8 算例 E:《放弃显式推断，走向对抗博弈——生成对抗网络 (GAN)》

在上一章的 VAE 中，我们看到了“摊销推断”的极致：通过一个全局编码器 ϕ 去近似复杂的真实后验 $P(Z|X)$，并通过最大化证据下界 (ELBO) 来显式地对齐数据分布。然而，如果真实的数据分布 $P_{data}(X)$ 过于复杂，导致哪怕是高斯似然（MSE 损失）都无法捕捉其尖锐的细节呢？

本章将引入一种完全不同的范式——生成对抗网络 (GAN)。它彻底放弃了“从观测推断隐变量”的执念 [14]，转而通过两个神经网络的博弈，隐式地实现全局分布的强力对齐。

8.1 生成对抗网络 (GAN) 的算例定义

为了与 VAE 形成完美的镜像对比，我们将 GAN 的核心组件进行重新梳理。在 GAN 中，我们不再拥有推断网络（Encoder），取而代之的是一个负责“挑刺”的鉴赏家（Discriminator）。

8.1.1 变量定义与逻辑对齐 (对比 VAE)

符号	描述	在 VAE 中的逻辑对应
X_{real}	真实观测变量（来自 P_{data}）	对应 VAE 的输入 X
Z	低维隐空间噪声（Latent Noise）	对应 VAE 的隐变量 Z
θ	生成器参数 (Generator, G_θ)	对应 VAE 的 Decoder 参数 θ
ϕ	判别器参数 (Discriminator, D_ϕ)	**取代了** VAE 的 Encoder 参数 ϕ

8.1.2 概率模型设定：只有生成，没有推断

在 GAN 的框架下，概率结构被极度简化：

1. **隐变量先验 (Prior)**：与 VAE 完全一致，假设隐空间服从简单的标准先验：

$$P_z(Z) = \mathcal{N}(Z|0, \mathbf{I}) \quad 或 \quad \text{Uniform}(-1, 1)$$

2. **观测似然 (Likelihood) ——GAN 无法进行似然估计的梯度计算**：

回顾 VAE:在 VAE 中,给定隐变量 Z,观测值 X 的生成是一个包含噪声的**采样过程**:$X \sim \mathcal{N}(f_\theta(Z), \sigma^2\mathbf{I})$。在 VAE 中，我们假设 $P_\theta(X|Z)$ 是高斯分布，从而导出了 MSE 损失（见 eq. 6-3）。或者假设其为 0,1 二值向量，则导出 BCE 损失（见 eq. 6-2）。

GAN 的变化：但在 GAN 中，生成器 $G_\theta(Z)$ 被视为一个确定性的狄拉克分布 (Dirac Delta) 映射。我们不再显式计算似然 $\log P(X|Z)$，而是直接输出伪造数据：

$$X_{fake} = G_\theta(Z)$$

在这里，生成器 $G_\theta(Z)$ 是一个确定性神经网络，这意味着一旦随机噪声 Z 给定，输出的假样本 X_{fake} 就被唯一确定，没有任何波动的余地。这被形象地称为从高斯似然到狄拉克分布 (Dirac Delta) 的**坍缩**。

[14]（何谓“放弃推断”？——即不再关心对 $P(Z|X)$ ，因此也没有变分函数 $q(Z)$, 也没有 PGM 中的推断图部分，也不会有 ELBO。还有哪些变分推断中常见的要素不再纳入考虑？请同学们思考。）

何谓“坍缩”?

狄拉克分布 $\delta(x)$ 是一个广义函数，可以将它直观地理解为方差趋近于 0 的高斯分布的极限：

$$\delta(x-\mu)=\lim_{\sigma\to 0}\mathcal{N}(x;\mu,\sigma^2\mathbf{I})$$

它具有极其极端的性质：在 $x=\mu$ 处概率密度无穷大，在其他任何地方概率密度为 0，且全空间的积分为 1。昵称：Spike-and-Slab Prior。 如果我们强行用概率密度函数 $P_\theta(X|Z)$ 来描述这种“确定性映射”，那么这个条件分布就坍缩成了一个质量全集中在 $G_\theta(Z)$ 这一点的狄拉克分布：

$$P_\theta(X|Z)=\delta(X-G_\theta(Z)).$$

坍塌“扼杀”了显式似然计算的可能...

如果我们像 VAE 那样，试图将这个分布代入最大似然估计（MLE），去计算对数似然 $\log P_\theta(X|Z)$，会发生什么？

- 当生成的样本哪怕与真实样本有一丁点微小的像素偏差（即 $X\neq G_\theta(Z)$）时，其概率密度为 0，对数似然 $\log(0)\to-\infty$。
- 只有当生成样本与真实样本做到在所有维度上绝对一致（即 $X=G_\theta(Z)$）时，对数似然才为 $\log(\infty)\to\infty$。

这种处处不可导的极端概率使得我们彻底丧失了通过计算梯度来优化对数似然函数的可能性。这就是 GAN 必须引入“对抗博弈”的底层数学原因：既然显式的似然公式已经不可导、不可算，我们就只能被迫引入一个神经网络（判别器 D_ϕ），用它的梯度来充当引导生成器前进的“平滑替代信号”。

3. **判别器网络 (The Critic)**：由于我们无法计算显式似然，我们引入一个二分类网络

$$D_\phi(X)\in(0,1).$$

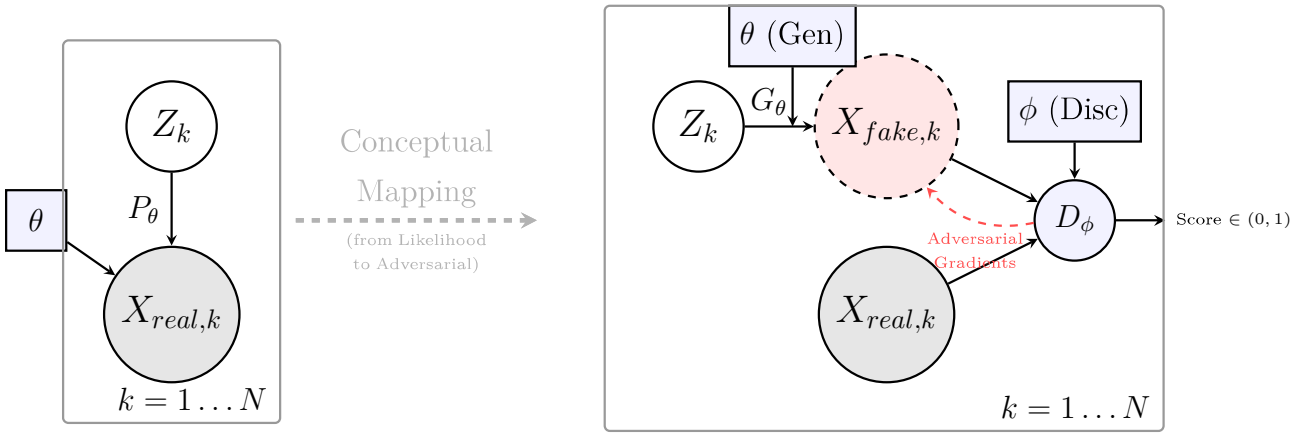
它的物理意义是：评估输入数据 X 来自真实数据分布 P_{data} 而不是生成分布 P_{fake} 的概率。

8.1.3 概率图模型与计算流图模型的对比

为了更深刻地理解 GAN 这种“隐式生成模型”与 VAE 这种“显式似然模型”的区别，我们将逻辑分解为两个层面：(a) 生成概率图模型 (Generative PGM)：描述我们假设的数据产生过程，即隐变量 Z 如何驱动生成观测数据 X。注意，GAN 的 PGM 仅包含生成路径，没有严谨的贝叶斯推断路径（$X\to Z$）。(b) 对抗计算流图 (Computational Graph)：描述 GAN 算法的实际具体实现。它不再通过计算似然函数来优化，而是通过生成器 G 与判别器 D 的博弈来隐式对齐分布。[15]

这种对比清晰地展示了 GAN 核心的范式转移：从“最大化显式似然”转向“最小化分布散度（通过对抗博弈实现）”。

[15]图右展示了一种广义的 PGM 图绘制思路。在标准的概率图模型（PGM）中，只有随机变量（如 Z 和 X）才用圆圈表示。但是，在计算图中，引入虚线圆圈 (X_{fake}) 代表一个中间变量或派生节点——它并不是独立产生的，而是由 Z 经过 G_θ 映射后的“瞬时状态”。计算图另外引入浅蓝色的圆圈 D_ϕ，代表的是动态的、正在运行的判别器实体——它接收 X 作为输入，进行复杂的非线性变换，最后输出 Score。也与模型参数方框 ϕ 做区别。



(a) 生成 PGM $P_\theta(X, Z)$ **(b) 对抗计算流图 $\min_G \max_D V(D, G)$**

图 8: **GAN 的概率图假设与对抗计算实现对比**。(a) 展示了理想化的生成概率图模型，描述 Z 如何通过 θ 生成数据分布。在 VAE 中，我们优化该似然的下界。(b) 展示了 GAN 的具体算法实现。由于确定性狄拉克映射导致似然不可算，我们彻底放弃了推断路径（$X \to Z$），引入神经网络判别器 D_ϕ，通过博弈（实线箭头表示数据流，红色虚线箭头表示对抗梯度反馈）来隐式地对齐分布。

8.2 GAN 的目标函数：Minimax 博弈

在 VAE 中，我们的目标是最大化对数边缘似然的下界 (ELBO)。但在 GAN 中，我们的目标函数是一个极小极大博弈 (Minimax Game) 的值函数 $V(D, G)$。

8.2.1 Score 函数 $V(D, G)$ 的定义

$$\min_G \max_D V(D, G) = \underbrace{\mathbb{E}_{X \sim P_{data}}[\log D_\phi(X)]}_{\text{真实数据的 Log-Likelihood 的期望}} + \underbrace{\mathbb{E}_{Z \sim P_z}[\log(1 - D_\phi(G_\theta(Z)))]}_{\text{Fake 数据被识别出的概率期望}} \tag{8-1}$$

1. 判别器 D_ϕ 的视角 (Max) 判别器的任务是最大化 $V(D, G)$。

- 对于真实数据 X_{real}，它希望输出概率 $D(X) \to 1$，使得 $\log D(X) \to 0$（最大值）。
- 对于生成数据 $X_{fake} = G(Z)$，它希望看穿伪造，输出概率 $D(G(Z)) \to 0$，使得 $\log(1 - 0) \to 0$（最大值）。

这本质上是在训练一个标准的二元逻辑回归分类器 (Binary Logistic Regression)。

2. 生成器 G_θ 的视角 (Min) 生成器的任务是最小化 $V(D, G)$。

- 它无法控制真实数据项，只能努力让判别器对自己的伪造品打高分。当生成器成功欺骗了判别器（得分很高）时，意味着 P_g 在判别器眼中已经和 P_{data} 重合了, 意味着模型具备了生成高质量样本的能力。
- 它希望 $D(G(Z)) \to 1$，使得 $\log(1 - 1) \to -\infty$（最小值）。

8.2.2 鞍点优化（Saddle Point Optimization）

虽然 SVM 诞生于统计学习时代，而 GAN 是生成模型的巅峰，但它们在数学结构上确实有着千丝万缕的联系。共同的数学形式表现为鞍点问题 (Saddle Point)。

在 SVM 的推导中，我们为了求解带约束的二次规划，引入了拉格朗日对偶性 (Lagrange Duality)。我们可以发现它的目标函数形式与 GAN 如出一辙。

SVM 的对偶问题：

$$\max_{\alpha} \min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)$$

这里我们通过优化权重 $\mathbf{w}$ 来最小化原始目标，同时通过拉格朗日乘子 α 来最大化约束惩罚。

GAN 的博弈问题：

$$\min_{G} \max_{D} V(D, G)$$

它们都在寻找一个鞍点——即在某一个维度上是极小值，而在另一个维度上是极大值。这两者在设计思路上都体现了某种“最坏情况下的稳健性”。

SVM 的逻辑：最佳分类面不是由所有点决定的，而是由那几个离边界最近的、最容易被分错的支持向量（Support Vectors）决定的。SVM 本质上是在说：“我要找到一个平面，使得它即使面对那些最难分的点，也能保持最大的间隔（Margin）。”

GAN 的逻辑：生成器 G 并不是在盲目学习，它是在针对那个最聪明的判别器 D。它在说：“我要学会一种生成方式，使得它即使面对那个最挑剔的鉴赏家，也能瞒天过海。”判别器 D 在 GAN 中的角色，其实非常像 SVM 中的那些支持向量——它们都代表了此时此刻最具有判别力的、最“硬”的信息，是它们在拉扯着模型边界的演进

另外，两者在强对偶性和弱对偶性上有一个重大的区别。

SVM（强对偶性）：因为 SVM 的原始问题是凸优化，满足 Slater 条件。所以对 SVM 来说：

$$\max_{\alpha} \min_{\mathbf{w},b} L = \min_{\mathbf{w},b} \max_{\alpha} L$$

这就是为什么我们在 SVM 中可以心安理得地先求导再最大化，或者反过来，结果是一样的。GAN（非凸困境）：神经网络是非凸的。对于 GAN，这种对偶性不再成立（只有弱对偶性 $\min\max \geq \max\min$）。因此，我们在 GAN 中必须严格遵守先 $\max_D$ 再 $\min_G$ 的博弈次序，否则模型就会因为找不到真正的鞍点而迅速崩溃。

8.2.3 第一个 Trick：交替优化策略 (Alternating Optimization)

在 VAE 中，ϕ (Encoder) 和 θ (Decoder) 是手牵手一起更新的（合作关系）。但在 GAN 中，两者是对抗关系，直接同时梯度下降会导致模型崩溃。因此我们必须交替训练：

1. **固定** G，将真实数据和生成数据喂给 D，用梯度上升法更新 ϕ，让 D 变聪明。
2. **固定** D，通过 D 的网络将梯度传回给 G，用梯度下降法更新 θ，让 G 骗过 D。

8.2.4 第二个 Trick：非饱和损失 (Non-Saturating Loss)

在训练初期，G 生成的图像极差，D 可以轻易以接近 100% 的置信度将其识别（$D(G(Z)) \approx 0$）。此时，生成器的损失函数 $\log(1 - D(G(Z))) \approx 0$，其梯度几乎消失，导致生成器无法学习。为了解决这个著名的**梯度消失问题**，在实际代码实现中，生成器的目标被替换为最大化 $\log D_\phi(G_\theta(Z))$。这改变了初期的梯度曲线，提供了更强烈的学习信号。

8.3 对抗网络实现的两个 Trick

GAN 核心交替优化 Trick 的 PyTorch 实现

```python
import torch
import torch.nn.functional as F

def train_gan_step(real_x, netG, netD, optG, optD):
    batch_size = real_x.size(0)

    # 定义真假标签 (Trick: 真实数据标为 1，生成数据标为 0)
    real_labels = torch.ones(batch_size, 1)
    fake_labels = torch.zeros(batch_size, 1)

    # ----------------------------------------
    # Step 1: 训练判别器 D (最大化 log(D(x)) + log(1 - D(G(z))))
    # ----------------------------------------
    optD.zero_grad()

    # 1.1 真数据馈入
    out_real = netD(real_x)
    lossD_real = F.binary_cross_entropy(out_real, real_labels)

    # 1.2 采样噪声 Z，生成假数据
    z = torch.randn(batch_size, latent_dim)
    fake_x = netG(z)

    # 1.3 假数据馈入 (注意使用 .detach() 防止梯度传回 G)
    out_fake = netD(fake_x.detach())
    lossD_fake = F.binary_cross_entropy(out_fake, fake_labels)

    # D 的总损失并反向传播
    lossD = lossD_real + lossD_fake
    lossD.backward()
    optD.step()

    # ----------------------------------------
    # Step 2: 训练生成器 G (Trick 2: 最大化 log(D(G(z))))
    # ----------------------------------------
    optG.zero_grad()

    # 此时让判别器重新评估生成数据
    out_fake_for_G = netD(fake_x)

    # 生成器的目标是让 D 认为这是真数据 (使用 real_labels)
    lossG = F.binary_cross_entropy(out_fake_for_G, real_labels)

    lossG.backward()
    optG.step()

    return lossD.item(), lossG.item()
```

8.4 总结：VAE 与 GAN 的差异

本讲义前几章中涉及的混合推断模型与 VAE 是基于配方法、变分下界等概率计算来进行隐变量建模；那么 GAN 则是通过引入巧妙的神经网络角色，绕开了繁琐的积分和解析推导。

1. **丢失了推断能力**：由于废弃了 $q_\phi(Z|X)$，标准 GAN 给定一张图像 X 后，无法逆向推导出它的隐变量 Z。这就阻碍了 GAN 在诸如“单细胞轨迹推断”等需要获得细胞表征的下游任务中的直接应用。

2. **获得了极致的保真度**：VAE 受限于重构项（通常等价于 MSE），倾向于生成模糊、平均化的样本。而 GAN 的判别器强制要求生成样本在所有高频细节上与真实分布对齐，从而能生成极其锐利、逼真的数据。

即便如此，GAN 仍能从数学角度予以解读。如果说 VAE 的 ELBO 是在优化 KL 散度 $\mathcal{D}_{KL}(q\|P)$，那么 GAN 的数学本质是在优化真实分布 P_{data} 与生成分布 P_{fake} 之间的 Jensen-Shannon 散度 (JSD)。这个散度只需要通过采样即可获得，人们转而会想办法对神经网络的采样计算进行设计，如同我们前一节课在 VAE 中的讨论中所涉及到的——这里的更多讨论，离变分推断的主线逐渐远离，我需要回到更多的变分模型，相关的细节补充就留给同学们课后思考吧。[16]

[16] 当判别器 D 训练到最优状态 $D^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_{fake}(x)}$ 时，代入价值函数可得：$V(G, D^*) = 2 \cdot JSD(P_{data}\|P_{fake}) - 2\log 2$。因此，生成器最小化 V，实际上就是在最小化两个分布之间的 JS 散度。与 VAE 中受限于高斯假设的 KL 散度相比，JS 散度不需要显式写出概率密度函数，仅需通过采样即可计算。