深度生成模型

深度生成模型就是利用深层神经网络可以近似任意函数的能力来建模一个复杂的分布$p_r(\boldsymbol{x})$。假设一个随机向量 Z 服从一个简单的分布 $p(\mathrm{z})$，$\mathbf{z} \in \mathrm{Z}$，我们使用一个深层神经网络 $\mathrm{g}: \mathrm{Z} \rightarrow \mathrm{X}$，并使得 $\mathrm{g}(\mathrm{z})$服从$p_r(\boldsymbol{x})$。

概率生成模型

密度估计

给定一组数据$\mathrm{D} = \{\mathrm{x}^{(i)}\}, 1 \leq \mathrm{i} \leq \mathrm{N}$，假设它们都是独立地从相同的概率密度函数为$p_r(\boldsymbol{x})$的未知分布中产生的。密度估计（Density Estimation）是根据数据集 D 来估计其概率密度函数$p_\theta(\boldsymbol{x})$。在机器学习中，密度估计是一种非常典型的无监督学习问题。如果要建模的分布包含隐变量，就需要利用 EM 算法来进行密度估计。

应用于监督学习

生成模型也可以应用于监督学习。监督学习的目标是建模输出标签的条件概率密度函数 $\mathrm{p}(\mathrm{y}|\mathrm{x})$。根据贝叶斯公式，

$$\mathrm{p}(\mathrm{y}|\mathrm{x}) = \frac{p(x,y)}{\sum_y p(x,y)}$$

将监督学习问题转换为联合概率密度函数 $\mathrm{p}(\mathrm{x}, \mathrm{y})$的密度估计问题。

判别模型 和生成模型相对应的另一类监督学习模型是判别模型（Discriminative Model）。判别式模型直接建模条件概率密度函数$\mathrm{p}(\mathrm{y}|\mathrm{x})$，并不建模其联合概率密度函数$\mathrm{p}(\mathrm{x},\mathrm{y})$。常见的判别模型有 logistic 回归、支持向量机、神经网络等。由生成模型可以得到判别模型，但由判别模型得不到生成模型。

生成样本

生成样本就是给定一个概率密度函数为$p_\theta(\boldsymbol{x})$的分布，生成一些服从这个分布的样本，也称为采样。

- 根据隐变量的先验分布$\mathrm{p}(\mathrm{z},\theta)$进行采样，得到样本 z；
- 根据条件分布$\mathrm{p}(\mathrm{x}|\mathrm{z},\theta)$进行采样，得到 x。

变分自编码器

含隐变量的生成模型

假设一个生成模型中包含隐变量，即有部分变量是不可观测的，其中观测变量 X 是一个高维空间 X 中的随机向量，隐变量 Z 是一个相对低维的空间 Z 中的随机向量。这个生成模型的联合概率密度函数可以分解为

$$p(x,z|\theta) = p(x|z,\theta)p(z|\theta)$$

这些分布的形式已知，只是参数 θ 未知，可以通过最大化似然来进行估计。给定一个样本 x，其对数边际似然$logp(x|\theta)$可以分解为

$$\log p(x|\theta) = \mathrm{ELBO}(\mathrm{q},\mathrm{x}|\theta,\emptyset) + D_{KL}(q(z|\emptyset)||p(z|x,\theta))$$

$$\mathrm{ELBO}(\mathrm{q},\mathrm{x}|\theta,\emptyset) = E_{z\sim q(z|\emptyset)}[log\frac{p(x,z|\theta)}{q(z|\emptyset)}]$$

- E-step：寻找一个密度函数$q(z|\emptyset)$使其等于或接近于后验密度函数$p(z|x,\theta)$；
- M-step：保持$q(z|\emptyset)$固定，寻找θ来最大化$\mathrm{ELBO}(\mathrm{q},\mathrm{x}|\theta,\emptyset)$。

在 EM 算法的每次迭代中，理论上最优的$q(z|\emptyset)$为隐变量的后验概率密度函数$p(z|x,\theta)$，

$$p(z|x,\theta) = \frac{p(x,z|\theta)p(z|\theta)}{\int_z p(x,z|\theta)p(z|\theta)dz}$$

变分自编码器（Variational Autoencoder，VAE）是一种深度生成模型，其思想是利用神经网络来分别建模两个复杂的条件概率密度函数。

- 用神经网络来产生变分分布$q(z|\emptyset)$，称为推断网络。理论上$q(z|\emptyset)$可以不依赖 x。但由于$q(z|\emptyset)$的目标是近似后验分布$p(z|x,\theta)$，其和 x 相关，因此变分密度函数一般写为$q(z|x,\emptyset)$。推断网络的输入为 x，输出为变分分布$q(z|x,\emptyset)$。
- 用神经网络来产生概率分布$p(x,z|\theta)$，称为生成网络。生成网络的输入为 z，输出为概率分布$p(z|x,\theta)$。

推断网络看作是“编码器”，将可观测变量映射为隐变量。生成网络可以看作是“解码器”，将隐变量映射为可观测变量。但变分自编码器背后的原理和自编码器完全不同。变分自编码器中的编码器和解码器的输出为分布（或分布的参数），而不是确定的编码。

推断网络

假设$q(z|x,\emptyset)$是服从对角化协方差的高斯分布，

$$q(z|x,\emptyset) = \mathrm{N}(\mathrm{z}|\mu_I,\sigma_I^2\mathrm{I})$$

$$\begin{bmatrix}\mu_I\\ \sigma_I\end{bmatrix} = f_I(x,\emptyset)$$

其中推断网络$f_I(x,\emptyset)$可以是一般的全连接网络或卷积网络。

推断网络的目标

推断网络的目标是使得$q(z|x,\emptyset)$来尽可能接近真实的后验$p(z|x,\theta)$，需要找到变分参数$\emptyset^*$来最小化两个分布的 KL 散度。

$$\emptyset^* = arg\min_{\emptyset} D_{KL}(q(z|\emptyset)||p(z|x,\theta))$$

一般使用的是变分推断方法，即用简单的分布 q 去近似复杂的分布$p(z|x,\theta)$。但是在深度生成模型中，$p(z|x,\theta)$是非常复杂的分布，很难用简单的分布去近似。因此，我们需要找到一种间接的计算方法。

$$\emptyset^* = arg\min_{\emptyset} D_{KL}(q(z|\emptyset)||p(z|x,\theta))$$

$$= arg\min_{\emptyset} \log p(x|\theta) - \mathrm{ELBO}(\mathrm{q},\mathrm{x}|\theta,\emptyset) = arg\max_{\emptyset} \mathrm{ELBO}(\mathrm{q},\mathrm{x}|\theta,\emptyset)$$

生成网络

- **先验分布$p(z|\theta)$**　一般假设隐变量 z 的先验分布为各向同性的标准高斯分布$\mathrm{N}(\mathrm{z}|0,\mathrm{I})$。隐变量 z 的每一维之间都是独立的。
- **条件概率分布$p(x|z,\theta)$**　建模条件分布$p(x|z,\theta)$通过生成网络来建模。为了简单起见，我们同样用参数化的分布族来表示条件概率分布$p(x|z,\theta)$，这些分布族的参数可以用生成网络来计算得到。

$$\theta^* = arg\max_{\emptyset} \mathrm{ELBO(q,x|\theta,\emptyset)}$$

模型汇总

推断网络和生成网络的目标都为最大化证据下界$\mathrm{ELBO(q,x|\theta,\emptyset)}$。因此，变分自编码器的总目标函数为

$$\max_{\emptyset,\theta} \mathrm{ELBO(q,x|\theta,\emptyset)} = \max_{\emptyset,\theta} \mathrm{E}_{z\sim q(z|\emptyset)}[log\frac{p(x|z,\theta)p(z|\theta)}{q(z|\emptyset)}]$$
$$= \max_{\emptyset,\theta} \mathrm{E}_{z\sim q(z|\emptyset)}[logp(x|z,\theta)] - D_{KL}(q(z|\emptyset)||p(z|x,\theta))$$

其中$\mathrm{E}_{z\sim q(z|\emptyset)}[logp(x|z,\theta)]$一般通过采样的方式进行计算，对于每个样本，根据$q(z|x,\emptyset)$采集 M 个$z^{(m)}, 1 \le m \le M$，

$$\mathrm{E}_{z\sim q(z|\emptyset)}[logp(x|z,\theta)] \approx \frac{1}{M}\sum_{m=1}^{M} logp\left(x|z^{(m)},\theta\right)$$

从 EM 算法角度来看，变分自编码器优化推断网络和生成网络的过程，可以分别看作是 EM 算法中的 E 步和 M 步。但在变分自编码器中，这两步的目标合二为一，都是最大化证据下界。

变分自编码器可以看作神经网络和贝叶斯网络的混合体。贝叶斯网络中的节点可以看成是一个随机变量。在变分自编码器中，我们仅仅将隐藏编码对应的节点看成是随机变量，其它节点还是作为普通神经元。这样，编码器变成一个变分推断网络，而解码器可以看作是将隐变量映射到观测变量的生成网络。

训练

给定一个数据集 D，包含 N 个从未知数据分布中抽取的独立同分布样本$\mathrm{x}^{(1)}$, $\mathrm{x}^{(2)}$, …, $\mathrm{x}^{(N)}$。变分自编码器的目标函数为

$$J(\emptyset,\theta|D) = \sum_{n=1}^{N}(\frac{1}{M}\sum_{m=1}^{M} logp\left(x^{(n)}|z^{(n,m)},\theta\right) - D_{KL}(q(z|x^{(n)},\emptyset)||N(z|0,I)))$$

若采用随机梯度方法，每次从数据集中才一个样本 x，然后根据$q(z|x,\emptyset)$采样一个隐变量 z，则目标函数变为

$$J(\emptyset,\theta|x) = logp(x|z,\theta) - D_{KL}(q(z|x,\emptyset)||N(z|0,I))$$

再参数化 在变分自编码器中，一个问题是如何求随机变量 z 关于参数$\emptyset$的导数。因为随机变量 z 采样自后验分布$q(z|x,\emptyset)$，和参数$\emptyset$相关。但由于是采样的方式，无法直接计算函数 z 关于$\emptyset$的导数。

如果$q(z|x,\emptyset)$的随机性独立于参数$\emptyset$，我们可以通过再参数化（Reparameterization）方法来计算导数。再参数化是实现通过随机变量实现反向传播的一种重要手段，并用随机梯度下降训练整个网络，可以提高变分自编码器的训练效率。

生成对抗网络

显式密度模型和隐式密度模型

假设在低维空间 Z 中有一个简单容易采样的分布 p(z)，p(z)通常为标准多元正态分布 N(0,I)。我们用神经网络构建一个映射函数 G:Z→X，称为生成网络。利用神经网络强大的拟合能力，使得 G(z)服从数据分布$p_r(x)$。这种模型就称为隐式密度模型（Implicit Density Model）。所谓隐式

模型就是指并不对显示地建模$p_r(\boldsymbol{x})$，而是建模生成过程。

网络分解

判别网络

如何确保生成网络产生的样本一定是服从真实的数据分布。既然我们不构建显示密度函数，就无法通过最大似然估计等方法来训练。

生成对抗网络（Generative Adversarial Networks， GAN）是通过对抗训练的方式来使得生成网络产生的样本服从真实数据分布。在生成对抗网络中，有两个网络进行对抗训练。一个是判别网络，目标是尽量准确地判断一个样本是来自于真实数据还是生成网络产生的；另一个是生成网络，目标是尽量生成判别网络无法区分来源的样本。这两个目标相反的网络不断地进行交替训练。当最后收敛时，如果判别网络再也无法判断出一个样本的来源，那么也就等价于生成网络可以生成符合真实数据分布的样本。

判别网络（Discriminator Network） $D(x,\emptyset)$的目标是区分出一个样本 x 是来自于真实分布$p_r(\boldsymbol{x})$还是来自于生成模型$p_\theta(\boldsymbol{x})$，因此判别网络实际上是一个两类分类器。用标签 y=1 来表示样本来自真实分布，y=0 表示样本来自模型，判别网络$D(x,\emptyset)$的输出为 x 属于真实数据分布的概率，即

$$p(y=1|x)=D(x,\emptyset)$$

给定一个样本(x,y)，y={1,0}表示其自于$p_r(\boldsymbol{x})$还是$p_\theta(\boldsymbol{x})$，判别网络的目标函数为最小化交叉熵，即最大化对数似然。

$$\min_{\emptyset} -(E_x[ylogp(y=1|x)]+(1-y)\log p(y=0|x))$$

$$=\max_{\emptyset}\left(E_{x\sim p_r(x)}[logD(x,\emptyset)]+E_{x'\sim p_\theta(x')}\left[\log\left(1-D(x',\emptyset)\right)\right]\right)$$

$$=\max_{\emptyset}\left(E_{x\sim p_r(x)}[logD(x,\emptyset)]+E_{z\sim p_\theta(z)}\left[\log\left(1-D(G(z,\theta),\emptyset)\right)\right]\right)$$

生成网络

生成网络（Generator Network） 目标刚好和判别网络相反，即让判别网络将自己生成的样本判别为真实样本。

$$\max_{\emptyset}\left(E_{z\sim p(z)}\left[\log\left(D(G(z,\theta),\emptyset)\right)\right]\right)=\min_{\emptyset}\left(E_{z\sim p(z)}\left[\log\left(1-D(G(z,\theta),\emptyset)\right)\right]\right)$$

训练

生成对抗网络的训练比较难，往往不太稳定。一般情况下，需要平衡两个网络的能力。对于判别网络来说，一开始的判别能力不能太强，否则难以提升生成网络的能力。然后也不能太弱，否则针对它训练的生成网络也不会太好。在训练时需要使用一些技巧，使得在每次迭代中，判别网络比生成网络的能力强一些，但又不能强太多。

每次迭代时，判别网络更新 K 次而生成网络更新一次，即首先要保证判别网络足够强才能开始训练生成网络。

一个生成对抗网络的具体实现：DCGAN

生成对抗网络是指一类采用对抗训练方式来进行学习的深度生成模型，其包含的判别网络和生成网络都可以根据不同的生成任务使用不同的网络结构。

深度卷积生成对抗网络（Deep Convolutional Generative Adversarial Networks， DCGAN）：判别网络是一个传统的深度卷积网络，但使用了带步长的卷积来实现下采样操作，不用最大汇

算法 13.1: 生成对抗网络的训练过程

输入: 训练集 $\mathcal{D}$，对抗训练迭代次数 T，每次判别网络的训练迭代次数 K，小批量样本数量 M

随机初始化 θ, ϕ;

for $t \leftarrow 1$ **to** T **do**

  // 训练判别网络 $D(\mathbf{x}, \phi)$

  for $k \leftarrow 1$ **to** K **do**

    // 采集小批量训练样本

    从训练集 $\mathcal{D}$ 中采集 M 个样本 $\{\mathbf{x}^{(m)}\}, 1 \leq m \leq M$;

    从分布 $\mathcal{N}(\mathbf{0}, \mathbf{I})$ 中采集 M 个样本 $\{\mathbf{z}^{(m)}\}, 1 \leq m \leq M$;

    使用随机梯度上升更新 ϕ，梯度为

$$\frac{\partial}{\partial \phi}\left[\frac{1}{M}\sum_{m=1}^{M}\left(\log D(\mathbf{x}^{(m)}, \phi) + \log\left(1 - D\big(G(\mathbf{z}^{(m)}, \theta), \phi\big)\right)\right)\right];$$

  end

  // 训练生成网络 $G(\mathbf{z}, \theta)$

  从分布 $\mathcal{N}(\mathbf{0}, \mathbf{I})$ 中采集 M 个样本 $\{\mathbf{z}^{(m)}\}, 1 \leq m \leq M$;

  使用随机梯度上升更新 θ，梯度为

$$\frac{\partial}{\partial \theta}\left[\frac{1}{M}\sum_{m=1}^{M} D\big(G(\mathbf{z}^{(m)}, \theta), \phi\big)\right];$$

end

输出: 生成网络 $G(\mathbf{z}, \theta)$

聚（pooling）操作。生成网络使用一个特殊的深度卷积网络来实现。

通过一些经验性的网络结构设计使得对抗训练更加稳定：使用代步长的卷积（在判别网络中）和微步卷积（在生成网络中）来代替汇聚操作，以免损失信息；使用批量归一化；去除卷积层之后的全连接层；在生成网络中，除了最后一层使用 Tanh 激活函数外，其余层都使用 ReLU 函数；在判别网络中，都使用 LeakyReLU 激活函数。

模型分析

将判别网络和生成网络合并，整个生成对抗网络的整个目标函数看作最小化最大化游戏（Minimax Game）：

$$\min_{\theta}\max_{\emptyset}\left(E_{x\sim p_r(x)}[logD(x,\emptyset)] + E_{x'\sim p_\theta(x')}\big[\log\big(1 - D(x',\emptyset)\big)\big]\right)$$

$$= \min_{\theta}\max_{\emptyset}\left(E_{x\sim p_r(x)}[logD(x,\emptyset)] + E_{z\sim p(z)}\big[\log\big(1 - D(G(z,\theta),\emptyset)\big)\big]\right)$$

假设$p_r(\boldsymbol{x})$和$p_\theta(x)$已知，则最优的判别器为

$$D^*(x) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_\theta(x)}$$

将最有判别器$D^*(x)$代入目标函数：

$$L(G|D^*) = E_{x\sim p_r(x)}[logD^*(x)] + E_{x\sim p_\theta(x)}[\log(1-D^*(x))]$$

$$= E_{x\sim p_r(x)}\left[log\frac{p_r(x)}{p_r(x)+p_\theta(x)}\right] + E_{x\sim p_\theta(x)}\left[\log\frac{p_\theta(x)}{p_r(x)+p_\theta(x)}\right]$$

$$= D_{KL}(p_r||p_a) + D_{KL}(p_\theta||p_a) - 2log2 = 2D_{JS}(p_r||p_\theta) - 2log2$$

当判断网络为最优时，生成网络的优化目标是最小化真实分布p_r和模型分布p_θ之间的 JS 散度。当两个分布相同时，JS 散度为 0，最优生成网络G*对应的损失为$L(G^*|D^*)$=-2log2。当两个分布没有重叠时，它们之间的 JS 散度恒等于常数 log2。对生成网络来说，目标函数关于参数的梯度为 0。

$$\frac{\partial L(G|D^*)}{\partial\theta} = 0$$

在实际训练生成对抗网络时，我们一般不会将判别网络训练到最优，只进行一步或多步梯度下降，使得生成网络的梯度依然存在。然而，判别网络也不能太差，否则生成网络的梯度为错误的梯度。

模型坍塌

$$\begin{aligned}
\mathcal{L}'(G|D^*) &= \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\log D^*(\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\log\frac{p_r(\mathbf{x})}{p_r(\mathbf{x})+p_\theta(\mathbf{x})}\cdot\frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})}\right] \\
&= -\mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\log\frac{p_\theta(\mathbf{x})}{p_r(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\log\frac{p_\theta(\mathbf{x})}{p_r(\mathbf{x})+p_\theta(\mathbf{x})}\right] \\
&= -D_{\mathrm{KL}}\left(p_\theta\|p_r\right) + \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\log\left(1-D^*(\mathbf{x})\right)\right] \\
&= -D_{\mathrm{KL}}\left(p_\theta\|p_r\right) + 2D_{\mathrm{JS}}(p_r\|p_\theta) - 2\log 2 - \mathbb{E}_{\mathbf{x}\sim p_r(\mathbf{x})}\left[\log D^*(\mathbf{x})\right]
\end{aligned}$$

$$\max_\theta L'(G|D^*) = \min_\theta D_{KL}(p_\theta||p_r) - 2D_{JS}(p_r||p_\theta)$$

其中 JS 散度$D_{JS}(p_r||p_\theta) \in [0, log2]$为有界函数，因此生成网络的目标为更多的是受逆向 KL 散度$D_{KL}(p_\theta||p_r)$影响，使得生成网络更倾向于生成一些更“安全”的样本，从而造成模型坍塌（Model Collapse）问题。

前向和逆向 KL 散度 因为 KL 散度是一种非对称的散度，在计算真实分布p_r和模型分布p_θ之间的 KL 散度时，按照顺序不同，有两种 KL 散度：前向 KL 散度（Forward KL divergence）$D_{KL}(p_r||p_\theta)$和逆向 KL 散度（Reverse KL divergence）$D_{KL}(p_\theta||p_r)$。前向和逆向 KL 散度分别定义为

$$D_{KL}(p_r||p_\theta) = \int p_r(x)log\frac{p_r(x)}{p_\theta(x)}dx$$

$$D_{KL}(p_\theta||p_r) = \int p_\theta(x)log\frac{p_\theta(x)}{p_r(x)}dx$$

在前向 KL 散度中，（1）当$p_r(x)\to 0$而$p_\theta(x)>0$时，$p_r(x)log\frac{p_r(x)}{p_\theta(x)}\to 0$。不管$p_\theta(x)$如何取值，都对前向 KL 散度的计算没有贡献；（2）当$p_r(x)>0$而$p_\theta(x)\to 0$时，$p_r(x)log\frac{p_r(x)}{p_\theta(x)}\to\infty$，前向 KL 散度会变得非常大。因此，前向 KL 散度会鼓励模型分布$p_\theta(x)$尽可能覆盖所有真实分布$p_r(x)>0$的点，而不用回避$p_r(x)\approx 0$的点。

在逆向 KL 散度中，（1）当$p_r(x)$→0 而$p_\theta(x)$>0 时，$p_\theta(x)log\frac{p_\theta(x)}{p_r(x)}$→∞。即当$p_\theta(x)$接近于 0，而$p_\theta(x)$有一定的密度时，逆向 KL 散度会变得非常大。（2）当$p_\theta(x)$→0 时，不管$p_r(x)$如何取值，$p_\theta(x)log\frac{p_\theta(x)}{p_r(x)}$→0。因此，逆向 KL 散度会鼓励模型分布$p_\theta(x)$尽可能避开所有真实分布$p_r(x) \approx 0$的点，而不需要考虑是否覆盖所有布$p_r(x) > 0$的点。

改进模型

GAN 中交叉熵（JS 散度）不适合衡量生成数据分布和真实数据分布的距离，如果通过优化 JS 散度训练 GAN 会导致找不到正确的优化目标。

W-GAN：通过用 Wassertein 距离替代 JS 散度来优化训练的生成对抗网络。

对于真实分布 p_r 和模型分布 p_θ，它们的 1st-Wasserstein 距离为

$$W^1(p_r, p_\theta) = \inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}\Big[||\mathbf{x}-\mathbf{y}||\Big], \tag{13.51}$$

其中 $\Gamma(p_r, p_\theta)$ 是边际分布为 p_r 和 p_θ 的所有可能的联合分布集合。

当两个分布没有重叠或者重叠非常少时，它们之间的 KL 散度为+∞，JS 散度为 log2，并不随着两个分布之间的距离而变化。而 1st-Wasserstein 距离可以依然衡量两个没有重叠分布之间的距离。

根据 Kantorovich-Rubinstein 对偶定理，两个分布 p_r 和 p_θ 之间的 1st-Wasserstein 距离的对偶形式为：

$$W^1(p_r, p_\theta) = \sup_{\|f\|_L \le 1} \mathbb{E}_{\mathbf{x}\sim p_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_\theta}[f(\mathbf{x})], \tag{13.52}$$

其中 $f : \mathbb{R}^d \to \mathbb{R}$ 为 1-Lipschitz 函数，满足

$$\|f\|_L \triangleq \sup_{\mathbf{x}\neq\mathbf{y}} \frac{|f(\mathbf{x}) - f(\mathbf{y})|}{|\mathbf{x}-\mathbf{y}|} \le 1. \tag{13.53}$$

我们可以将 1-Lipschitz 连续的约束宽松为 K-Lipschitz 连续，等于计算 p_r 和 p_θ 之间的 $K \cdot W^1(p_r, p_\theta)$。

令 $f(\mathbf{x}, \phi)$ 为一个神经网络，假设存在参数集合 Φ，对于所有的 $\phi \in \Phi$，$f_\phi(\mathbf{x})$ 为 K-Lipschitz 连续函数。那么公式（13.52）中的上界可以转换为

$$\max_{\phi\in\Phi} \mathbb{E}_{x\sim p_r}[f(\mathbf{x}, \phi)] - \mathbb{E}_{x\sim p_\theta}[f_(\mathbf{x}, \phi)], \tag{13.55}$$

其中 $f(\mathbf{x}, \phi)$ 称为评价网络（Critic Network）。对于真实样本，$f(\mathbf{x}, \phi)$ 的打分要尽可能的高；对于模型生成的样本，$f(\mathbf{x}, \phi)$ 的打分要尽可能的低。和标准 GAN 中的判别网络的值域为 $[0, 1]^l$ 不同，评价网络 $f(\mathbf{x}, \phi)$ 的值域没有限制。

生成网络的目标是使得生成样本的 $f(\mathbf{x}, \phi)$ 得分尽可能高。

$$\max_\theta \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\Big[f\Big(G(\mathbf{z}, \theta), \phi\Big)\Big]. \tag{13.56}$$

因为 $f(\mathbf{x}, \phi)$ 为不饱和函数，所以生成网络参数 θ 的梯度不会消失，理论上解决了原始 GAN 训练不稳定的问题。并且 W-GAN 中生成网络的目标函数不再是两个分布的比率，在一定程度上缓解了模型坍塌问题，使得生成的样本具有多样性。

算法13.2给出 W-GAN 的训练过程。和原始 GAN 相比，W-GAN 的评价网络最后一层不使用 sigmoid 函数，损失函数不取对数。