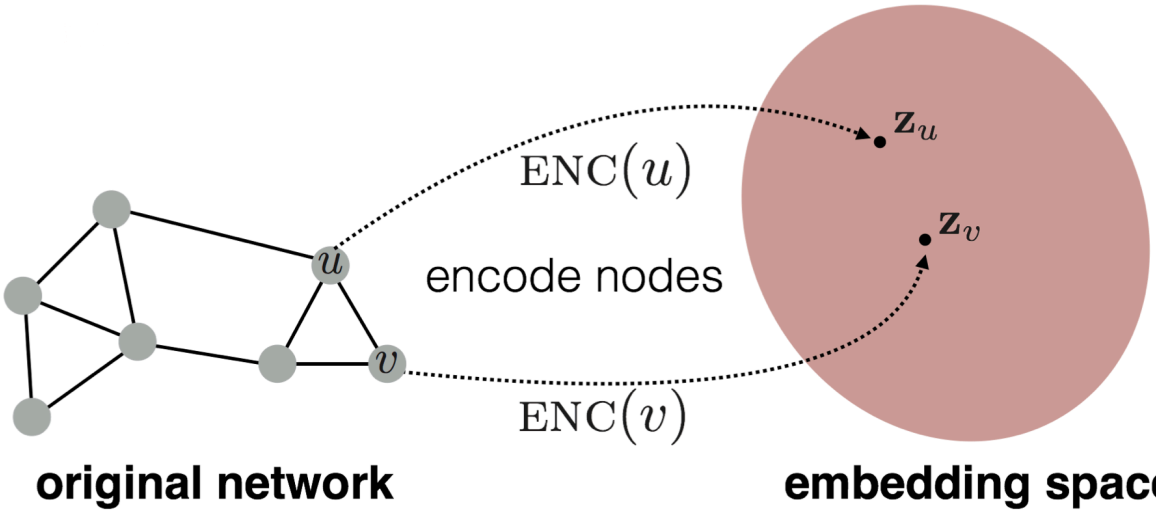
Node Representation Learning（节点表示学习）

在本节中，我们研究了几种在嵌入空间中表示图形的方法。“嵌入”是指将网络中的每个节点映射到一个低维空间，这将使我们深入了解节点的相似性和网络结构。鉴于因特网和物理世界中图结构的广泛存在，图形的表示学习在诸如连接预测和异常检测之类的应用中起着重要作用。但是，现代机器学习算法是为简单的序列或网格（例如，固定大小的图像/网格或文本/序列）设计的，但是网络通常具有复杂的拓扑结构和多模型特征。我们将探索嵌入方法来解决这些难题。

Embedding Nodes（节点嵌入）

节点嵌入的目标是对节点进行编码，以使嵌入空间(例如点积)中的相似度近似于原始网络中的相似度，我们将探讨的节点嵌入算法通常包括三个基本阶段：

1. 定义编码器(即从节点到嵌入的映射)。下面我们提供了一个图表来说明编码过程，将地图节点 u和v 映射到低维向量 z_u 和z_v：



2. 定义节点相似性函数（即原始网络中相似性的度量），它明确了向量空间中到原始网络中的映射关系。
3. 优化编码器的参数，以使u和v 在原网络中与在节点嵌入之间的点积近似：
$simailary(u,v) \approx z_u^T z_v$

"Shallow" Encoding（浅编码）

如何定义编码器以将节点映射到嵌入空间?
“浅”编码是最简单的编码方法，编码器只是一个嵌入查找，它可以表示为：

$$\mathrm{ENC(v)} = \mathrm{Zv}$$
$$\mathrm{Z} \in \mathbb{R}^{\mathrm{d}\times|\mathrm{v}|}, \mathrm{v} \in \mathbb{I}^{\mathrm{lv}}$$

矩阵中的每一列Z表示要嵌入的节点，其中的总行数等于嵌入的尺寸/大小。v是指示向量指示节点 v，除了节点所在列为1，其余节点都为0。我们看到，每个节点都以“浅”编码形式映射为唯一的嵌入向量。生成节点嵌入的方法有很多（例如DeepWalk，node2vec，TransE），选择方法的关键取决于它们如何定义节点相似性。

Random Walk（随机游走）

现在让我们尝试定义节点相似性。在这里我们介绍**随机游走(Random Walk)**，这是一种定义节点相似性和训练节点嵌入的高效表达方式。（随机游走对节点相似性具有灵活的随机定义，该定义结合了本地和高阶邻域信息，并且在训练时不需要考虑所有节点对；只需要考虑随机游走时同时发生的配对）。给定一个图和一个起点，我们随机选择它的一个邻居，然后移动到该邻居；然后我们随机选择该点的一个邻居，然后移动到该点，依此类推。以这种方式选择的点的（随机）序列是图形上的随机游动。所以 **similarity(u, v)** 定义为 **u** 和 **v** 在网络上随机游走时同时发生的概率。可以按照以下步骤生成随机游走嵌入：

1. 估计从节点**u**使用随机行走策略 R 访问至节点 **v** 的概率。最简单的想法是行走固定长度，对每个节点开始无偏的随机游走（例如，DeepWalk from Perozzi et al., 2013）。
2. 优化嵌入方法，从而以对这些随机游走统计信息进行编码，因此嵌入之间的相似性（例如点积）会编码随机游走相似性。

Random walk optimization and Negative Sampling（随机游走优化和负采样）

由于我们想找到能够保留相似度的d维节点的嵌入，因此我们需要学习节点嵌入，以使附近的节点在网络中靠得很近。具体来说，我们可以定义通过某种策略 R 得到的附近的节点 $N_R(u)$ 作为节点 u 的邻居集合 。让我们回想一下从随机游走中学到的知识，我们可以使用某种策略 R 从图上的每个节点开始执行短的固定长度的随机游走去收集 $N_R(u)$ ，这是从初始节点进行随机游走策略得到的点的多集。注意 $N_R(u)$ 可以具有重复元素，因为可以在随机行走中多次访问节点。然后，我们可以优化嵌入，以最大程度地提高随机游走并发的可能性，我们将损失函数定义为：

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(\mathbf{P}\left(\mathbf{v}|\mathbf{z_u}\right))$$

其中使用softmax参数化 $-\log(\mathbf{P}\left(\mathbf{v}|\mathbf{z_u}\right))$ ：

$$P(v|\mathbf{z_u}) = \frac{\exp\left(\mathbf{z_u^\top z_v}\right)}{\exp\left(\sum_{\mathbf{n}\in\mathbf{V}} \mathbf{z_u^\top z_n}\right)}$$

合并后为：

$$\mathcal{L} = \sum_{u\in V} \sum_{v\in N_R(u)} -\log\left(\frac{\exp\left(\mathbf{z_u^\top z_v}\right)}{\exp\left(\sum_{\mathbf{n}\in\mathbf{V}} \mathbf{z_u^\top z_n}\right)}\right)$$

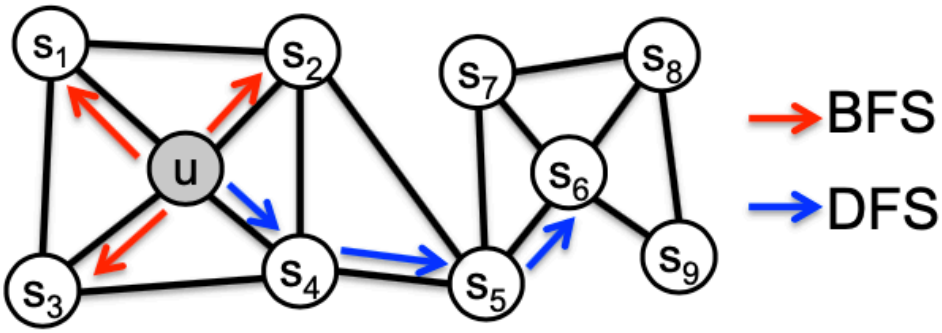
为了优化随机游走嵌入，我们需要找到嵌入 z_u 最小化 $\mathcal{L}$。但是，不做任何更改来优化会导致计算太复杂，节点上的嵌套总和复杂度为 $\mathrm{O}\left(|\mathrm{V}|^2\right)$。这里我们介绍负采样估算损失（要了解有关负采样的更多信息，请参阅Goldberg等人的 *word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method (2014)*）。从技术上讲，负采样是一个不同的目标，但是它是噪声对比估计（Noise Contrastive Estimation，NCE）的一种形式，它近似最大化softmax的对数概率。新的公式常常应用于逻辑回归函数（sigmoid）来区分目标节点 v 和从背景分布 P 采样的节点 n。

$$\log\left(\frac{\exp\left(\mathbf{z_u^\top z_v}\right)}{\exp\left(\sum_{\mathbf{n}\in\mathbf{V}} \mathbf{z_u^\top z_n}\right)}\right) \approx \log\left(\sigma\left(\mathbf{z_u^\top z_v}\right)\right) - \sum_{\mathbf{i}=1}^{\mathbf{k}} \log\left(\mathbf{z_u^\top z_{n_i}}\right), \mathbf{n_i} \sim \mathbf{P_v}$$

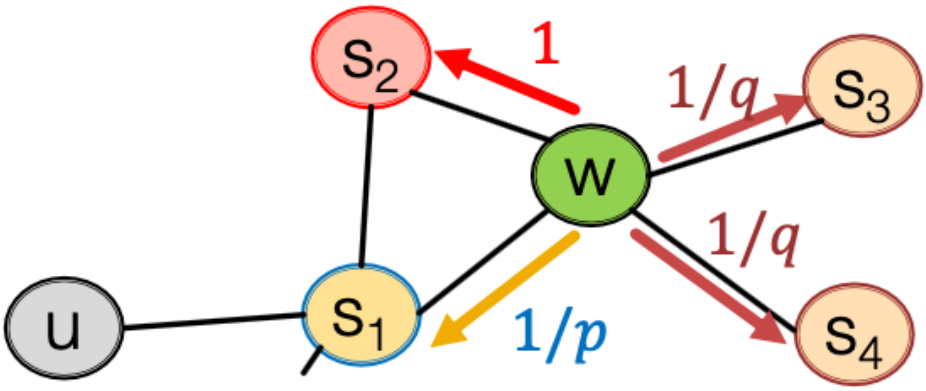
P_v 表示在所有节点上随机分布。我们只是针对 k 个随机的"负采样样本"标准化，而不是针对所有节点进行标准化。这样，我们需要采样 k 个与度成正比的负采样节点计算损失函数。注意 k 越大给出的估计越可靠，但它也有可能导致更高的偏差。实际上，我们选择 k 为5至20之间。

Node2vec

到目前为止，我们已经描述了在给定随机游走统计的情况下如何优化嵌入。但我们应该使用哪些策略来运行这些随机游走？如前所述，最简单的想法是从每个节点开始运行固定长度的无偏的随机游走（即*DeepWalk from Perozzi et al., 2013*），问题是这种相似性概念太受约束。我们观察到，节点 u 的网络邻域 $N_R(u)$ 的灵活概念导致了丰富的节点嵌入，Node2Vec的想法是使用灵活的，带偏差的随机游走，可以在网络的本地视图和全局视图之间进行权衡（*Grover and Leskovec，2016*）。使用两种经典策略 BFS 和 DFS 定义节点 u 的 $N_R(u)$：



BFS可以提供邻居的局部微观视图，而DFS可以提供邻居的全局宏观视图。在这里我们可以定义返回参数 p 和进出参数 q 并使用偏置 2^{nd}-order的随机游走探索网络邻居，对过渡概率建模以返回到先前的节点，并定义 q 为BFS和DFS的“比率”。具体来说，如下图所示，walker来自边 (s_1, w)，现在位于 w， 1, 1/q, 和1/p 表示访问下一个节点的概率（此处 w, 1, 1/q, 和1/p是非标准化概率）：



所以现在 $N_R(u)$ 是偏置行走访问过的节点。让我们将我们的发现汇总起来来说明node2vec算法：

1. 算随机游走概率
2. 对每个节点 u 开始模拟长度为 l 的 r 个随机游走
3. 使用随机梯度下降优化node2vec目标

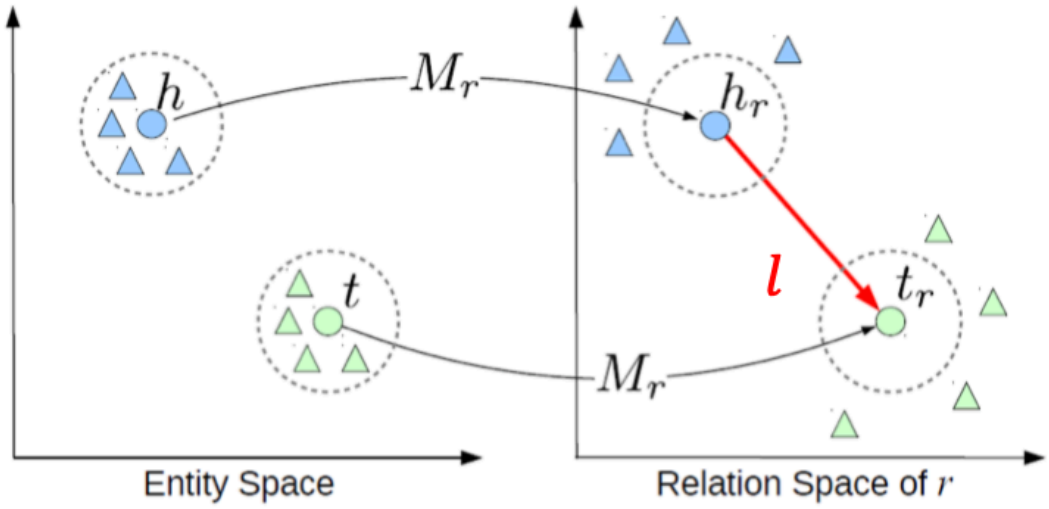
TransE

在这里，我们来看一下多关系图上的表示学习。多重关系图是具有多种类型的边的图，它们在诸如知识图之类的应用程序中非常有用，在知识图中，节点称为实体，边称为关系。例如，可能有一个节点表示 "*JKRowling*" ，另一个节点表示 *"Harry Potter"*，并且它们之间的边的类型为 *"is author of"*。为了为这种类型的图创建嵌入，我们需要捕获边的类型，因为不同的边表示不同的关系。

TransE(Bordes, Usunier, Garcia-Duran. NeurIPS2013)是一种特殊算法，旨在学习多关系图的节点嵌入。创建一个多关系图 $G=(E,S,L)$, G 由一组实体 E (即节点), 一组边 S ，以及一组可能的关系 L 组成。在TransE中，实体之间的关系表示为三元组：

$$(h,l,t)$$

其中 $h\in E$ 是头实体或源节点，$l\in L$是关系 $t\in E$ 是尾部实体或目标节点。与以前的方法类似，将实体嵌入到实体空间 $\mathbb{R}^k$ 中。TransE的主要创新之处在于每个关系 l 也作为向量 $l\in\mathbb{R}^k$ 的嵌入 。



也就是说，如果 $(h,l,s)\in S$, TransE尽力确保：

$$\mathbf{h}+\mathbf{l}\approx\mathbf{t}$$

同时，如果边缘 (h,l,t) 不存在，TransE尽力确保：

$$\mathbf{h}+\mathbf{l}\neq\mathbf{t}$$

TransE通过最小化以下损失来实现这一目标：

$$\mathcal{L}=\sum_{(h,l,t)\in S}\left(\sum_{(h',l,t')\in S'_{(h,l,s)}}\left[\gamma+d(\mathbf{h}+\mathbf{l},\mathbf{t})-d\left(\mathbf{h}'+\mathbf{l},\mathbf{t}'\right)\right]_+\right)$$

这里 (h',l,s') 是从集合 $S'_{(h,l,t)}$ 中 (h,l,t) 选择的“损坏的”三个元素，这些元素要么 h 要么 t（但不能同时使用）替换为随机实体：

$$S'_{(h,l,t)}=\{(h',l,t)\,|h'\in E\}\cup\{(h,l,t')\,|t'\in E\}$$

另外，$\gamma > 0$ 是一个叫做 margin 的标量，公式 $\mathbf{d(...)}$ 是欧几里得距离，并且 $[]_+$ 是正部分函数（定义为 $\mathbf{max(0, .)}$）。最后，为了确保嵌入的质量，TransE限制所有实体嵌入的长度为1，也就是说，对于每个 $e \in E$:

$$\|\mathbf{e}\|_2 = 1$$

下图为TransE算法的伪代码:

input Training set $S = \{(h, \ell, t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

initialize $\boldsymbol{\ell} \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$

  $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$

  $\mathbf{e} \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

loop

  $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$

  $S_{batch} \leftarrow$sample(S, b) // sample a minibatch of size b

  $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets

  for $(h, \ell, t) \in S_{batch}$ **do**

    $(h', \ell, t') \leftarrow$sample$(S'_{(h,\ell,t)})$ // sample a corrupted triplet

    $T_{batch} \leftarrow T_{batch} \cup \{((h, \ell, t), (h', \ell, t'))\}$

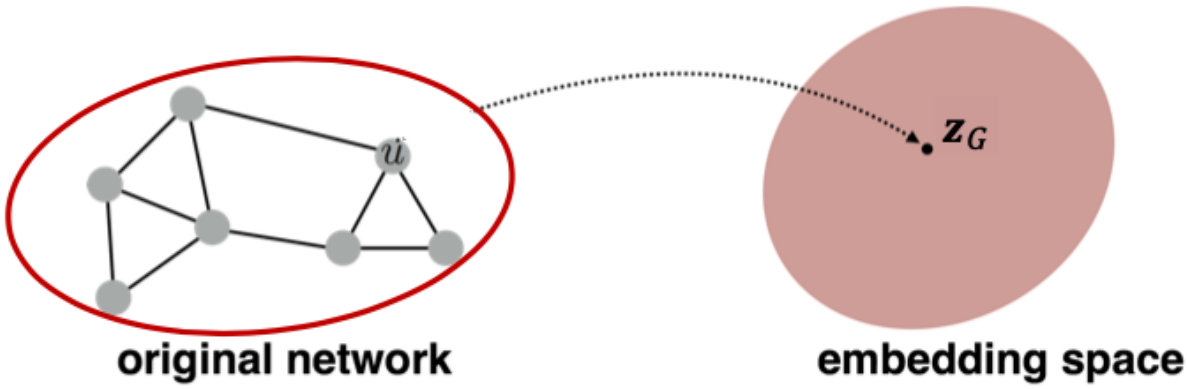
  end for

  Update embeddings w.r.t. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})]_+$

end loop

Graph Embedding（图嵌入）

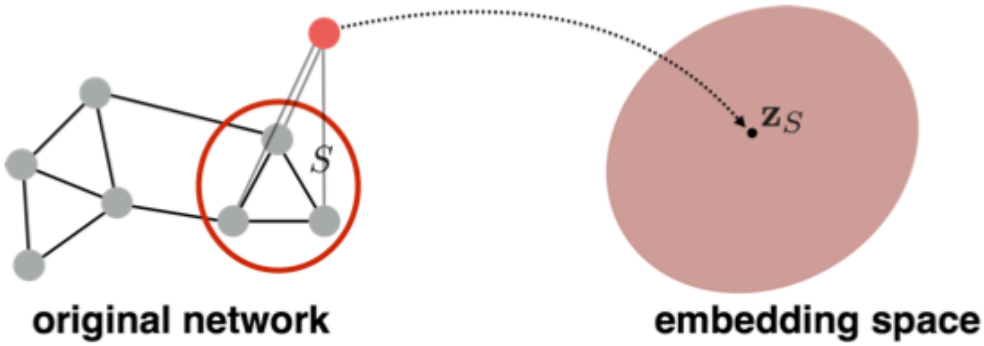
我们可能还想在某些应用中嵌入整个图 G（例如，对有毒分子与无毒分子进行分类，识别异常图）。



有几种想法可以完成图形嵌入:

1. 简单的想法 (*Duvenaud et al., 2016*) 是在（子）图 G 上运行标准的图形嵌入技术，然后对（子)图 G 中的节点嵌入求和（或取平均值）。

2. 引入“虚拟节点”来表示（子）图并运行标准的图形嵌入技术：



要了解更多关于利用虚拟节点进行子图嵌入的内容，请参阅(*Li et al., Gated Graph Sequence Neural Networks (2016)*)

3. 我们还可以使用匿名游走嵌入。为了学习图嵌入，我们可以列举 l 个步骤中所有可能的匿名游走 a_i 并记录其计数，并将图形表示为这些游走中的概率分布。要了解有关匿名步行嵌入的更多信息，请参阅(*Ivanov et al., Anonymous Walk Embeddings (2018)*)