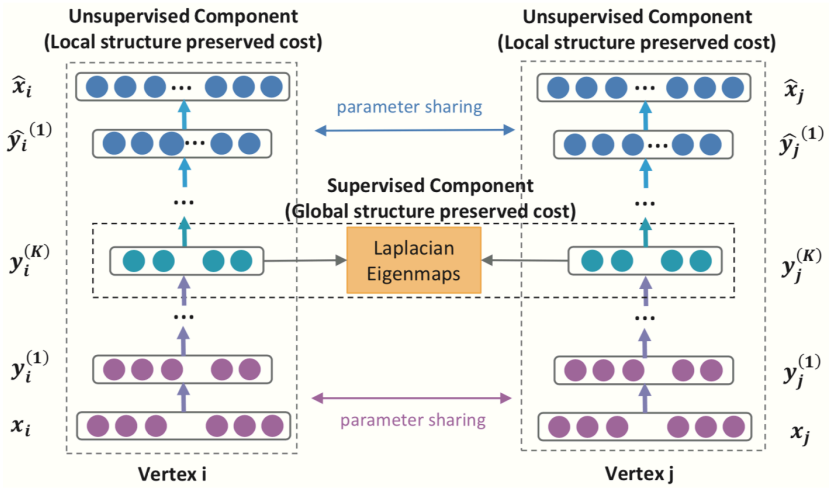
本文总结了 2016 年以来多篇基于 NN 的 network embedding 方法。

1. Structural deep network embedding（SDNE）

本方法基于 auto-encoder，认为浅层模型不足以捕捉网络的高度非线性结构，所以提出了一种半监督的深度模型，从而包含了多个非线性函数；类似于 line，试图同时考虑顶点间的一阶相似度和二阶相似度来保留网络局部和全局的网络特征。设计了一个无监督的部分（AE）来保留二阶相似度（点的邻居相似度），同时由于有顶点直接相连的信息（一阶相似度），利用这些信息在 latent space 进行了一个有监督的学习。网络结构以及符号表示如下：



Symbol	Definition
n	number of vertexes
K	number of layers
$S = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$	the adjacency matrix for the network
$X = \{\mathbf{x}_i\}_{i=1}^n, \hat{X} = \{\hat{\mathbf{x}}_i\}_{i=1}^n$	the input data and reconstructed data
$Y^{(k)} = \{\mathbf{y}_i^{(k)}\}_{i=1}^n$	the k-th layer hidden representations
$W^{(k)}, \hat{W}^{(k)}$	the k-th layer weight matrix
$\mathbf{b}^{(k)}, \hat{\mathbf{b}}^{(k)}$	the k-th layer biases
$\theta = \{W^{(k)}, \hat{W}^{(k)}, \mathbf{b}^{(k)}, \hat{\mathbf{b}}^{(k)}\}$	the overall parameters

在上表中，S 是网络的邻接矩阵，$\mathbf{s}_i = \{s_{i,j}\}_{j=1}^n$，如果 ij 之间有边连接，$s_{ij}>0$。为了得到 latent representation $Y^{(k)}$，我们需要优化重建误差：

$$\mathcal{L} = \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2$$

其中 $x_i = s_i$，由于 si 非常稀疏，可能会出现重建的结果都是零的情况，此外，两点之间是否存在链接应该具有不同的置信度，这是因为两点不存在链接不一定意味着这两点不相似，而存在链接的两点一定是相似的。所以目标函数修改为下式：

$$\mathcal{L}_{2nd} = \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b}_i\|_2^2$$
$$= \|(\hat{X} - X) \odot B\|_F^2$$

如果 $s_{ij}=0$，则 $b_{ij}=1$，否则 $b_{ij}=\beta>1$。该学习过程虽然没有显式捕捉 si 之间的相似度，但是因为要重建 si，邻居的信息会保留在 latent representation 中。此外，

SDNE 还利用了一阶相似度，约束了有链接的顶点的 latent representations 应该相似，这可以看做是一个有监督学习，要求相似的点 embedding 的距离近：

$$\begin{aligned}\mathcal{L}_{1st} &= \sum_{i,j=1}^{n} s_{i,j}\|\mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)}\|_2^2 \\ &= \sum_{i,j=1}^{n} s_{i,j}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2\end{aligned}$$

SDNE 的整体的目标函数如下，最后一项是一个 L2 正则化项：

$$\begin{aligned}\mathcal{L}_{mix} &= \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg} \\ &= \|(\hat{X} - X) \odot B\|_F^2 + \alpha \sum_{i,j=1}^{n} s_{i,j}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 + \nu\mathcal{L}_{reg}\end{aligned}$$

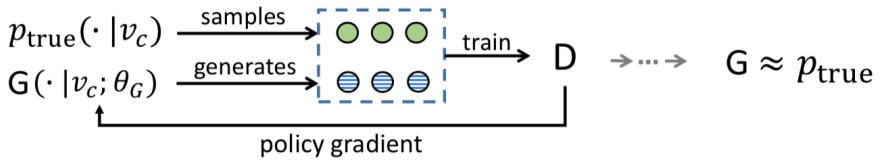
$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{k=1}^{K}(\|W^{(k)}\|_F^2 + \|\hat{W}^{(k)}\|_F^2)$$

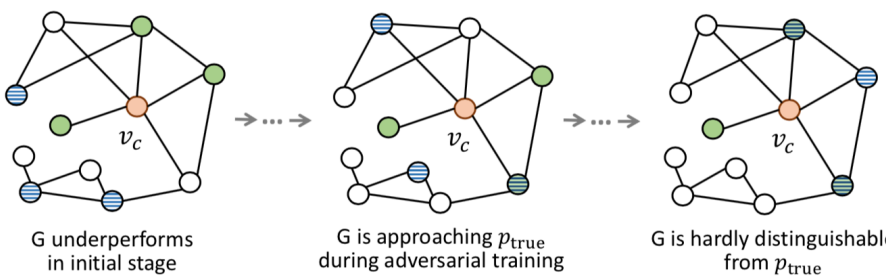
2. GraphGAN

本文将现有的 GE 方法分为两种：一种是生成模型，假设对于每个点 vc，存在一个潜在的真实邻接分布 $p_{true}(v|v_c)$，那么与 vc 相邻的顶点就可以看作是基于这个条件分布的采样，生成模型通过最大似然这些边来学习 embedding 结果，比如 deepwalk；另一种是判别模型，这种方法并不认为边是从潜在的条件分布中采样得到的，而是将两点都作为特征想要学习一个分类器，预测两点之间是否有边，也就是得到概率 p(edge|(vi,vj))。融合这两者的模型有 LINE，本文则是利用 GAN 来融合。

GraphGAN 的框架如下：首先存在真实邻接分布 $p_{true}(v|v_c)$，$N(v_c)$是与 v_c 相邻的点的集合，可以看做是从真实邻接分布 $p_{true}(v|v_c)$中的采样；生成器 $G(v|v_c;\theta_G)$是对真实分布的近似，可以用来生成与 v_c 可能相邻的点；判别器 $D(v,v_c;\theta_D)$生成一个标量表示这两点相邻的概率。GraphGAN 的 value function 为：

$$\begin{aligned}\min_{\theta_G}\max_{\theta_D} V(G,D) = \sum_{c=1}^{V}\Big(&\mathbb{E}_{v\sim p_{\text{true}}(\cdot|v_c)}\big[\log D(v,v_c;\theta_D)\big] \\ &+ \mathbb{E}_{v\sim G(\cdot|v_c;\theta_G)}\big[\log\big(1 - D(v,v_c;\theta_D)\big)\big]\Big).\end{aligned}$$





本方法要训练的参数为 θ_G 和 θ_D，两者分别为网络节点的两种低维向量表示，最后使用生成器的参数作为学到的 embedding 结果。判别器很简单的建模为两个点低维表示的内积，再经过 sigmoid。训练时类似于逻辑回归，将来自真实分布的点作为正样本，来自生成器的点作为负样本，使用最大似然法学习 label 即可。

$$D(v, v_c) = \sigma(\mathbf{d}_v^\top \mathbf{d}_{v_c}) = \frac{1}{1 + \exp(-\mathbf{d}_v^\top \mathbf{d}_{v_c})}$$

$$\nabla_{\theta_D} V(G, D) = \begin{cases} \nabla_{\theta_D} \log D(v, v_c), \ if \ v \sim p_{\text{true}}; \\ \nabla_{\theta_D} \big(1 - \log D(v, v_c)\big), \ if \ v \sim G. \end{cases}$$

而生成器的训练要更复杂一点，不同于图像生成，样本直接通过 NN 得到，从而训练生成器参数时梯度可以传回去，生成网络顶点是通过对一个离散分布采样得到的，无法直接进行梯度下降，本文提出使用 policy gradient 的方法：

$$\begin{aligned} & \nabla_{\theta_G} V(G, D) \\ =& \nabla_{\theta_G} \sum_{c=1}^{V} \mathbb{E}_{v \sim G(\cdot|v_c)} \big[\log\big(1 - D(v, v_c)\big)\big] \\ =& \sum_{c=1}^{V} \sum_{i=1}^{N} \nabla_{\theta_G} G(v_i|v_c) \log\big(1 - D(v_i, v_c)\big) \\ =& \sum_{c=1}^{V} \sum_{i=1}^{N} G(v_i|v_c) \nabla_{\theta_G} \log G(v_i|v_c) \log\big(1 - D(v_i, v_c)\big) \\ =& \sum_{c=1}^{V} \mathbb{E}_{v \sim G(\cdot|v_c)} \big[\nabla_{\theta_G} \log G(v|v_c) \log\big(1 - D(v, v_c)\big)\big]. \end{aligned}$$

GraphGAN 还提出了一种新的 softmax 的方法对生成器进行建模，如果直接使用 softmax 方法，如下式所示，对每一个点更新时都要计算所有的点

$$G(v|v_c) = \frac{\exp(\mathbf{g}_v^\top \mathbf{g}_{v_c})}{\sum_{v \neq v_c} \exp(\mathbf{g}_v^\top \mathbf{g}_{v_c})}$$

本文认为一个好的 softmax 的方法是要满足可归一化、图结构感知、计算有效这三点需求，负采样不满足第一点，它更多的是一种优化的算法无法满足 $\sum_{v \neq v_c} G(v|v_c; \theta_G) = 1$，而生成器则需要得到一个合法的概率分布，优化的步骤靠判别器传梯度；第二点是希望两点相连的概率随着他们的最短路径长度增加而降低；本文提出了 graph softmax 方法，计算分布 $G(*|v_c;\theta_c)$使，先以 v_c 为根，通过 BFS 算法得到一棵 BFS 树，$N_c(v)$是 v 的邻居点集合，那么 $v_i \in N_c(v)$（包括子节点和父节点）在 v 下的条件概率为：

$$p_c(v_i|v) = \frac{\exp(\mathbf{g}_{v_i}^\top \mathbf{g}_v)}{\sum_{v_j \in \mathcal{N}_c(v)} \exp(\mathbf{g}_{v_j}^\top \mathbf{g}_v)}$$

在 BFS 树上从根到每个点 v 都有唯一的一条路径 $P_{v_c \to v} = (v_{r_0}, v_{r_1}, ..., v_{r_m})$，其中第一个点是 v_c，第 m 个点为 v，那么生成

各点的概率由下式给出：

$$G(v|v_c) \triangleq \left(\prod_{j=1}^{m} p_c(v_{r_j}|v_{r_{j-1}})\right) \cdot p_c(v_{r_{m-1}}|v_{r_m})$$

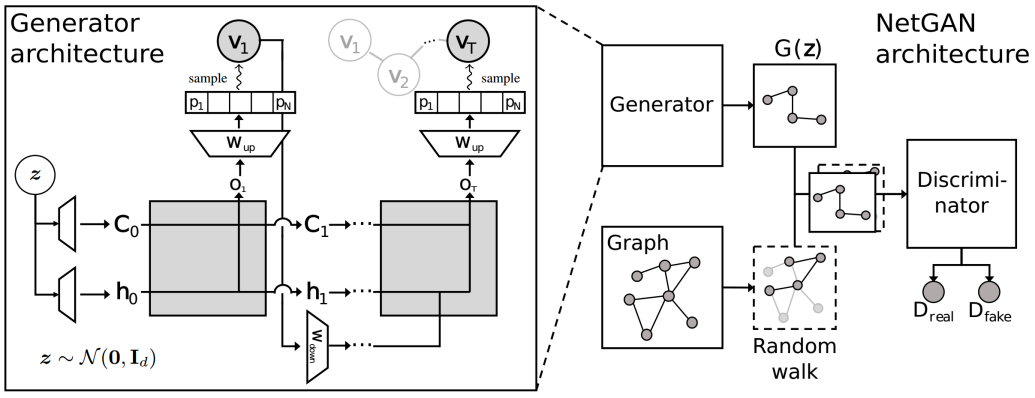
可以证明该定义满足上述三点，最终算法如下：

Algorithm 2 GraphGAN framework

Require: dimension of embedding k, size of generating samples s, size of discriminating samples t
Ensure: generator $G(v|v_c;\theta_G)$, discriminator $D(v,v_c;\theta_D)$
1: Initialize and pre-train $G(v|v_c;\theta_G)$ and $D(v,v_c;\theta_D)$;
2: Construct BFS-tree T_c for all $v_c \in \mathcal{V}$;
3: **while** GraphGAN not converge **do**
4: **for** G-steps **do**
5: $G(v|v_c;\theta_G)$ generates s vertices for each vertex v_c according to Algorithm 1;
6: Update θ_G according to Eq. (4), (6) and (7);
7: **end for**
8: **for** D-steps **do**
9: Sample t positive vertices from ground truth and t negative vertices from $G(v|v_c;\theta_G)$ for each vertex v_c;
10: Update θ_D according to Eq. (2) and (3);
11: **end for**
12: **end while**
13: **return** $G(v|v_c;\theta_G)$ and $D(v,v_c;\theta_D)$

3. NetGAN

NetGAN 的主要思路是通过 random walk 来学习网络的拓扑结构。给定输入图的 N 个顶点，定义一个 N*N 邻接矩阵 A∈{0,1}。从 A 中采样长度为 T 的一些 random walks，采样方法是 Node2vec 中提出的二阶 random walk 采样策略以更好的捕捉局部和全局的图结构，采样的结果作为正样本。生成器的目标是生成一些模拟 random walk 的序列，判别器的目标是区分输入的序列是否是 random walk 的结果。由于生成器可以产生 random walk 的集合，所以可以产生一个生成图的邻接矩阵。整体的模型如下：



(a) (b)

生成器的部分使用了 LSTM 技术，神经网络 f_θ 在每一步 t 都产生两个值：要采样的下一个的概率分布 p_t，当前的 LSTM 单元维持的“记忆”状态 m_t（包括 LSTM 中的单元状态 C_t 和隐藏状态 h_t），从 p_t 中的采样和 m_t 又作为下一步函数 f_θ 的输入。z 是一个服从多元正态分布的隐变量，z 的采样经过一个全连接层得到初始的状态值 m_0。过程如下：

$$
\begin{aligned}
& & \boldsymbol{z} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d) \\
& & \boldsymbol{m}_0 &= g_{\theta'}(\boldsymbol{z}) \\
\boldsymbol{v}_1 &\sim \mathrm{Cat}(\boldsymbol{p}_1), & (\boldsymbol{p}_1, \boldsymbol{m}_1) &= f_\theta(\boldsymbol{m}_0, \mathbf{0}) \\
\boldsymbol{v}_2 &\sim \mathrm{Cat}(\boldsymbol{p}_2), & (\boldsymbol{p}_2, \boldsymbol{m}_2) &= f_\theta(\boldsymbol{m}_1, \boldsymbol{v}_1) \\
&\vdots & &\vdots \\
\boldsymbol{v}_T &\sim \mathrm{Cat}(\boldsymbol{p}_T), & (\boldsymbol{p}_T, \boldsymbol{m}_T) &= f_\theta(\boldsymbol{m}_{T-1}, \boldsymbol{v}_{T-1})
\end{aligned}
$$

本文选择建模一个长序列，而不是像 GraphGAN 建模浅层模型，文章认为虽然使用长度为 2 的 random walk（也就是浅层模型），在模型有足够的容量时是可以保存所有存在的边，但是 NetGAN 并不想单纯的拷贝已有的图，而是希望能生成合理的图，使用长序列并且“记忆”前几步的状态可以更好的学习图的拓扑结构。

生成器采用了两个 tricks，一个是在每个时间步都要计算 N 维的概率分布 p_t，这需要很多的计算时间，所以本文选择生成一个 H 维的概率分布 o_t，然后用一个 N*H 维的矩阵 W 进行上投影到 R^N。第二，和 GraphGAN 一样，由于要从一个分类分布中采样一个离散变量，导致在梯度下降时梯度无法从这个采样值传回去，（如果是连续变量可以使用 reparameterization trick，类似 VAE 的处理），本文使用了 Gumbel softmax 的方法，π 是分类分布的参数 pt，ϵ是独立同分布满足 Gumbel 标准分布的随机变量：

$$
(x_k)_{1\le k\le K} = \mathbf{softmax}((\epsilon_k + \log\pi_k)_k) \Leftrightarrow x_k = \frac{\exp\left((\log\pi_k+\epsilon_k)/\tau\right)}{\sum_j \exp\left((\log\pi_j+\epsilon_j)/\tau\right)}
$$

判别器基于标准 LSTM，每一个时间步输入 random walk 的一个节点，最后输出一个标量。训练模型基于 Wasserstein GAN。

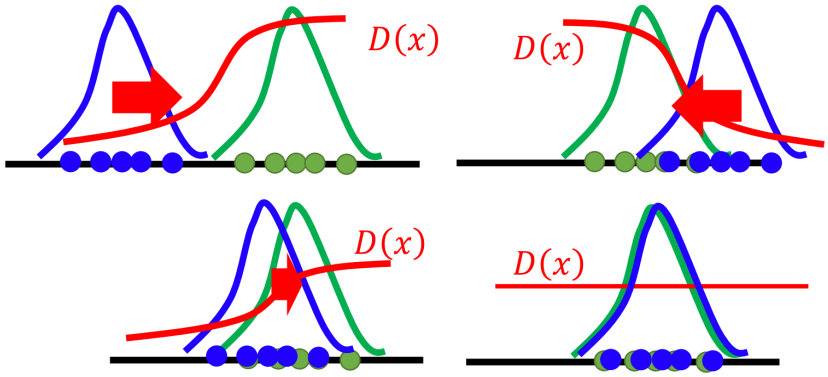
在生成图像的 GAN 网络中，D 的任务是区分是否是真实图像，由于是 G 是一个连续的函数，z 的一点改变就可能产生出完全不同的逼真图像，所以训练到收敛仍可以保证 G 具有生成能力。而在本任务中，D 区分序列是否是从已有 graph 中生成，如果我们将模型训练到收敛，那么相当于是记忆了已有的 graph，生成的序列是完全可以从已有 graph 中能得到的 random walk，而使模型丧失了泛化能力。NetGAN 提出了两种提前终止的策略，第一种称为 VAL-CRITERION，在训练的过程中维持一个最近的 1000 次迭代中的序列，用来进行 link prediction，直到预测结果不再提高。第二种称为 EO-CRITERION，指定原 graph 的一些边，当生成 graph 覆盖这些边时终止，这样可以方便的权衡希望生成 graph 更接近原 graph 还是具有更好的生成能力。

训练完成后，如何得到生成的 binary 的邻接矩阵呢，NetGAN 先用大量的生成的 random walk 进行对每条边的出现计数，得到计数矩阵 S，然后处理成对称矩阵 $s_{ij} = s_{ji} = \max\{s_{ij}, s_{ji}\}$，在矩阵 S 中高 degree 的顶点会反复出现，而低 degree 的顶点出现频率很低，如果直接进行二值化，可能会产生很多孤立的点，所以先对每一个顶点 i 采样一条边(i,j)：$p_{ij} = \frac{s_{ij}}{\sum_v s_{iv}}$，然后再广泛的采样边：$p_{ij} = \frac{s_{ij}}{\sum_{u,v} s_{uv}}$，直到达到指定数量。

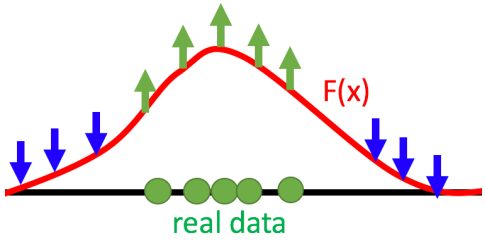
四．GAN-HBRN

4.1 energy-based GAN

在普通的 GAN 中，判别器起到的作用是评估两个 divergence 的区别，引导生成器学习的方向，下图展示了每次生成器训练完成之后，最优判别器 $D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$ 对生成器的分布的引导，当生成分布和真实分布重合时，判别器变成一条直线，不包含任何信息。如果只是为了起到这个作用，判别器的网络可以比生成器简单，因为不需要它去描述数据的分布，甚至可以是 linear 的（一条倾斜直线）。



但是关于 GAN 还有另一种解释，就是认为判别器具有描述数据分布的能力，（真实训练中使用 D 的网络结构往往也是很复杂的），比如在 improved GAN 的中就提出，由于 D 比 G 学的快，所以让 G 去跟 D 学习。如果将判别器作为一个 evaluation function，对于一个输入的 x 会赋予不同的分值，对真实数据赋予较大的分值，对非真实数据赋予小分值，由于我们无法穷举所有非真实数据，所以将生成器看做是一种负采样。对采样的打分值引导生成器的学习方向，当生成器能生成真实数据的样本时，判别器仍然保留了打分值（真实分布的信息），而不是变成一条直线。这就是 energy-based（EBGAN）的观点。



能量的概念来自于物理，一个物理系统具有不同的状态，当系统和它周围的环境处于热平衡时，系统处于状态 i 的概率：

$$p_i = \frac{1}{Z_T} e^{-\frac{E_i}{T}}$$

其中 Zt 是归一化常数，T 是系统所处的温度：

$$Z_T = \sum_i e^{-\frac{E_i}{T}}$$

基于能量的模型，通常是根据需求定义一个能量函数 E(x)，即将 x 映射到一个标量，得到能量函数后就可以计算该 x 出现的概率，也就是说，一个变量的取值出现的概率越大，它具有的能量越小。实际上，能量函数可以当做是一个 evaluation function，打分的范围没有限制，因为计算概率时会进行归一化。

EBGAN 将判别器的作为一个待训练 energy function，对与真实数据赋予较低的能量值，引导生成器产生低能量值的数据，判别器具体使用的模型是 autoencoder，将重建误差作为能量值。

。因此设计了两个 loss，分别训练判别器和生成器

$$\mathcal{L}_D(x,z) = D(x) + [m - D(G(z))]^+$$
$$\mathcal{L}_G(z) = D(G(z))$$

其中$[*]^+ = \max(0,*)$，这样判别器 loss 中使用了一个 margin loss，这是因为对数据 x，减小它的能量是相对困难的，而增大能量非常简单，因为训练网络生成和原图相似的图片比生成完全不同的图片要难。如果没有加 margin，增大生成数据的能量同时减小真实数据的能量，结果会主要做前者；反之当 D(G(z)) ≥ m 之后就没有梯度使它继续增大。

定义：

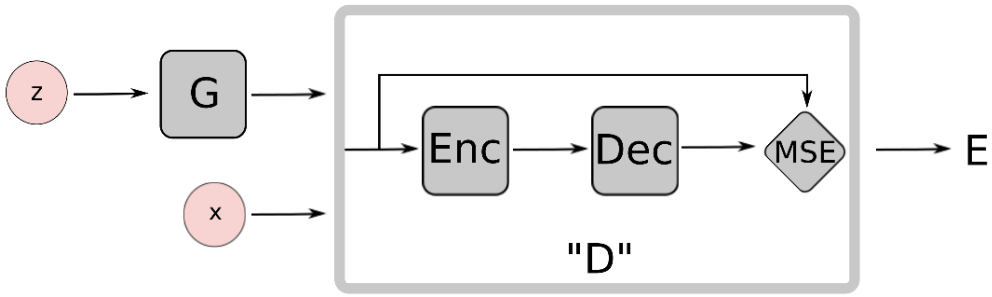
$$V(G,D) = \int_{x,z} \mathcal{L}_D(x,z) p_{data}(x) p_z(z)$$

$$U(G,D) = \int_z \mathcal{L}_G(z) p_z(z) \mathrm{d}z$$

训练 D 时最小化 V，训练 G 时最小化 U。当系统达到纳什均衡时，在所有点都有 $P_{data}(x) = P_G(x)$，且 $V(G^*,D^*)=m$。在 $P_{data}(x)$不为 0 的区域，$D^*(x)=\gamma \in [0,m]$。最后的 D 的图像是一个有凹陷的曲线，而不是一条不包含信息的直线。理论上这是由于 AE 有 bottleneck，不会 Identity 的还原所有 x 的信息，只有少量的区域可以被正确的重建，而真实数据被训练为低 energy 其他区域就会是高 energy。

判别器具体的定义如下，结构如图所示：

$$D(x) = ||Dec(Enc(x)) - x||$$



最后，为了解决 mode collapse 问题，本文还提出了一个 pull-away term，类似于 minibatch discrimination，PT 也是将一个 batch 输入到 D 中而不是单一的 x，然后在 encoder output 层，减小两两之间的 cos 相似度，以增加 generate 出来的各图片的 diversity。

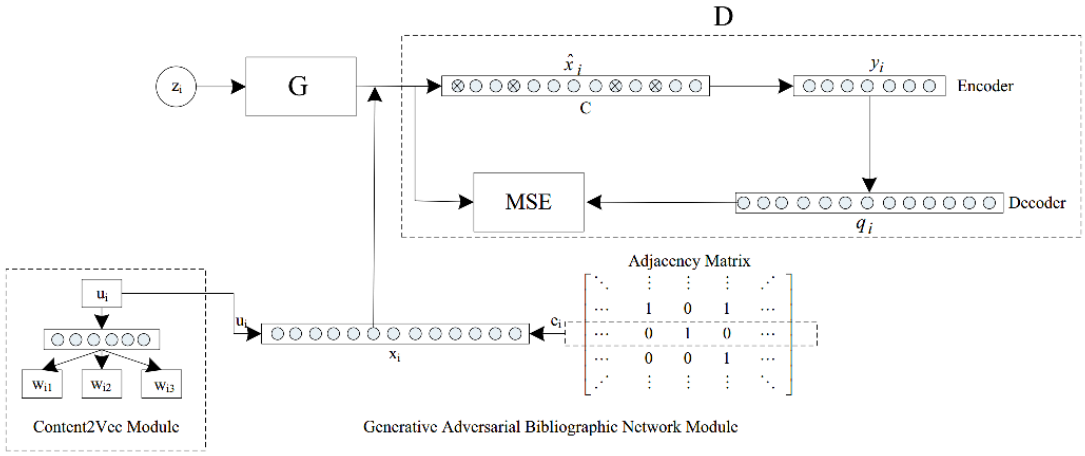
$$f_{PT}(S) = \frac{1}{N(N-1)} \sum_i \sum_{j \neq i} \left(\frac{S_i^\top S_j}{\|S_i\| \|S_j\|} \right)^2$$

EBGAN 也可以看做是一种优化的 AE 方法，一般的 AE 只是减小样本的重建误差，可能会出现 AE 网络记忆了所有的图片，成为一个 identity function，常见的解决手段有限制 bottleneck 的大小，加 noise 等等。而本方法由于要增大生成器生成的负样本的重建误差，是一种有效的 regular 的手段。

4.2 GAN-HBNR

基于 EBGAN， Generative Adversarial Network based Heterogeneous Bibliographic Network Representation（GAN-HBNR）提出了一种学习异构书目网络表示的方法，能够同时利用顶点内容信息和网络结构，学到的分布式表示反过来可以计算顶点之间的相似性从而实现了推荐的效果。

网络结构记为 G=<V,E,C>，其中 V 包括表示论文的 n 个顶点和表示作者的 m 个顶点，E 包括描述论文之间、作者之间、论文和作者的三种类型的边，边是 binary 的，C 是每个顶点相关的内容信息。最后，B 是一个(n+m)*(n+m)的邻接矩阵。模型如下：



模型包括两部分，首先对每个顶点 vi 使用 doc2vec 的方法得到一个内容表示 ui，优化目标为：

$$O = \sum_{i=1}^{n+m} \log P(w_{-b} : w_b \mid v_i)$$

得到 ui 后，将顶点 vi 和其他所有点的邻接关系表示为向量 ei，拼接 ui 和 ei 作为 GAN 的输入 xi。GAN 部分使用的是 EBGAN 模型，为了增加鲁棒性，在判别器的部分使用了 denoising AE。使用 encoder 的输出也就是向量 y 作为新的网络表示。本方法的一个优势是对于新顶点，可以直接输入 encoder 得到 embedding 结果。

本方法得到了论文和作者表示后，主要想要用于论文引用推荐，方法越很简单，直接计算待预测论文与其他论文表示的内积，即可得到相似度。

本方法虽然用了 GAN，但主要起作用的是判别器中的 AE 部分，生成器的存在主要是是为了提高 AE 的效果。得到的 embedding 结果只能用于推荐，而不具有空间上的关系（不连续）。