, 经过判别器得到分数, 然后经过 \log 再统统平均起来(当然希望这个越大越好, 因为希望真实的图片得分高); 对于生成器生成的 m 张图片当然希望值越小越好。Discriminator 输出的结果就表示两个分布间的距离。

$$V(G, D) = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))] \quad (4)$$

那么, $D^* = \arg \max_D V(D, G)$ 。这种过程其实就是相当于训练一个二元分类问题: P_G 相当于类别 1; P_{data} 相当于类别 2; 然后使他们的交叉熵损失最小。下面将给出证明, $\max V(D, G)$ 得到的判别器, 实际和 JS 散度有关。

5.3 判别器与散度之间的关系

下面将给出 $\max_D V(G, D)$ 这是个什么东西。对于 G 固定的情况。

$$\begin{aligned} V &= E_{x \sim P_{\text{data}}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))] \\ &= \int_x P_{\text{data}}(x) \log D(x) dx + \int_x P_G(x) \log(1 - D(x)) dx \\ &= \int_x [P_{\text{data}}(x) \log D(x) + P_G(x) \log(1 - D(x))] dx \end{aligned} \quad (5)$$

对于给定的 x , 最优化结果 D^* , 为最大化,

$$P_{\text{data}}(x) \log D(x) + P_G(x) \log(1 - D(x))$$

对 $D(x)$ 求导, 可以求解出,

$$D^*(x) = \frac{P_{\text{data}}}{P_{\text{data}} + P_G} \quad (6)$$

那么, 可得,

$$\begin{aligned} \max_D V(G, D) &= V(G, D^*) \\ &= E_{x \sim P_{\text{data}}} \left[\log \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_G(x)} \right] + E_{x \sim P_G} \left[\log \frac{P_G(x)}{P_{\text{data}}(x) + P_G(x)} \right] \\ &= \int_x P_{\text{data}} \log \frac{\frac{1}{2}P_{\text{data}}(x)}{\frac{1}{2}(P_{\text{data}}(x) + P_G(x))} dx + \int_x P_G \log \frac{\frac{1}{2}P_G(x)}{\frac{1}{2}(P_{\text{data}}(x) + P_G(x))} dx \\ &= -2 \log 2 \int_x P_{\text{data}} \log \frac{P_{\text{data}}(x)}{\frac{1}{2}(P_{\text{data}}(x) + P_G(x))} dx + \int_x P_G \log \frac{P_G(x)}{\frac{1}{2}(P_{\text{data}}(x) + P_G(x))} dx \\ &= -2 \log 2 + \text{KL} \left(P_{\text{data}} \parallel \frac{P_{\text{data}} + P_G}{2} \right) + \text{KL} \left(P_G \parallel \frac{P_{\text{data}} + P_G}{2} \right) \\ &= -2 \log 2 + \mathbb{D}_{\text{JS}}(P_{\text{data}} \| P_G) \end{aligned} \quad (7)$$

所以得到了理论结果, **判别器的最大化过程就是求 JS 散度。实际上如果把目标函数写的不一样, 可以等价为其他不同的散度!** 在每一次训练过程中, 判别器的训练是求此时生成器的 P_{data} 和 P_G 的散度; 然后生成器再一次训练, 来最小化此时的散度, 得到新的 P_G , 于是得到了,

$$G^* = \arg \min_G \max_D V(G, D) \quad (8)$$

可以观察如下事例, 假设这个世界上只有三种生成器: G_1, G_2, G_3 。分析的结果, 我们首先要是 V 函数最大化, 也就是下图中的三个红点。然后下一步则是最小化 G , 比较三个 V 函数最大值中最小的那个, 即为 G_3 。

具体的算法:

1. 首先固定生成器, 找到一个能够使 V 最大的 D ;
2. 然后固定 D , 找到能够使这个最大 D 情况下 V 最小的 G 。不停的迭代....

下面将从数学看为什么这个算法是在解上面的最小最大过程:

已知, $G^* = \arg \min_G \max_D V(G, D)$, 可以将其看为, $G^* = \arg \min_G L(G)$ 。可以采用梯度下降的方法寻找 G^* , 令 θ_G 表示为 G 的参数。

$$\theta_G \leftarrow \theta_G - \eta \frac{\partial L(G)}{\partial \theta_G} \quad (9)$$

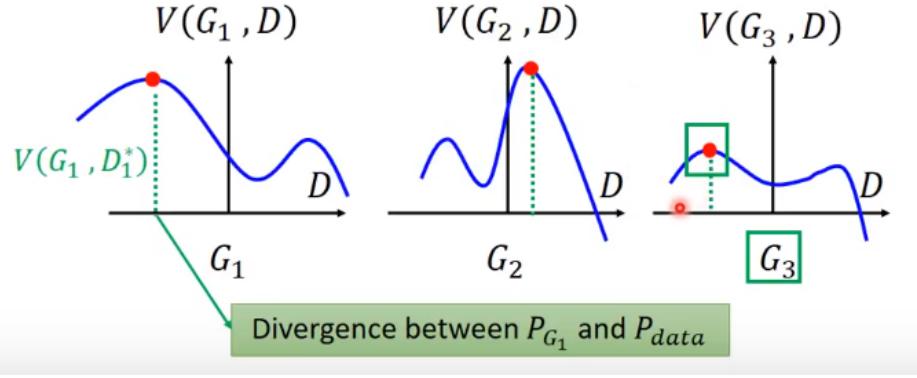


图 32: min-max 求解过程

然后, 在 $L(G)$ 中有一个 max 操作, 它是否依然是可微的呢? 主要问题是考虑当存在最大化操作的算子的时候, 我们是否可以求其微分。假设, $f(x) = \max\{f_1(x), f_2(x), f_3(x)\}$ 。那么, 实际 $\frac{df(x)}{d(x)} = \frac{df_i(x)}{dx}$, 其中, $f_i(x)$ 是最大值, 如下图所示。先看哪个子函数在这个点最大, 微分值就是最大的那个子函数的微分值。也就是梯度下降依然适用, 就是每次更新参数的时候先看自己在那个范围, 再用这个范围的对应的函数值最大的函数求梯度, 然后更新。之后, 不断的重复这个过程,

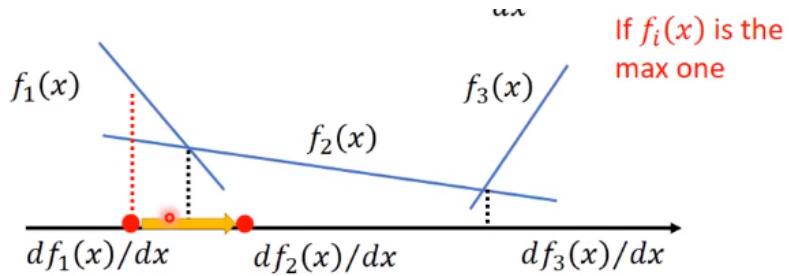


图 33: 求 $f(x)$ 梯度

那么在 GAN 中, 1. 我们首先有个初始的 G_0 ; 2. 然后我们要算 G_0 对 $L(G)$ 的梯度。但是由于 $L(G)$ 有 max 操作所以我们不知道 $L(G)$ 长什么样子; 我们要把 max 的 D 找出来; 3. 就可以更新 G 的参数得到 G_1 ; 如下图所示,

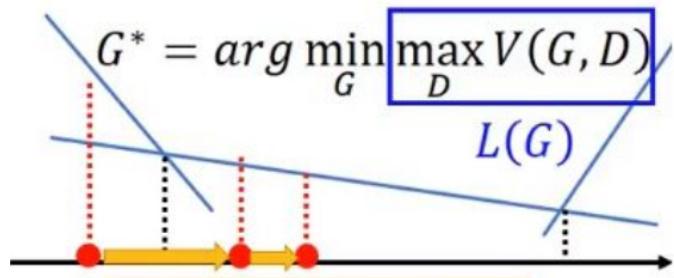


图 34: GAN 训练过程

实际上, 横坐标可以看成是 G , 纵坐标看成 $V(G, D)$ 。在横坐标某个位置, 如图中红点的位置, 首先找 $V(G, D^*)$, 然后对 D 求梯度。

但是，遗憾的是，在 train 的过程中，并不是真正的 minimize JS 散度，为什么呢？刚开始的第一步，先找一个 G_0 ，然后固定 G 找 G_0 下，令 V 函数最大 D ，此时的 $V(G_0, D^*)$ 就是 G_0 下的散度；当 G 在 train 时变成了 G_1 ，函数变成了 $V(G_1, D^*)$ ；此时由于 D 固定，所以 JS 散度会变得不再是此刻 G 下的 JS 散度了。因为函数发生了改变， D^* 不一定是新函数的最大值，可能存在 $V(G_1, D') > V(G_1, D^*)$ 。所以，在固定 D 训练 G 的过程中，非常有可能训着训着 G 的改变较大了，此时的 $V(G, D^*)$ 就不是 JS 散度了。如下图所示，

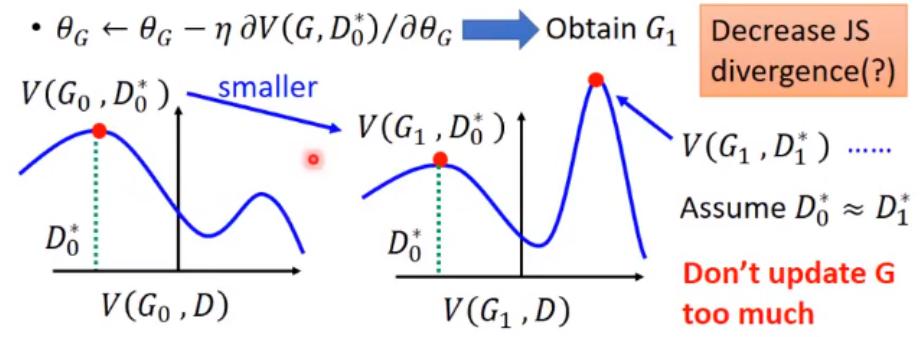


图 35: GAN 训练过程中 JS 散度不匹配的问题。

因此前提假设是： $V(G_0, D)$ 和 $V(G_1, D)$ 是很像的；(G 的参数变化很小) 因此在 train 生成器的时候不能更新的太多，但是在 train 判别器 D 的时候要 train 到底，因为这是是在找当前 G 的 JS 散度。

5.4 GAN 的实际操作

实际是没办法求期望的，都是用 sample 来代替。

那么，对应 G ，首先要计算 $\max_D V(G, D)$ 。 $\{x^i\}_{i=1}^m \sim P_{\text{data}}(x)$, $\{\tilde{x}^i\}_{i=1}^m \sim P_G(x)$ 。

$$\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i)) \quad (10)$$

实际上这就是二分类问题，最小化 Cross-Entropy 是一样的。最终的训练算法如下所示，首先 train 判别器 D ，实际上没办法 train 到 $V(G, D)$ 收敛，可以定义训练 k 次；(JS 散度是什么) 之后 train 生成器 G ：其中下图中第一项是与生成器无关的，所以直接不用训练了，由于 G 不能训练太多，所以 updata 一次就好（最小化 JS 散度）。所以这下应该大家能理解，为什么 D 需要训练很多次，而 G 只训练一次的原因了吧！

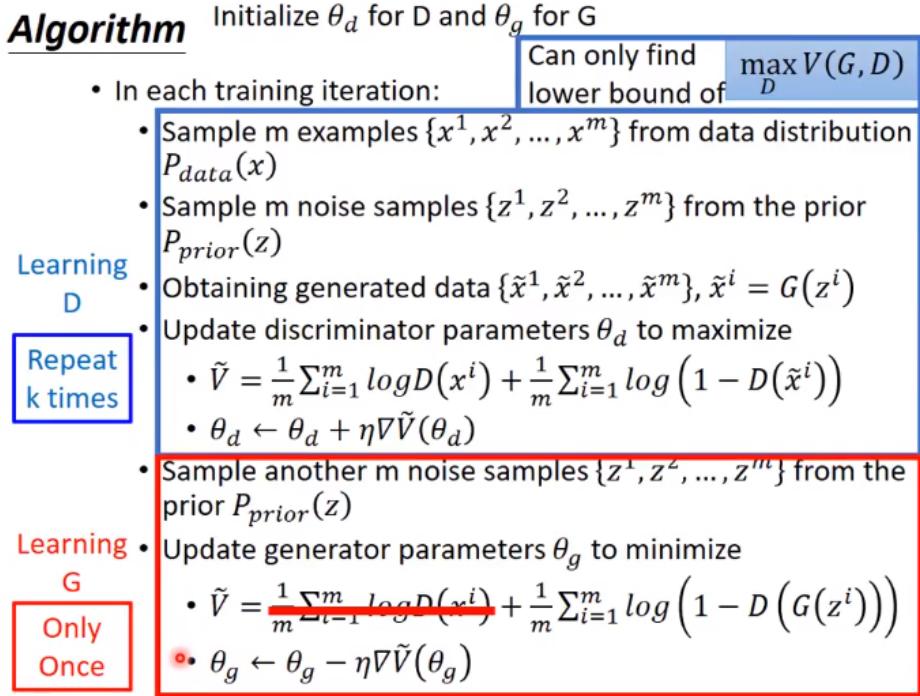


图 36: GAN 的算法流程

6 f -GAN: General Framework of GAN

前面章节已经清晰的描述了, $\max_D V(G, D)$ 的过程实际上为近似 JS 散度, 那么定义不同的 $V(G, D)$ 函数形势, 可以是求其他不同的散度, 来测量不同的距离, 任何的 f 散度都可以融入到 GAN 中去。 f -散度可以写出统一的表达从而推出各种不同散度下的 GAN。而为什么实际运用中 f -GAN 没有什么用呢? 因为经过大量的实验表明, 虽然这个理论很漂亮, 但是使用不同的散度来度量分布之间的距离, 所得到的实验效果差不太多。

假设 P 和 Q 是两个分布, 其中 $p(x)$ 和 $q(x)$ 表示从分布中采样出 x 样本的概率。其中, f 散度定义为,

$$\mathbb{D}_f(P\|Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx \quad (11)$$

其中, 定义 $f(1) = 0$, f 函数为凸函数。那么有

$$\begin{aligned} \mathbb{D}_f(P\|Q) &= \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx \\ &\geq f\left(\int_x q(x) \frac{p(x)}{q(x)}\right) = f(1) = 0 \end{aligned} \quad (12)$$

其中, 第一行到第二行用到了 Jense 不等式。所以, $\mathbb{D}_f(P\|Q) \geq 0$, 当且仅当 $P = Q$ 时取得等号。实际

$f(x)$ 取不同的值, 可以得到不同的 Divergence, 下面分别是 KL 散度, Reverse KL 散度和 Chi Square。

$$\begin{aligned}
 f(x) &= x \log x \\
 \mathbb{D}_f(P\|Q) &= \int_x q(x) \frac{p(x)}{q(x)} \log \left(\frac{p(x)}{q(x)} \right) dx = \int_x p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \\
 f(x) &= -\log x \\
 \mathbb{D}_f(P\|Q) &= \int_x q(x) \left(-\log \left(\frac{p(x)}{q(x)} \right) \right) dx = \int_x q(x) \log \left(\frac{q(x)}{p(x)} \right) dx \\
 f(x) &= (x-1)^2 \\
 \mathbb{D}_f(P\|Q) &= \int_x q(x) \left(\frac{p(x)}{q(x)} - 1 \right)^2 dx = \int_x \frac{(p(x) - q(x))^2}{q(x)} dx
 \end{aligned} \tag{13}$$

6.1 Fenchel Conjugate

对于任意的凸函数 f , 有一个共轭函数 f^* ,

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\} \tag{14}$$

比如, $f^*(t_1) = \max_{x \in \text{dom}(f)} \{xt_1 - f(x)\}$, 假设我们穷举所有的 x , 计算出 $x_1 t_1 - f(x_1), x_2 t_1 - f(x_2), x_3 t_1 - f(x_3)$, 比如 $x_1 t_1 - f(x_1)$ 是最大的, 那么 $f^*(t_1) = x_1 t_1 - f(x_1)$ 。假设, 设 x 为固定的 t 为变量, 比如先取 $x = x_1$, t 为变量, 那么 $x_1 t - f(x_1)$ 为一次函数, 同理可以画出 $x \in \text{dom}(f)$ 的所有直线。然后当 $t = t_1$ 时, 取出最大值即可, 如下图所示。所以, 可以得到 $f^*(t_1) = \max_{x \in \text{dom}(f)} \{xt_1 - f(x)\}$ 的函数, 即为图中红色的线条。

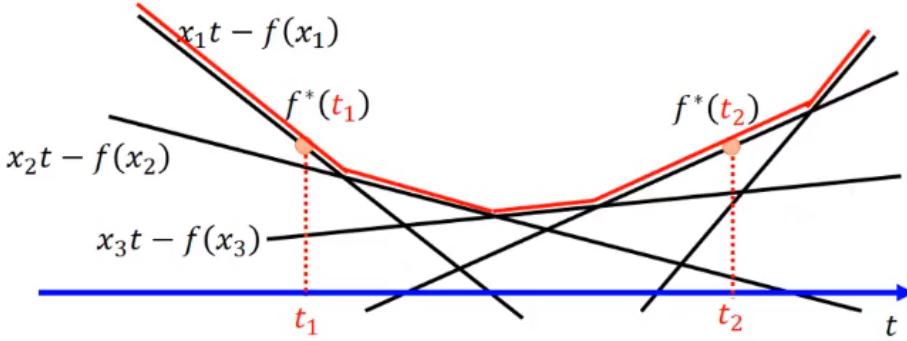


图 37: Fenchel Conjugate 示意图

下面可以举例 $f(x) = x \log x$, f 散度即为 KL 散度。 $f^*(t) = \max_{x \in \text{dom}(f)} \{xt - x \log x\}$, 令 $g(x) = xt - x \log x$ 。对于给定的 t , 需要找到 x 来令 $g(x)$ 最大化。对 x 求导得, $t - \log x - 1 = 0$, $t - 1 = \log x$, 那么有 $x = \exp(t-1)$ 。将其带入有, $f^*(t) = t \exp(t-1) - (t-1) \exp(t-1) = \exp(t-1)$ 。并且有, $f^{**} = f$ 。

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\} \longleftrightarrow f(x) = \max_{t \in \text{dom}(f^*)} \{xt - f^*(t)\} \tag{15}$$

6.2 Connection with GAN

那么大家可能会想，这个和 GAN 有什么关系呢？下一步进一步推导，

$$\begin{aligned}\mathbb{D}_f(P\|Q) &= \int_x q(x)f\left(\frac{p(x)}{q(x)}\right)dx \\ &= \int_x q(x)\left(\max_{t \in \text{dom}(f^*)} \left\{x\frac{p(x)}{q(x)} - f^*\left(\frac{p(x)}{q(x)}\right)\right\}\right)dx\end{aligned}\tag{16}$$

那么，求解 D 的过程就是求解这个函数最大化的过程， D 函数输入是 x 输出是 D ，那么一定有，

$$\begin{aligned}\mathbb{D}_f(P\|Q) &\geq \int_x q(x)\left(\frac{p(x)}{q(x)}D(x) - f^*(D(x))\right) \\ &= \int_x p(x)D(x)dx - \int_x q(x)f^*(D(x))dx\end{aligned}\tag{17}$$

所以， $D_f(P\|Q)$ 可以等价的写为：

$$\mathbb{D}_f(P\|Q) \approx \max_D \int_x p(x)D(x)dx - \int_x q(x)f^*(D(x))dx\tag{18}$$

那么有，

$$\begin{aligned}D_f(P\|Q) &\approx \max_D \int_x p(x)D(x)dx - \int_x q(x)f^*(D(x))dx \\ &= \max_D \{E_{x \sim P}[D(x)] - E_{x \sim Q}[f^*(D(x))]\}\end{aligned}\tag{19}$$

同理，改写一下就有，

$$D_f(P_{\text{data}} \| P_G) = \max_D \{E_{x \sim P_{\text{data}}}[D(x)] - E_{x \sim P_G}[f^*(D(x))]\}\tag{20}$$

而这个和 GAN 的基本理论是一样的，套用一下就有，

$$\begin{aligned}G^* &= \arg \min_G D_f(P_{\text{data}} \| P_G) \\ &= \arg \min_G \max_D \{E_{x \sim P_{\text{data}}}[D(x)] - E_{x \sim P_G}[f^*(D(x))]\} \\ &= \arg \min_G \max_D V(G, D)\end{aligned}\tag{21}$$

也就是说，如果我想使用不同的散度来衡量分布间的距离，只要改 $f^*(D(x))$ 的形式即可。

6.3 Model Collapse & Model Dropping

那么这一招 f 散度有什么用呢？它也许可以用来解决 Model Collapse 问题。所谓模型坍缩就是生成的图片经常在一张图片附近，real data 非常的大，但是 generated 会集中在一个很小的分布附近，如下图所示，大家发现黄色头发的那个卡通人物经常出现。



图 38: Model Collapse 问题示意图

所谓 Model Dropping 指的是，加入 real data 有两堆，而 Generator 却集中在其中某一堆，如下所示，



图 39: Model Dropping 问题示意图

那么，为什么会出现这样的情况呢？一种远古的猜测是可能是 f 散度选的不好，最终的优化结果就是会导致这样的情况的出现。比如，下面两个例子，

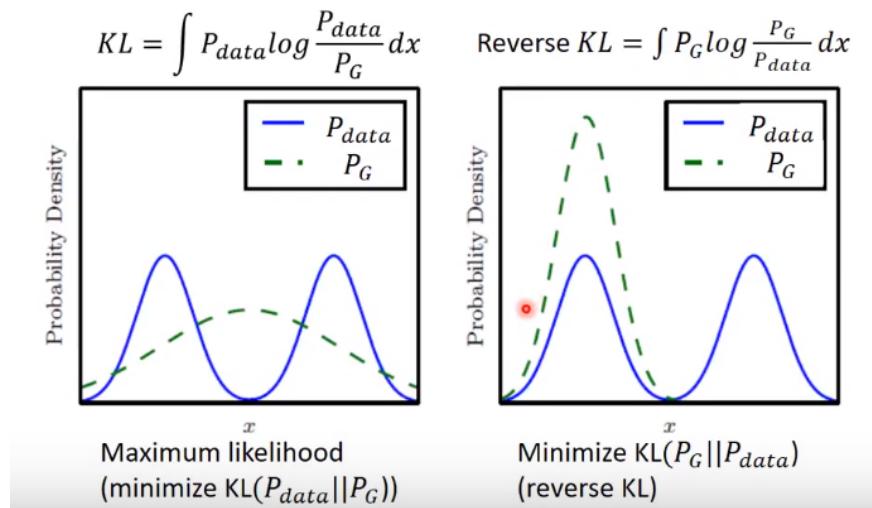


图 40: Model Collapse & Model Dropping 原因分析

P_G 是理想的训练结果，出现 Model Collapse & Model Dropping 的问题反而是正常的。那么，或许我们可以尝试使用不同的散度，来改变一下。但是，这个问题也不是这么简单的，至今也没有非常好的解决方法。

7 Tips for improving GAN

7.1 JS 散度的缺点

本小节描述的是如何让 GAN 更容易训练。最原始的 GAN 度量的是 P_g 和 P_{data} 之间的 JS 散度；但是有一个严重的问题：可能两个分布没有任何的重叠，其主要有两个方面的原因，

1. 一方面的原因是由于 image 在高维空间中的分布其实是低维的流形（二维平面折叠在三维空间中），高维空间中的低维流形几乎是可以忽略的；如下图所示，

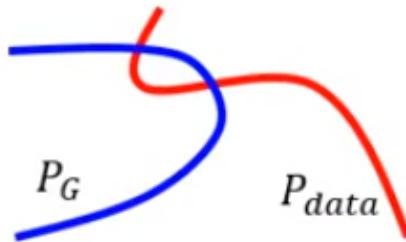


图 41: 高维空间中的低维流形显示

2. 在衡量 P_g 和 P_{data} 的散度的时候，我们是从两个分布中 sample 得到两部分 data，再去计算两个数据集之间的散度，即使计算分布有重叠，但是 data 之间是没有重叠的（我们很难从两个分布中采样到同样的数据），所以完全可以视 data 的得到的分布没有任何交集。

因为只要两个分布没有交集，那么 JS 散度的值一直是 $-\log 2$ 。比如下图中， G_1 显然比 G_0 要好，但是从 JS 散度上根本看不出来。这样的话，训练起来是有问题的；（实际上在训练的时候，生成器的目标是最小化 P_g 和 P_{data} 之间的 JS 散度，然后需要使用判别器来量出 JS 散度，但是对生成器来说 G_0 和 G_1 是一样的，因此生成器不会更新，那怎么训练得到 G_{100} 呢？仔细一想就知道是不可能的。）

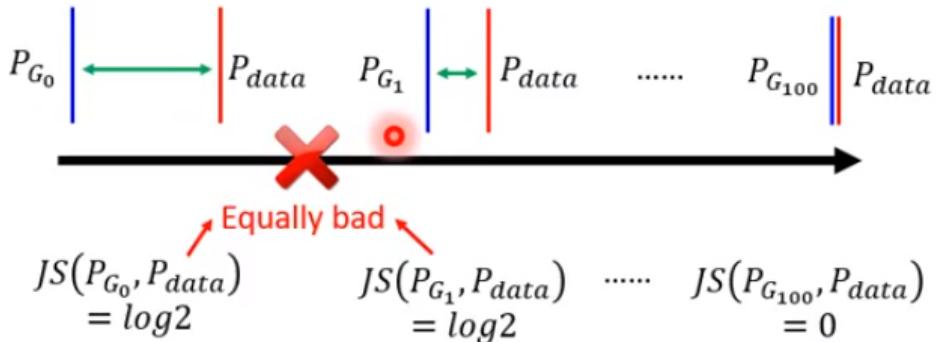


图 42: JS 散度优化流程

这样就导致原始的 GAN 非常难训练。下面介绍几种解决方法。

7.2 Least Square GAN (LSGAN)

首先看一个例子，当你学习的是二元分类器的话，会给蓝色（fake）的点 0 分，绿色（real）点 1 分；output 是 sigmoid 函数，在接近 0 或 1 的地方特别平。我们希望得到的是，当 train 一个生成器时，它会带领蓝色的点顺着梯度去移动分布，但是蓝色的点无法移动，因为它的附近梯度几乎是 0。那么，其解决思路非常的直观，将输出的 Sigmoid 函数换成 Linear 函数，此问题将从分类问题转换为回归问题。正样本越接近 1 越好，而负样本越接近 0 越好。

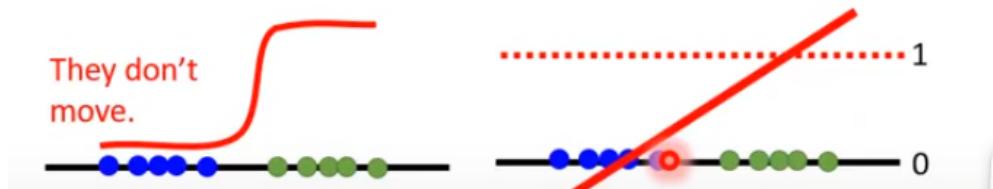


图 43: Least Square GAN (LSGAN) 基本思路

7.3 Wasserstein GAN (WGAN)

于是可以采用另一种方式来衡量 P_g 和 P_{data} 之间的距离：earth mover's distance；按照李老师课程中的解释为，假设有两堆 data P 和 Q ，开着一个推土机，将 P 土推到 Q 土，这个走过的距离就叫 earth mover's distance。那么就会出现一个问题，不同的移动方法虽然最终都可以到达最终的目的，但是移动的方法不一样，最终移动的土的距离是不一样的，如下图所示。

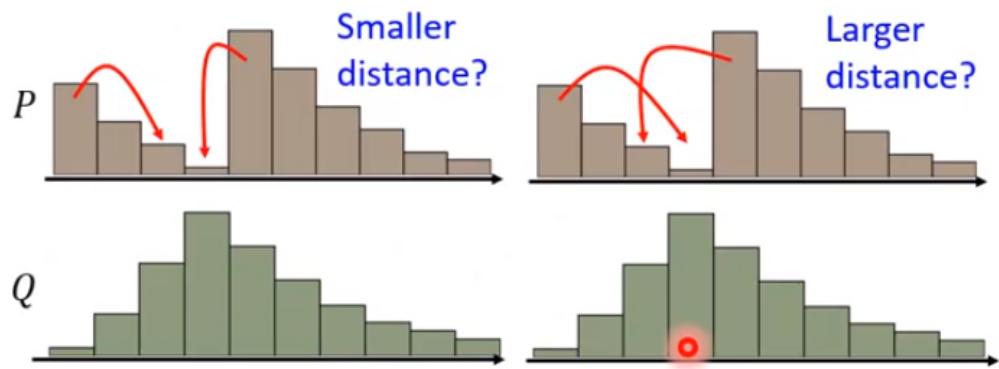


图 44: Wasserstein distance 的度量方法

穷举所有可能的铲土的方法，每一种方法我们叫做 moving plan，看哪个方法的距离最小，就是 earth mover's distance。

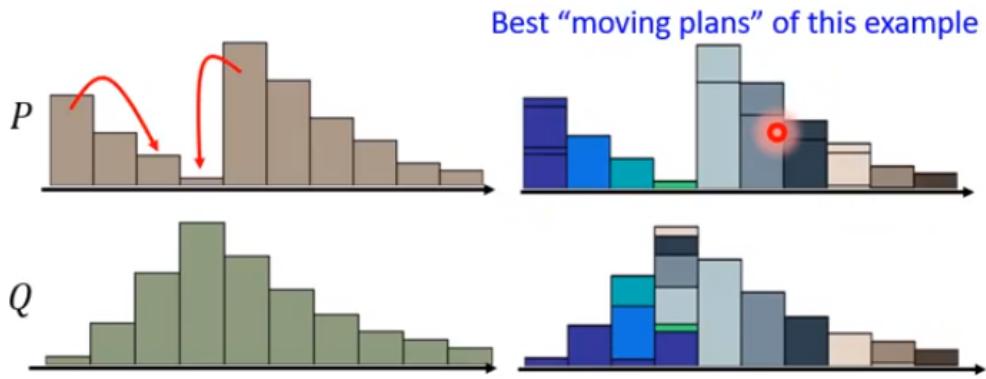


图 45: Wasserstein distance 最佳移动方法

想要把 P 搬到 Q ; 我们首先要定义一个 moving plans, 实际是一个矩阵, 每一个元素指的是从纵坐标的挪多少土到横坐标; 越亮表示挪的图越多。如下图所示,

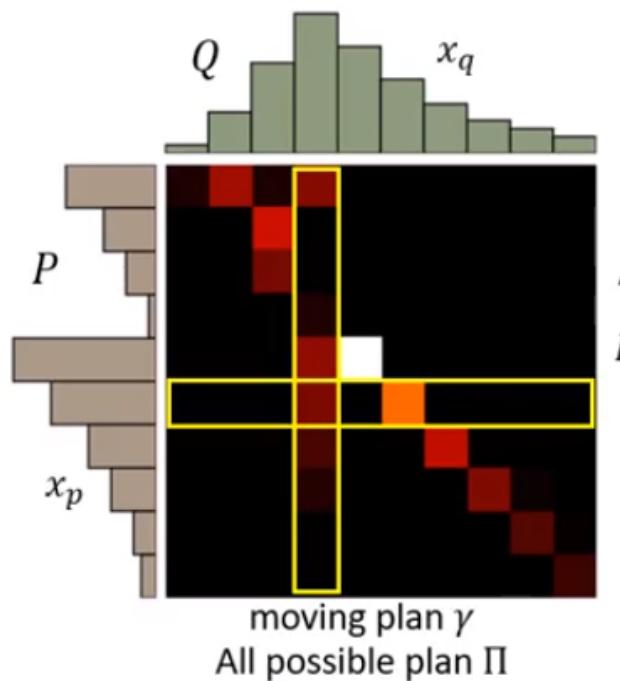


图 46: moving plan matrix

假设 moving plan 用 γ 表示, 那么他们所走的距离为:

$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\| \quad (22)$$

所谓, Earth Mover's Distance 为:

$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma) \quad (23)$$

那么可以看到将 JS 散度改写为 Wasserstein 距离之后有很大的好处。当使用 JS 散度的时候, 不重叠

都是 log2；当使用 Wasserstein 距离的时候，当两个分布虽然不重叠但是当距离变小的时候，散度也小。如下图所示，

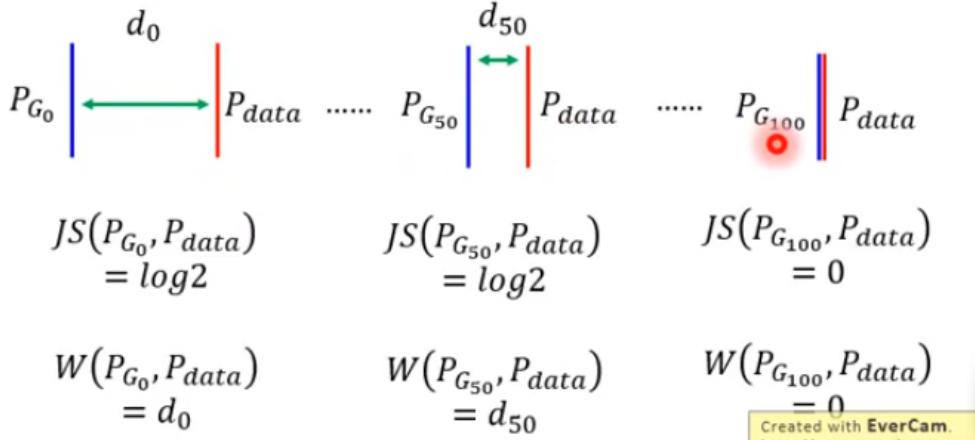


图 47: Wasserstein 距离衡量

那么，下一个问题为，如何修改判别器的公式，才能使判别器得到的结果是 wassertein distance 呢？这个推导并不简单，这里就简单的展示一下结果，

$$V(G, D) = \max_{D \in 1\text{-Lipschitz}} \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] \} \quad (24)$$

其中还要有一个约束，判别器必须是一个 1-Lipschitz 函数；它可以使函数变得非常平滑，如果没有这个约束，判别器一味地让：real 的分数就越来越高，generated 的分数越来越低；那么就崩溃了，永远不会收敛；因为值可以越来越大越来越小。因此必须有额外的限制，让判别器必须是平滑的。

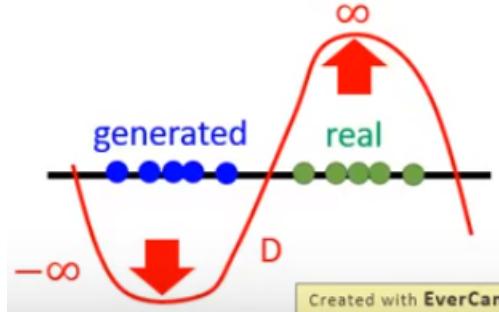


图 48: 为什么判别器要满足 1-Lipschitz 连续

而 K-Lipschitz 函数被定义为：

$$\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\| \quad (25)$$

那么下一个问题是，如何做到判别器满足 1-Lipschitz 连续呢？

7.3.1 Weight Clipping

这种方法没有严格的实现 1-Lipschitz，但是限制的函数的平滑性，非常的简单粗暴。其基本思路是，仍然使用梯度上升来 $\max V(G, D)$ ，但是当你发现某个参数 w 大于设定好的 C 就设定为 C ，小

于设定好的 $-C$ 就设定为 $-C$ 。这样就可以将网络的参数设定在 $[-C, C]$ 之间。

7.3.2 Improved WGAN

如果一个函数是 1-Lipschitz 等价于对于任意的 x 的梯度的范数总是小于等于 1, $\forall x, \|\nabla_x D(x)\| \leq 1$; 那么这个问题如何求解呢? 可以在后面加一个正则化项,

$$V(G, D) \approx \max_D \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] - \lambda \int_x \max(0, \|\nabla_x D(x) - 1\|) dx \} \quad (26)$$

但是实际上对于所有的 x 都求约束是几乎不可能的, 那么就退而求其次, 假设 X 先定好的分布中 $P_{penalty}$ 中 sample 出来的, 只管这部分的 x , 其他的就不管了。于是公式 (26) 将被改写为,

$$V(G, D) \approx \max_D \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] - \lambda \mathbb{E}_{x \sim P_{penalty}} [\max(0, \|\nabla_x D(x) - 1\|)] \} \quad (27)$$

那么 $P_{penalty}$ 究竟长什么样呢? 从 P_{data} 和 P_g 选取两个点, 连成直线, 从直线间 random sample 当作 $P_{penalty}$ 中的 x ; 为什么呢? 因为我们是想让 P_G 往 P_{data} 移动, 只有中间的区域才会影响结果。

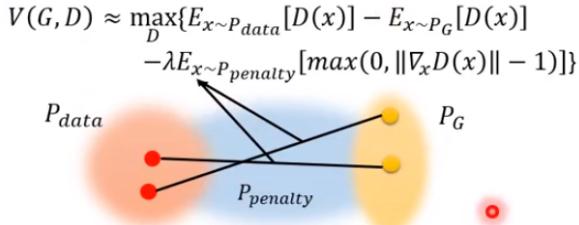


图 49: $P_{penalty}$ 具体化

在 WGAN 的文章中, 实际上是想让 $\|\nabla_x D(x)\|$ 越接近 1 越好, 最终的求解 Wasserstein 距离的目标函数为:

$$V(G, D) \approx \max_D \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] - \lambda \mathbb{E}_{x \sim P_{penalty}} [(\|\nabla_x D(x) - 1\|)^2] \} \quad (28)$$

实际上 WGAN 也不是那么的完美, 它也存在着自己的问题, 比如下图所示,



图 50: WGAN-GP 不 make sense 的地方

比如黄色的点是 P_G 的点, 红色的线是 P_{data} , 那么如果说黄色的点要靠近红色的点, 也应该是走最近的地方去靠近, 而不是穿过红色的点那么多的区域去靠近, 总感觉有点奇怪。还有 Spectrum Norm, 大家可以自己仔细看看, 可以真的让每一个地方的梯度都小于 1。

7.3.3 GAN 改成 WGAN

此处将展示的如何将 GAN 改写成 WGAN。实际上也挺简单的，前面都说到过了，

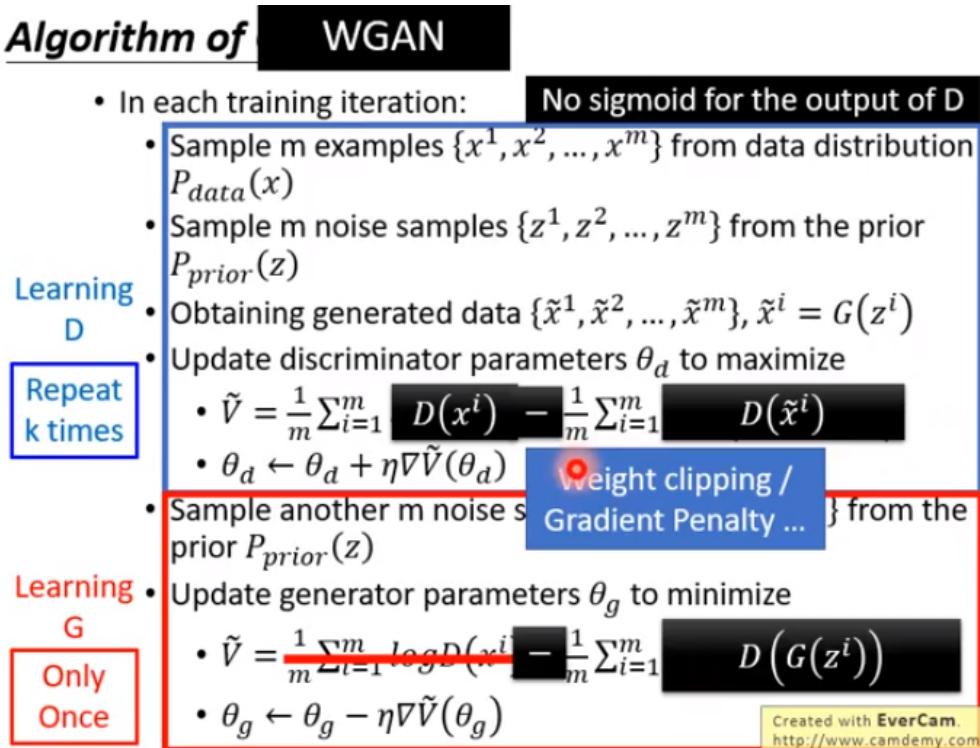


图 51: GAN to WGAN 的改进

7.4 Energy-based GAN (EBGAN)

Energy-based GAN 中是对 Discriminator 做了一些改变，将其改变为了 autoencoder，用的是重构误差乘上一个数来表示当前图片的质量。EBGAN 认为，重构质量难度越低的图片质量越高，反之亦然。EBGAN 的优点在于对于一般的判别器，需要让生成器慢慢变强；但是 autoencoder 的判别器可以 pre-train，可以用正样本进行预训练；一开始的判别器会很强，这样生成器一开始就会产生很好的 image；

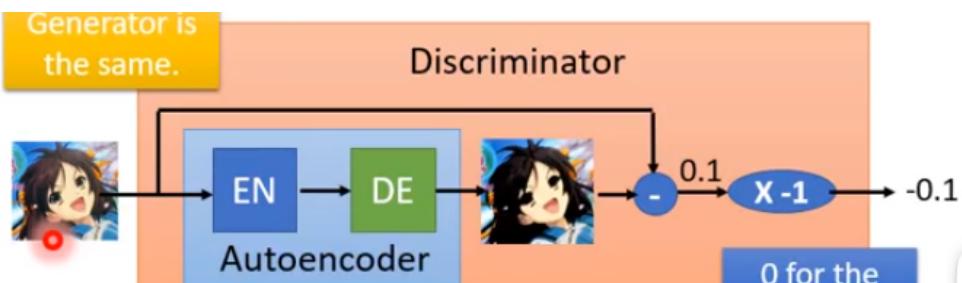


图 52: EBWGAN 的基本框架

8 Feature Extraction

本小节主要讲的是有关 GAN 做 Feature Extraction 的一些方法，包括 Info-GAN，VAE-GAN 和 BiGAN，还有如何用 Cycle 来做 Unsupervised learning 的一些方法。

8.1 Info-GAN

GAN 的 Generator 是输入一个向量，输出一张图片。那么，我们希望可以找到输入向量维度和输出特征之间的关系，比如我想改变输出图片的粗细，倾斜角度等等。实际上这个挺难做到的，往往不知道改变了输入向量维度的值会导致怎样的输出结果的改变。比如如下图片，期望改变 input 的 vector 可以对应到输出空间的改变，但是实际上这个对应关系可能非常的复杂，就会导致不知道自己在改什么东西。

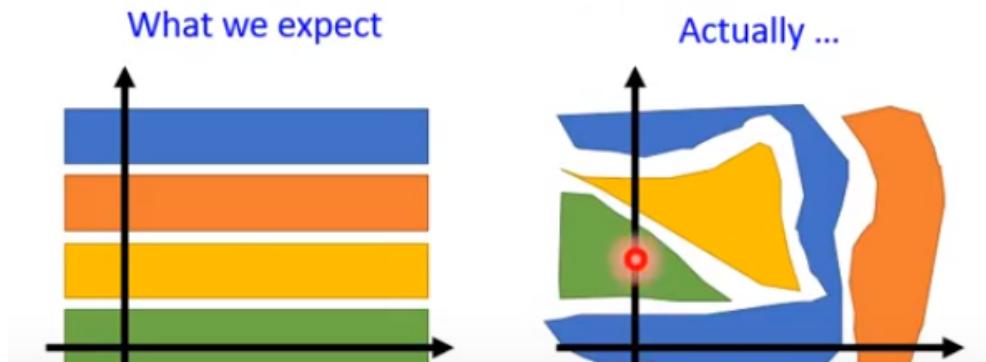


图 53: input sapce 和 output space 之间不是简单的 linear 关系

InfoGAN 的结构示意图如下所示，

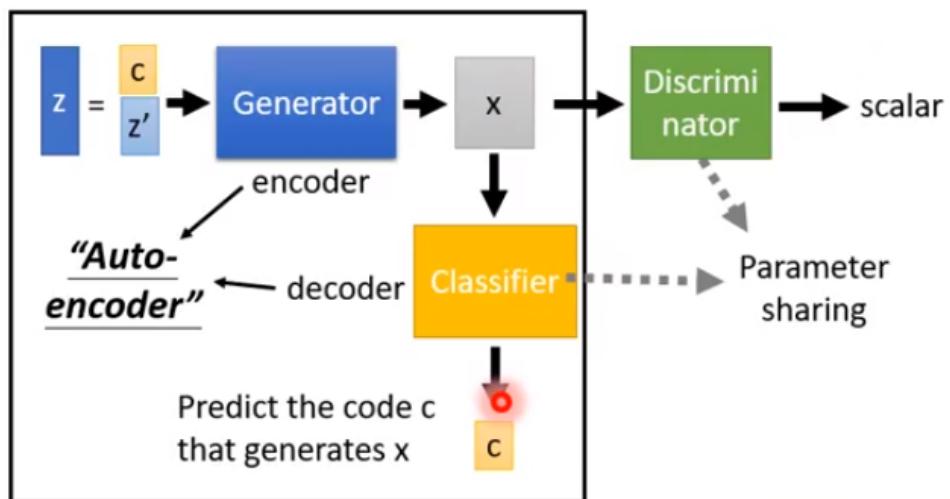


图 54: InfoGAN 的结构示意图

其主要思维是这样的，首先把 z 分成 c, z' 两个部分，其中我们希望通过训练使得 c 成为我们可

以控制的特征。因为原问题是输入对输出的影响不明确，现在我们希望 c 部分可以明确的影响输出 x ，而 z' 是完全随机的，我们无法解释的部分。如果学习到了 c 可以清晰的影响输出 x ，那么它应该是很容易被重构出来的。所以，可以将 Classifier 看成是 Decoder，实际上黑框里面的东西就是一个 Auto-Encoder。但是，必须对 x 加上一个 Discriminator，因为假如没有这个判别器来判断输出图片的质量的话，那么输出的 x 很容易欺骗 decoder，比如只要把 c 贴在 x 上就好了，肯定很容易被重构回来。输出效果还是比较好的，如下所示，

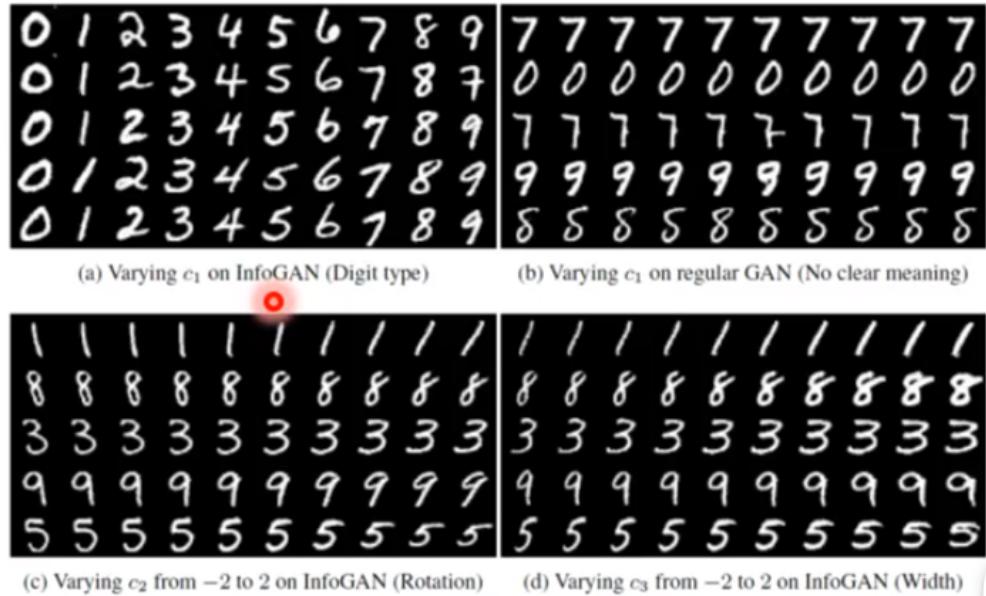


图 55: InfoGAN 的实验结果

8.2 VAE-GAN

VAE-GAN 是相互强化，相互提升。

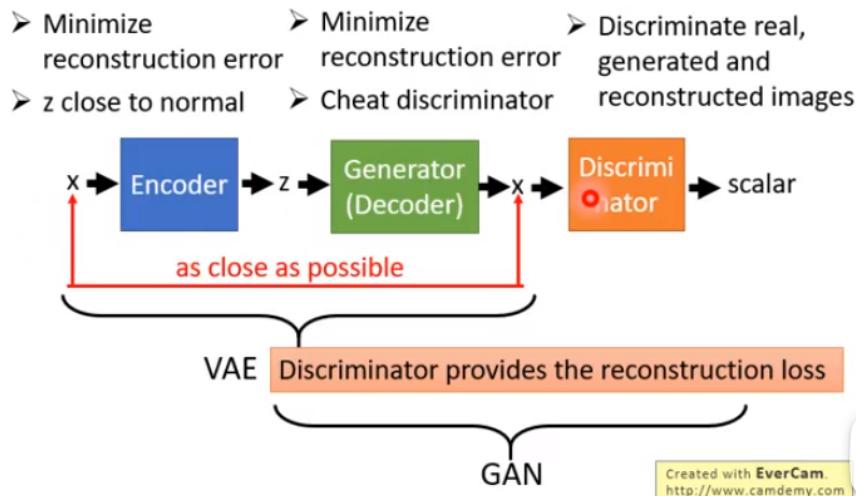


图 56: VAE-GAN 结构示意图

从 VAE 的角度说，在 x 末端加上一个 Discriminator 让图片更加真实，因为原始 vae 只是令重构 loss 更小，并不能有效的学习到全局的信息，而 Discriminator 可以帮助其解决这个问题。从 GAN 的角度说，GAN 的一个缺陷是 Generator 没有见过真实的图片长什么样，全靠自己的摸索，这个过程可能会很漫长。而 VAE 可以帮助其解决这个问题，因为可以见过原来的图片长什么样，采样得到的图片可以尽可能贴近见过的图片。

8.3 BiGAN

BiGAN 示意图如下所示，其算法过程如下所示，也非常容易看懂。和 VAE 不同的是，这里的 Encoder 输出的 z 和 Decoder 输入的 z 并不是同一个 z 。把两组 $\{(x, \tilde{z}), (\tilde{x}, z)\}$ 输入 Discriminator，希望 Discriminator 最终无法辨别出来。这实际就是 VAE 最原始的目的，Encoder 可以等价为 $P(x, z)$ ，Decoder 可以看成 $Q(x, z)$ ，而 Discriminator 希望 $P(x, z)$ 和 $Q(x, z)$ 无法被分辨出来。

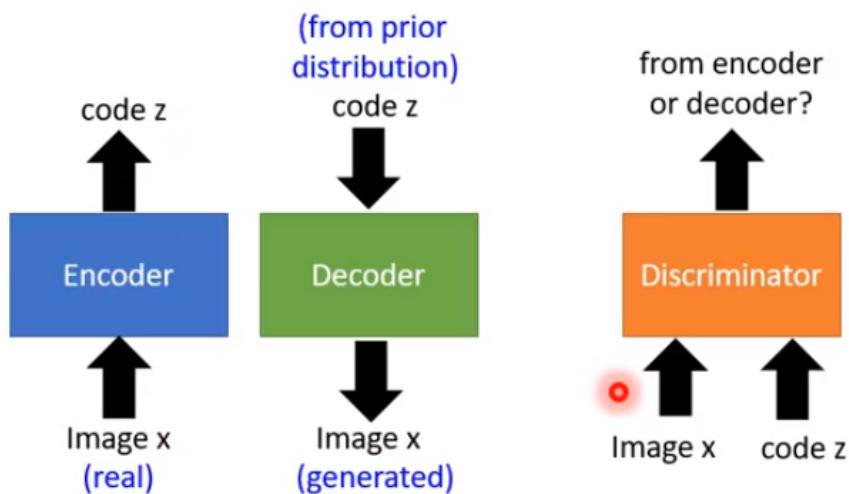


图 57: BiGAN 结构示意图

实际上，大家可能在这里提出两个问题，BiGAN 的最终目的好像和多加了一层的 Auto-Encoder 一样。

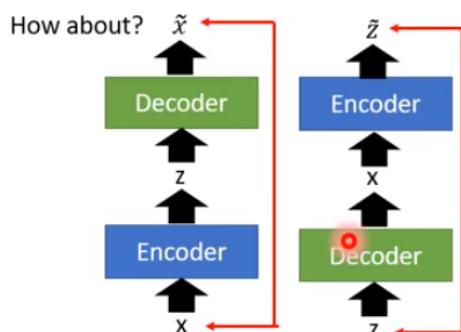


图 58: BiGAN 结构上和两层 Auto-Encoder 非常类似

实际上，当 BiGAN 和双层 Auto-encoder 都达到最优时，两者是可以等价的。但是现实情况是，不可能做到训练达到最优。而 BiGAN 的下限更高，泛化能力越强，至于为什么，李老师没有说的很清楚，看来要仔细的阅读原文了。

8.4 Triple GAN

既然有 BiGAN 就当然有 Triple GAN。这是一个半监督学习的方法，李老师在这里没有细说。

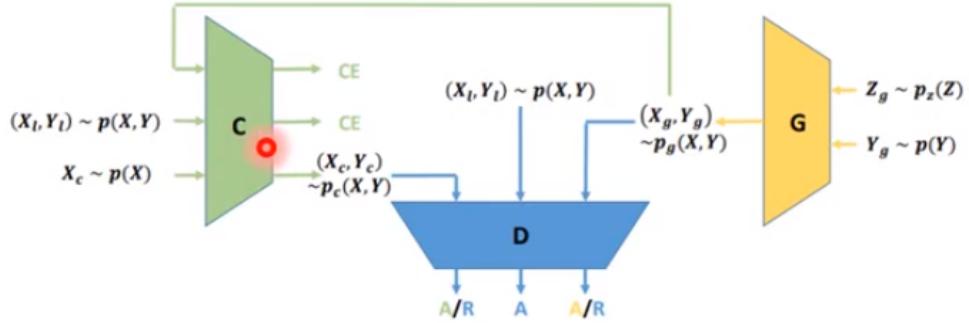


图 59: Triple GAN 结构示意图

李老师其中讲了一个例子，GAN 可以用来看做特征的分离，比如一段语音，可以用 GAN 来提取其中人物的特征和内容的特征，人物特征来说明说话的是谁，内容的特征来表示说话的人是谁。这样可以做到声纹的匹配，类似的应用还可以用到各种领域。GAN 看起来没什么用，确实也没什么用，但是 Conditional GAN 用处就很大了。

9 Improving Sequence Generation by GAN

sequence generator 的应用很多，比如机器翻译，人机对话，甚至强化学习也算是 sequence generator。使用 GAN 进行 sequence generator 主要有两大部分，分别是 Conditional Sequence Generation 和 Unsupervised Conditional Sequence Generation。

9.1 Conditional Sequence Generation

Conditional Sequence Generation 和传统的监督学习很像，往往输入是一个 sequence 或者 scale，输出是一个 sequence。sequence generator 和传统的 GAN 并不相同，原因在于 sequence generator 本身就要求了输出是满足某些特性的，例如时间的线性特征，马尔可夫性，无后效性等。传统的 sequence generator 是基于类似 RNN, LSTM, Transformer 等结构实现的，往往同时是一个 encoder-decoder 模型，它将输入作为时间序列编码，再用另一个 decoder 解码。传统的 sequence generator 有 ASR, Translation, Chatbot 等。但是传统的模型做分类或者回归还行，一到生成复杂的数据结构就不行了。其实原因和最开始为什么要用 GAN 一样，是因为传统的模型都是在用各种既定的损失函数，而只有专门训练的 Discriminator 可以准确的评价 generator 的输出。就像是 VAE 本身的问题一样，要的目的是像素对像素越靠近越好，但是像素对像素越靠近越好，得到的图片不一定就更符合人类的判断。比如下图给出的 Chatbot 的例子，将说明传统的监督学习方式为什么不行。

Review: Sequence-to-sequence

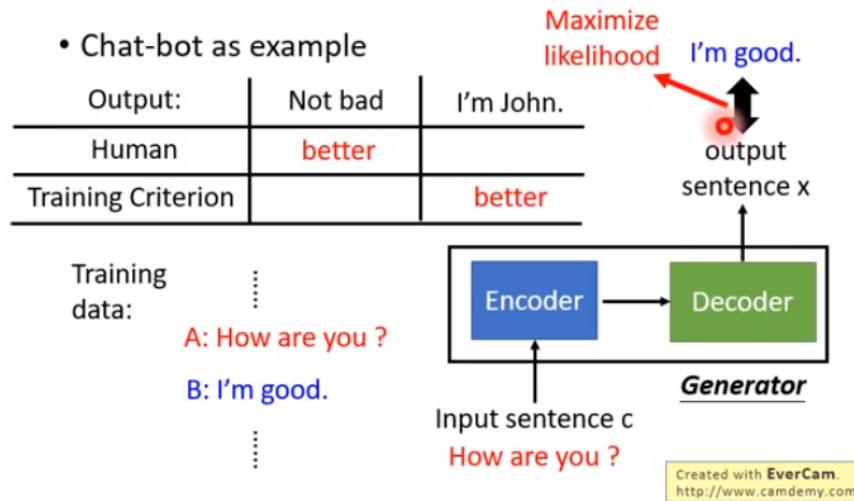


图 60: Chatbot 例子

输入是 How are you, 然后对应的输出是 I'm good. (这就相当于是 label 嘛, 毕竟是监督学习) 那么如果有两句话, 分别是 Not bad 和 I'm John, 哪一个更好呢? 显然是 Not bad 更好, 但是如果你用极大似然计算的话, I'm John 离 I'm good 更近啊。同时, 即便你的语料库里这两种答案都有, 那么传统的模型也会倾向于生成两者的平均, 那就不知到是什么东西了。

9.1.1 RL improve Sequence-to-Sequence

其实感觉用 RL 来解这个 Sequence-to-Sequence 问题有些奇怪。其求解的方法如下所示,

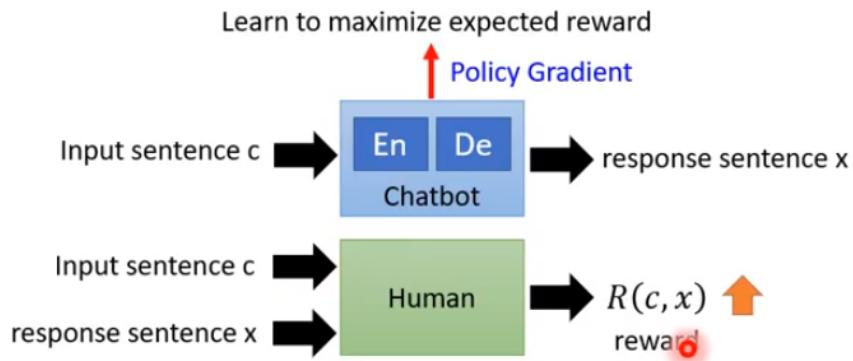


图 61: RL improve Sequence-to-Sequence 基本思路

就是让机器去网上随意和他人进行谈话, 然后让人类对机器的回答打分。实际用 human 来取代 discriminator 的作用。而更新 Chatbot 的方式使用的是 Policy Gradient。其计算方法如下所示,

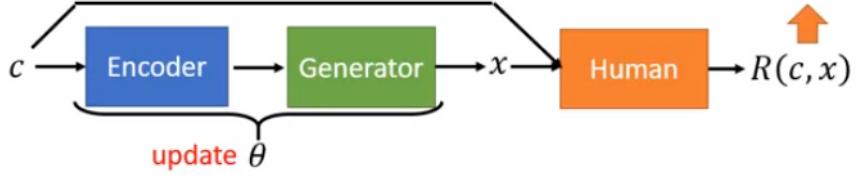


图 62: Policy Gradient 基本思路

我们的目标是 $\theta^* = \arg \max_{\theta} \bar{R}_{\theta}$

$$\bar{R}_{\theta} = \sum_c P(c) \sum_x R(c, x) P_{\theta}(x | c) = E_{c \sim P(c)} [E_{x \sim P_{\theta}(x|c)} [R(c, x)]] \quad (29)$$

其中， c 是输入的句子， x 是输出的句子。

$$\bar{R}_{\theta} = E_{c \sim P(h), x \sim P_{\theta}(x|c)} [R(c, x)] \quad (30)$$

根据， $c \sim P(h), x \sim P_{\theta}(x | c)$ 采样得到 $\{(c^i, x^i)\}_{i=1}^N$ ，就可以直接计算累计回报，

$$\bar{R}_{\theta} \approx \frac{1}{N} \sum_{i=1}^N R(c^i, x^i) \quad (31)$$

但是通过这样的推导，好像不知不觉 θ 就不见了，实际上 θ 是和采样有关的， θ 会影响采样的结果。

$$\begin{aligned} \nabla \bar{R}_{\theta} &= \sum_c P(c) \sum_x R(c, x) \nabla P_{\theta}(x | c) \\ &= \sum_c P(c) \sum_x R(c, x) P_{\theta}(x | c) \frac{\nabla P_{\theta}(x | c)}{P_{\theta}(x | c)} \\ &= \sum_c P(c) \sum_x R(c, x) P_{\theta}(x | c) \nabla \log P_{\theta}(x | c) \\ &= E_{c \sim P(c), x \sim P_{\theta}(x|c)} [R(c, x) \nabla \log P_{\theta}(x | c)] \\ &\approx \frac{1}{N} \sum_{i=1}^N R(c^i, x^i) \nabla \log P_{\theta}(x^i | c^i) \end{aligned} \quad (32)$$

于是，可以使用随机梯度下降法进行优化了，

$$\begin{aligned} \theta^{\text{new}} &\leftarrow \theta^{\text{old}} + \eta \nabla \bar{R}_{\theta^{\text{old}}} \\ \nabla \bar{R}_{\theta} &\approx \frac{1}{N} \sum_{i=1}^N R(c^i, x^i) \nabla \log P_{\theta}(x^i | c^i) \end{aligned} \quad (33)$$

但是，每一次更新参数都要重新收集 $R(c^i, x^i)$ ，实际上这非常的麻烦。怎么解决 reward 非常的难，所以来引入了 GAN，不再用人来判断一段对话的好坏，而是用机器 discriminator 来判断。

Conditional GAN

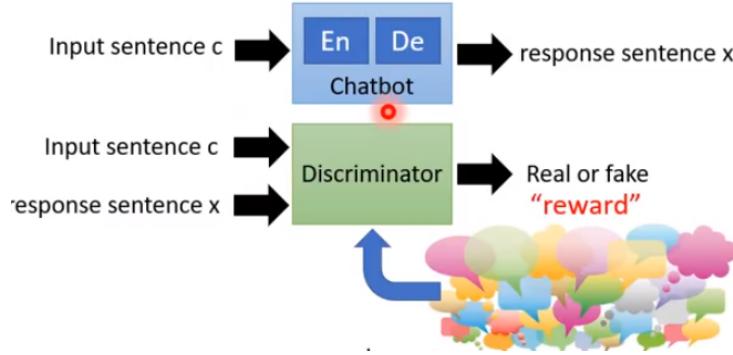


图 63: conditional GAN 实现对话评估的基本思路

实际上这和传统的 conditional GAN 是一样的。一切看着似乎都很完美，但是在实际操作过程中会有问题。问题在于 Discriminator 的结构上。既然 Discriminator 的输入是一个句子，输出也是一个 scale，那么 Discriminator 和 RNN 应该会非常类似。而 RNN 的每一个节点的输出往往都是从隐变量生成的分布的一个采样：

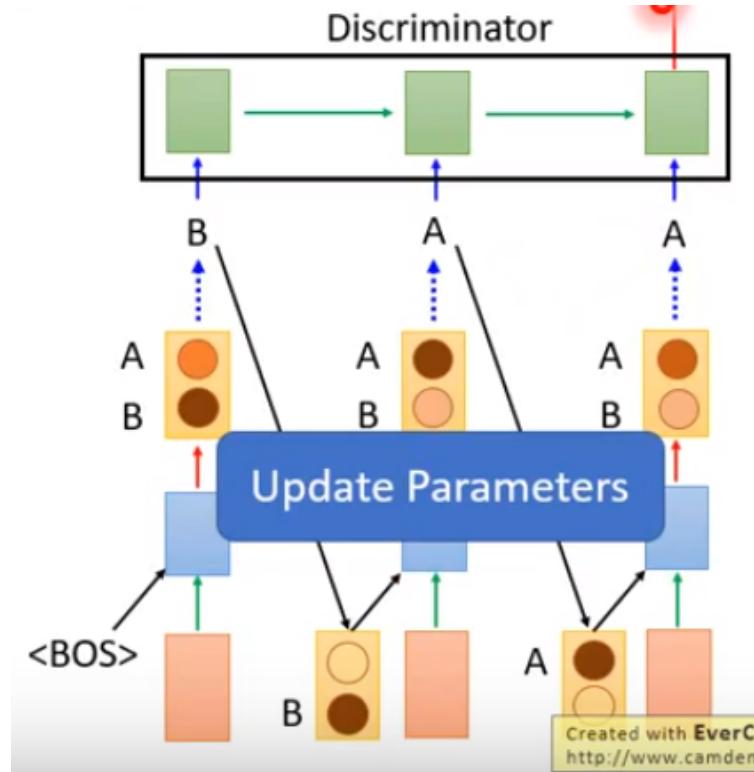


图 64: Discriminator 的基本网络架构

但是，因为 sampling process 的操作，让 Generator 没办法计算梯度。实际上我们之间训练 Generator 都是把 Discriminator 接在 Generator 前面，然后固定住 Discriminator 的参数调整 Generator。如果生成的是图片那还比较简单，但时当生成的是 sequence，每个节点的输出都是从分布中采样得到

的。如果这一步放到神经网络的最后一步还好，但是放到了中间，这样就没办法计算前面的梯度。而前几层正是我们要训练的几层。而主流的解决方法有两种，Continuous Input for Discriminator 和 Reinforcement Learning。

9.1.2 Continuous Input for Discriminator

Continuous Input for Discriminator 的思路非常的简单，就是 Discriminator 根本就不 sample 了，直接把分布输入 Discriminator 训练。模型如下所示，

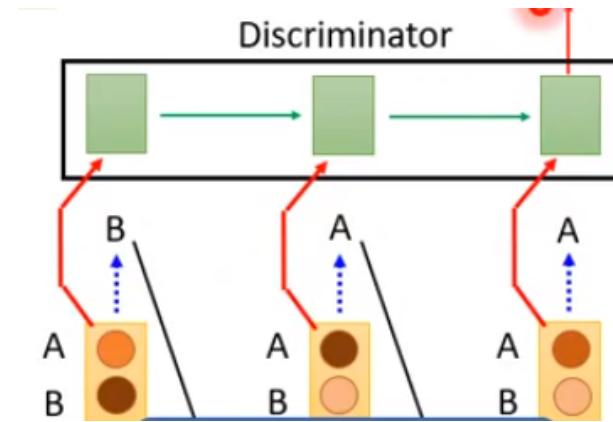


图 65: Continuous Input for Discriminator 的基本网络架构

但是问题是真实的样本往往都是 one-hot 的，而分布往往都不是 one-hot 的形式。这样的训练结果可能会很奇怪，Generator 为了欺骗 Discriminator 只要尽量的输出 one-hot 就可以了，而不会去管真实的语意。传统的 GAN 不行，但是 WGAN 往往可以解决这个问题。原因我们在介绍 WGAN 的时候就提到过了。

实际上这一部分是 NLP 的内容，对我的研究也没什么太大的用，这里就看个热闹了，不深入研究。

10 Evaluation

本小节讲的主要是如何评价一个 GAN 训练的好还是不好。用 GAN 产生了 image，怎么知道这个 image 好还是不好呢？其实最好的方式还是人来看，但是人来看并不能很客观的评价出一个模型的好坏。所以本小节想要用一种客观的评价指标来评价模型的好坏。

首先，采用的是极大似然估计（MLE）的方法，从真实的数据集中采样出一堆的样本点 $\{x_i\}$ ，计算生成模型 P_G 的极大似然。直观感觉就是 Generator 产生 real data 的概率越高当然也就越好。

$$L = \frac{1}{N} \sum_i \log P_G(x^i) \quad (34)$$

但是，我们没有办法计算 $P_G(x^i)$ ，只能从 P_G 中采样。 P_G 通常使用神经网络来生成的。

10.1 Kernel Density Estimation

那么，这怎么办呢？有一个方法叫做 Kernel Density Estimation，是让 GAN 生成大量样本，然后用 GMM 去拟合 P_G ，然后计算这个 GMM 和测试集对应的分布的 MLE。

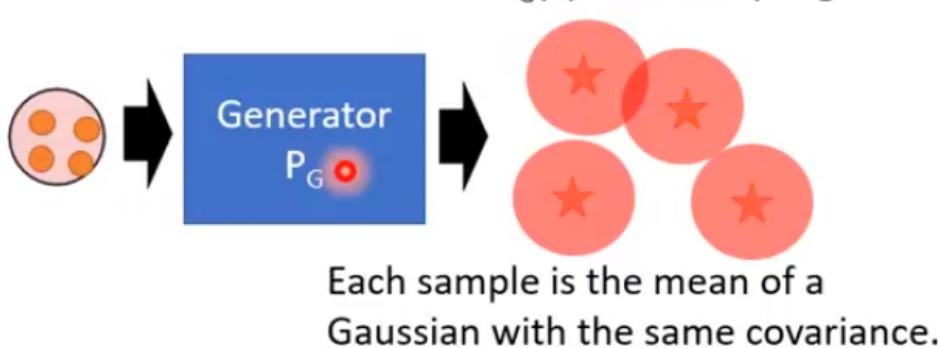
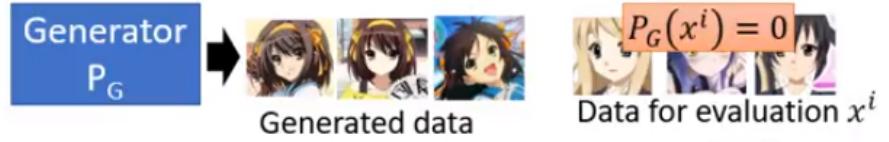


图 66: Kernel Density Estimation

那么，问题是用 GMM 拟合 GAN 和用 GMM 拟合原始图片分布的问题是一样的，那就是 GMM 的能力往往不够。你不知道 GMM 的模型需要多少个 Component 才合适，你也不知道 GAN 需要生成多少样本才能让 GMM 训练的比较好。

10.2 Likelihood v.s. Quality

- Low likelihood, high quality?
Considering a model generating good images (small variance)



- High likelihood, low quality?

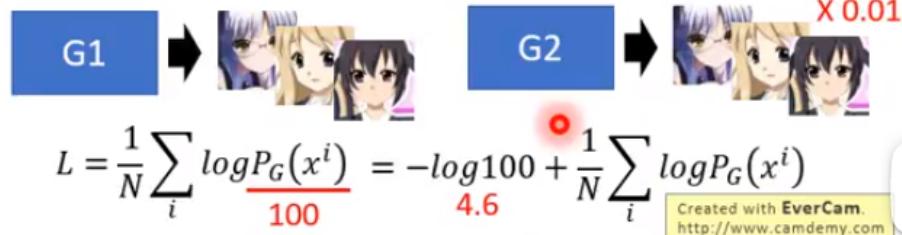


图 67: Likelihood v.s. Quality 并不一定是对应的关系

此图中描述的第一种情况是，假设 P_G 生成的图片质量非常的高，但是是测试数据集中没有的，这时就会认为 $P_G(x^i) = 0$ 。生成图片质量非常的差，这显然是不对的。

而第二种情况，假设有两个 Generator，显然第一个非常好，而第二个比较差，但是在计算的时候，数值差距并不大。

10.3 Objective Evaluation

其实也没有什么理论上更好的方法，不过有一些实践上看起来不错的方法。首先是 Objective Evaluation，这个方法就是说，找一个现成的分类器，看这个分类器能不能很好的处理你的 GAN 生成的图片。这常常包括两方面，一方面它希望 GAN 生成的每一张图片在分类的时候的概率分布最好是分得比较开的，这就表示生成图片的质量比较高。

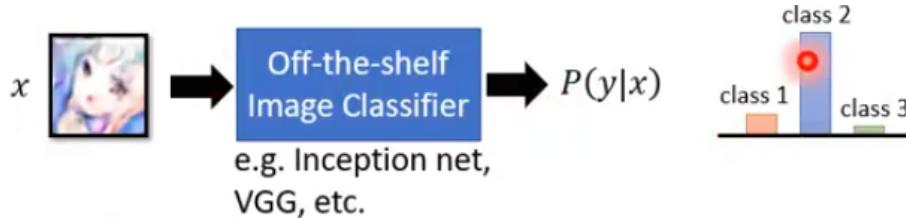


图 68: GAN 生成的每一张图片在分类的时候的概率分布最好是分得比较开

同时也也要考虑生成的图片的多样性。它希望你的 GAN 不只生成同一种类别的图片，换句话说，它希望多个图片的分类得到的分布相加后是 uniform 的：

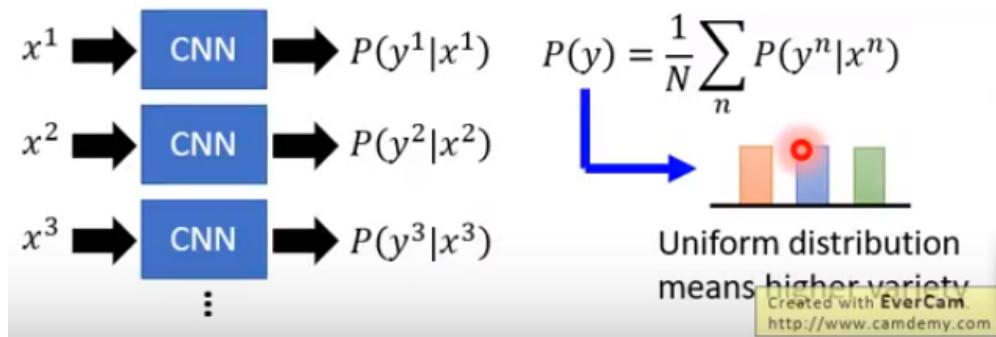


图 69: GAN 生成图片足够多样

所以，就可以定出 Inception Score 了，

$$\text{Inception Score} = \sum_x \sum_y P(y|x) \log P(y|x) - \sum_y P(y) \log P(y) \quad (35)$$

此外，我们不希望 GAN 仅仅是机械的记住了训练集中的数据，而是具有创造性。简单的方案是用 GAN 生成的图片去在原训练集中去做 KNN，但是要注意我们不能用 L1 或者 L2 范数去衡量两个图片的相似性，因为他们不具有平移不变性和旋转不变性。

10.4 Model Dropping

最后就是 mode dropping 的问题，也就是生成的图片的多样性不够。这个问题和 Mode collapse 很像。暂时没有理论上很好的方法，实际上你可以多训练几个 GAN 然后 ensemble 一下。