本文基于 Network embedding 的几篇经典论文，对其方法和思路进行汇总。

一．Deepwalk

1.1 ramdom walk 和 pagerank

随机游走是一个在图上游走的动态过程，其中时间可以是连续或者离散的，游走的空间为 d 维。最早的随机游走问题追溯到醉汉问题，即一个醉汉从点 O 出发，随机选择一个方向走 l 距离，然后再随机选择一个方向走 l 距离，重复这个过程 n 次，问醉汉的位置离初始点 O 的距离在 τ 和 τ+σ 之间的概率值。

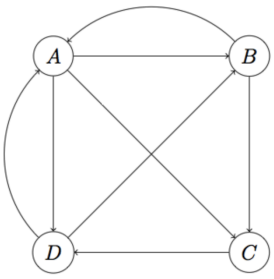
而现在随机游走问题常常表示为一系列独立同分布的 d 维随机变量 X 之和，Sn = X1+X2+…+Xn。Sn;n≥1 是整数时间的随机过程，称为随机游走。考虑最简单的随机游走过程，Xi 是一个服从伯努利分布的变量，可能的值为-1 和 1，其中取 1 的概率为 p，那么 Sn 为前 n 步游走的和，如果其中 m 个变量取 1，Sn = m –(n-m)，则：

$$Pr(S_n = 2m - n) = \frac{n!}{m!(n-m)!}p^m(1-p)^{n-m}$$

随机游走在推荐系统的应用中比较有名的一个方法是 pagerank，这是一种全局的 ranking 方法，即不考虑用户偏好，而利用 item 之间的网络结构关系对 item 的重要性进行排序。

Pagerank 的直观场景是网上冲浪，当冲浪者处于某个页面时，在页面提供的跳转链接中选择某个链接的概率取决于这个链接的重要性，同时有一个非常小的概率跳转到该页面没有指向的链接。所以 pagerank 随机游走过程可以看做是一个随机过程，在时刻 t 冲浪者处一个具体的点，然后在下一个时刻，冲浪者随机选择该点的出结点。转移矩阵确定之后，该随机过程会收敛到一个稳定的分布，则每个点的重要性可以被排序。

具体来说，对于有向图(V,e)，当冲浪者在 A 点时，他会以 1/deg(A) 向它的出结点跳转，这样我们可以得到转移矩阵，注意，矩阵的每一列的和都为 1。这样通过随机游走引入了一个马尔科夫链，我们用随机初始化每个页面的 rank 值 vi，也就是在任意时刻冲浪者处于 i 点的概率，比如设置为 1/N，则 Mv 就是新的 rank 值，迭代的用转移矩阵 M 乘以 v，v 最终会收敛



$$M = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 0 \end{bmatrix} \quad v = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

dead end：如果存在一个点出度为 0，那么转移矩阵竟无法满足每一列之和为 1，这种情况下我们可以为这些点赋予相同的转移概率，比如以 1/n 的概率连通到每个点。

非强连通图：马尔科夫链能够收敛的前提是其服从细致平稳分布。而当图 G 不是强连通的，则无法保证收敛，为此我们假设冲浪者在每个点都有极小的概率转移到图中任意点，这样网络图转换成了一个强连通图，则点 n 的 rank 值为：

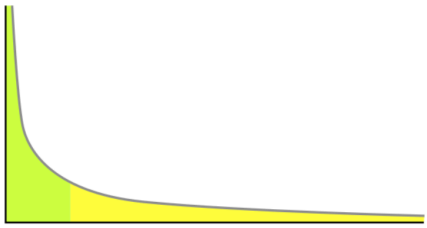
$$PR(n) = \alpha . \sum_{q:(q,n)\in\xi} \frac{PR(q)}{deg(q)} + (1-\alpha).\frac{1}{|V|}$$

1.2 deepwalk

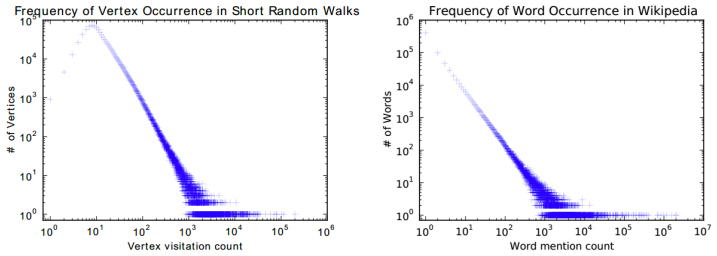
deepwalk 是一种通过一系列短的随机游走序列来学习社交网络各顶点的 embedding 结果的方法，也就是通过无监督的方法将图顶点映射到一个连续的低维的空间中，而不需要知道除了图结构以外的任何关于结点的信息。

文章使用随机游走序列作为 deepwalk 的输入，是因为：1. 随机游走序列可以得到网络的局部结构 2.可扩展性好，对于网络的变动只需要对变动的部分取一些随机游走序列而不需要重新训练 3. 可以被方便的并行化。得到这些序列后，deepwalk 利用了自然语言中的 skip-gram 模型。这是由于随机游走序列和自然语言存在的相似性：

幂律分布是一种广泛存在于生活中的分布，数学模型为 P(X≥x) = cx^-r，如下图所示，由于其形状也称为长尾分布。比较著名的例子就是 80\20 法则：百分之二十的人口占据了百分之八十的财富，统计发现，个人收入 X 不小于值 x 得概率与 x 的常数次幂成反比关系。



语言学家发现，如果将单词出现的频率按由大到小的顺序排列,则每个单词出现的频率与它的排名序号的常数次幂存在简单的反比关系：P(r) ～r^-α，这称为 zipf 定律。而通过实验观察到，在一系列短的随机游走中，图的顶点出现的频率也是服从幂律分布的。从而 deepwalk 考虑将自然语言中的 embedding 技术，迁移到网络中来对社交结构建模。



(a) YouTube Social Graph (b) Wikipedia Article Text

skip gram 的目标函数如下：

$$\underset{\Phi}{\text{minimize}} \quad -\log \Pr\left(\{v_{i-w},\cdots,v_{i+w}\} \setminus v_i \mid \Phi(v_i)\right)$$

优化这个函数使得具有相似邻居的顶点在特征空间相近。Deepwalk 在每个 epoch 中，首先将图的顶点顺序进行洗牌，然后以每一个顶点为起始点进行一次长度为 t 的游走，然后将这些游走序列作为 skip gram 算法的输入数据，同时用了分层的 softmax 提高计算速度：

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

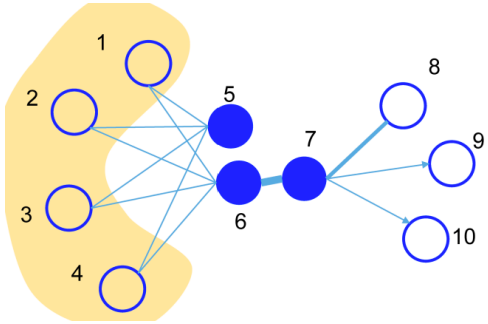
Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
  for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
    $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
    $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
  end for
end for

二．LINE: large-scale information network embedding

学习图节点表示时首先需要考虑了顶点之间是否有边连接，也就是图的局部特征（也称为一阶相似度）；同时一些没有边直接相连的点，也可能具有很高的相似度，比如共享很多朋友的两个点可能会相似，这称为二阶相似度，如下图的 56 点。Line 是一种适用于大规模的网络，使用于有向\无向、有\无权重的各种网络，可以捕捉网络的全局和局部特征的 embedding 方法。



line 首先将所有的网络都定义为 G=(V,E)，其中 E 是边 e 的集合，边 e=(u,v)是一个有序对，对于无向图 E 会包含(u,v)，(v,u)两个边，要求边的权重 w_{uv} 非负，看作是 uv 共同出现的次数（采样数据集时采样到(u,v)的概率会正比与 w_{uv}），这样将每个边都 binary 的（出现\不出现）。

为了使 embedding 后的向量可以描述顶点间的一阶相似度，定义了两个顶点的联合概率分布：

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^T \cdot \vec{u}_j)}$$

其中 ui 为顶点 vi 的低维向量表示，用两个向量的内积表示其相似度，再用 sigmoid 限制到 0 到 1 之间，表示 vi 和 vj 是朋友的概率，而 vi 和 vj 共现的经验概率正比于由边(i,j)的权重$\hat{p}_1(i,j)=\frac{w_{ij}}{W}$，其中 W 是图中所有边的权重和。希望这两个分布的 KL 散度最小，则目标函数为：

$$O_1=-\sum_{(i,j)\in E} w_{ij}\log p_1(v_i,v_j)$$

可以看到省略了一些常数项之后实际上是两个分布交叉熵；一阶相似度的训练只适用于无向图。

如果一阶相似度由两点的边的权重来描述，那么$p_u=(w_{u,1},\ldots,w_{u,|V|})$是 u 与所有点的一阶相似度，则 pu 和 pv 的相似度则描述了两点的二阶相似度。训练描述二阶相似度的表示时，每个顶点 vi 有一个向量表示 ui 和作为上下文的向量表示 ui'（**实际上等同于 skip-gram 中 softmax 前的矩阵**），对于有向边(i,j)，定义点 vj 是顶点 vi 的上下文的概率分布为：

$$p_2(v_j|v_i)=\frac{\exp(\vec{u}_j'^T\cdot\vec{u}_i)}{\sum_{k=1}^{|V|}\exp(\vec{u}_k'^T\cdot\vec{u}_i)}$$

可以看到先计算 V 中每个点与顶点 vi 的相似度，然后使用 sigmoid 函数。令上下文的条件概率分布 p2(*|vi)趋近于上下文的经验分布，目标函数为：

$$O_2=\sum_{i\in V}\lambda_i d(\hat{p}_2(\cdot|v_i),p_2(\cdot|v_i))$$

λi 表示顶点 vi 的重要性，此处使用 vi 的出度$d_i=\sum_{k\in N(i)}w_{ik}$，d()使用 kl 散度，经验分布正比于边(i,j)的权重$\hat{p}_2(v_j|v_i)=\frac{w_{ij}}{d_i}$，则目标函数为:

$$O_2=-\sum_{(i,j)\in E} w_{ij}\log p_2(v_j|v_i)$$

分析这个目标函数，我们知道最小化负似然对数就相当于最大化真实分布和建模分布的 kl 散度：kl(P(x)| P'(x|θ)) = ∫ P(x){−lnP'(x|θ) + lnP(x)}dx；由于真实分布 P(x)未知，要求 lnP'(x|θ)在 P(x)下的期望，我们用训练集（也就是从真实分布中采样的样本）{xn}对 lnP(xn|θ)求和来代替。单词\网络结点的预测任务实际上是一个多分类问题，我们用伯努利分布的拓展 1-of-K 分布来建模每次分类 p(*|vi)，由于是有监督学习，这个分布待确定的 K 个参数建模为关于输入 vi 的函数，该函数中的参数即为训练目标；上述为无权图中的结点预测，而本文的目标实际上就是一个加上了权重的多分类问题的最大似然，将有权图中每个边看做是一个 1-of-k 的多分类问题，这是由于我们并不能将图中存在的边看做是一个无偏的采样，但是由真实分布已知（假设其正比于权重），可以按该分布采样或者直接求 KL 散度。

这样可以学到每个点的上下文分布，有相似上下文的点的经验分布是相似的，所以用学到的表示求得的上下文分布也是相似的，也就是说该表示蕴含了二阶相似性；该训练过程适用于有向\无向图。Deepwalk 模型其实是一种以深度优先的方式学习二阶相似度的模型，对一个节点预测 V 个图节点哪些会是它的 w 度以

内的邻居（通过随机游走采样），而 line 学习二阶相似度时是广度优先的，对节点预测所有 V 个点哪些是它的（一度邻居）出节点。

Line 分开训练两种表示，然后将两者拼接起来，在具体的训练过程中，对边(i,j)优化 log p2(vj|vi)需要对图的所有顶点求内积，所以使用负采样的优化方法：

$$\log \sigma(\vec{u}_j'^{T} \cdot \vec{u}_i) + \sum_{i=1}^{K} E_{v_n \sim P_n(v)}[\log \sigma(-\vec{u}_n'^{T} \cdot \vec{u}_i)]$$

其中采样的负样本的概率取决于这些顶点的度：$P_n(v) \propto {d_v}^{3/4}$。在优化 O1 时，也需要进行负采样，这是为了避免出现平凡解，所有边的内积 ui*uj=∞，只需要将上式的 uj'换位 uj 即可。

优化中的另一个问题是边的权重对梯度下降带来的额影响，对边(i,j)计算梯度如下式：

$$\frac{\partial O_2}{\partial \vec{u}_i} = w_{ij} \cdot \frac{\partial \log p_2(v_j|v_i)}{\partial \vec{u}_i}$$

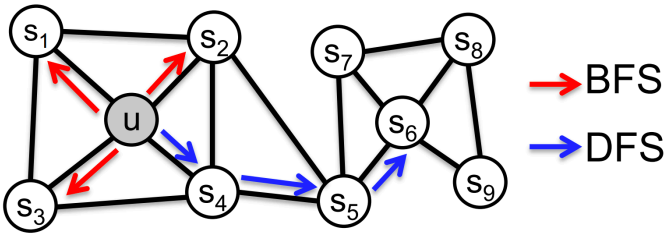
由于梯度都会乘一个边的权重，如果边的权重方差很大时，就会很难确定一个适当的学习率，使得大权重的边不会梯度爆炸同时使小权重的边梯度下降的速度不会太慢。使用边采样的方法可以解决这个问题，首先将所有的边都认为是 binary 的，然后根据边的权重对边进行采样训练。

比起 deepwalk，由于 line 只用了一度朋友建模，对于出度小的结点，很难有效训练出点的低维表示，所以 line 还尝试用宽度优先的方式利用二度朋友建模（即利用一个点的所有二度朋友，而不是 deepwalk 中深度优先的采样一个二度朋友。）定义用户 i 和其二度朋友 j 的权重为：

$$w_{ij} = \sum_{k \in N(i)} w_{ik} \frac{w_{kj}}{d_k}$$

三．Node2vec

Node2vec 认为，以上的方法受限于使用严格定义的邻居，而对网络中特有的一些连接模式不敏感。社交网络中的结点会组织为一些社区（聚类），社区中的点会有一些结构性质，比如在社区中作为中心点，node2vec 希望在同一个社区中的点有相似的表示，同时在社区中扮演角色相同的点的表示相似（如下图 u，s6）



node2vec 的学习框架与上述的方法相似，将网络特征学习座位是一个最大似然的优化问题。对于顶点 u，f(u)表示该点 d 维的特征表示，Ns(u)⊂V 表示通过本方法设计的邻居采样策略 S 得到的该点的网络邻居。要优化的目标为：

$$\max_f \sum_{u \in V} \log Pr(N_S(u)|f(u))$$

为了使上述目标易于计算，本方法提出了两个假设：条件独立性假设，即灭一个邻居点的在f(u)条件下的似然概率独立于其他邻居点；特征空间对称性假设，源结点和邻居点在特征空间中对彼此的影响是相同的，因此条件似然概率建模基于两个点的特征空间的内积

$$Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|f(u))$$

$$Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

则目标函数为：

$$\max_f \sum_{u \in V} \left[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \right]$$

其中$Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$，为了降低计算复杂度，可以通过负采样简化这个目标函数。

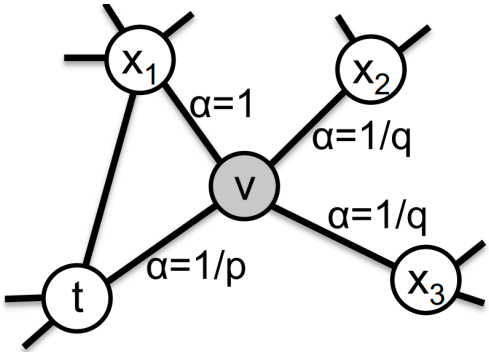
Node2vec 的贡献在于它对邻居的采样策略，在图中经典的搜索策略有 BFS（宽度优先采样）和 DFS（深度）。而在网络结点上预测任务主要依据结点两种层面的相似性：同质一致性假设在同一个聚类的点相似，DFS 可以探索大范围的网络从而易于推断社区；结构一致性假设具有相同角色的点相似，BFS 更容易通过观察到点的一二度邻居从而学到点的结构性质。Node2vec 实现了一个在 BFS 和 DFS 之间的插值的采样方法，基于随机游走：

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

Z 是标准化常数，$\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}$

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

d_{tx}代表顶点 t 和 x 之间的最短路径长度，可能的取值为 0、1、2。从而参数 p、q 分别控制着探索的速度和远离起始点的速度。



其中 p 是返回参数，控制游走的过程中返回上一个点的可能性，如果将其设置的较大(>max(q,1))，那么经过两步之后又采样到当前点的可能性较小，减少了 2-hop 冗余。若设置很小(<min(q,1))，则游走将一直起始点徘徊。Q 是 in-out 参

数，控制搜索向内或者向外的点，如果 q>1，随机游走会选择离 t 近的点，类似一个 BFS 的搜索，如果 q<1，游走过程会选择离 t 远的点，是一个类似 DFS 的搜索，但由于使用了随机游走框架，搜索的过程并不是严格的增加距起始点的距离。由于下一个状态取决于当前点和上一个点，本方法是一个二阶马尔科夫链。

为了减小时间复杂度，node2vec 使用了一种通过一次随机游走得到多个顶点的邻居的机制，假设需要为每个点采样 k 个邻居，那么进行长度为 l（l>k）的随机游走，就可以得到 l-k 个点的邻居，而不需要为这些点每个进行一次游走采样。

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
π = PreprocessModifiedWeights(G, p, q)
$G' = (V, E, \pi)$
Initialize $walks$ to Empty
for $iter = 1$ **to** r **do**
  for all nodes $u \in V$ **do**
    $walk$ = node2vecWalk(G', u, l)
    Append $walk$ to $walks$
f = StochasticGradientDescent(k, d, $walks$)
return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
Inititalize $walk$ to $[u]$
for $walk\_iter = 1$ **to** l **do**
  $curr = walk[-1]$
  V_{curr} = GetNeighbors($curr$, G')
  s = AliasSample(V_{curr}, π)
  Append s to $walk$
return $walk$

四．基于负采样的 Skip-gram 的本质：PMI 矩阵分解

4.1 相关概念

4.1.1 熵与互信息：

熵的本质是香农信息量 log(1/p)的期望。即一个事件发生的概率为 p，如果它发生了，携带的信息量为 log(1/p)，发生概率越小的事件，发生后提供越多信息。随机变量 X 真实分布为 p，那么该变量携带的平均香农信息量即为 X 的熵：

$$H(p)=\sum_i p(i) * log\frac{1}{p(i)}$$

如果用非真实分布 q 来编码来自分布 p 的随机变量，所需的平均编码长度（平均信息量）也就是两个分布的交叉熵：

$$H(p,q)=\sum_i p(i) * log\frac{1}{q(i)}$$

交叉熵比熵多出来的信息量就是相对熵（KL 散度）：

$$\begin{aligned}\mathrm{KL}(p\|q) &= -\int p(\mathbf{x})\ln q(\mathbf{x})\,\mathrm{d}\mathbf{x} - \left(-\int p(\mathbf{x})\ln p(\mathbf{x})\,\mathrm{d}\mathbf{x}\right) \\ &= -\int p(\mathbf{x})\ln\left\{\frac{q(\mathbf{x})}{p(\mathbf{x})}\right\}\mathrm{d}\mathbf{x}.\end{aligned}$$

以上是一个随机变量的情况，如果有一对随机变量 X,Y ~ p(X,Y)，那么描述这个变量对所需的平均信息量即为 X,Y 的联合熵：

$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(x,y)$$

对于这个变量对，如果我们知道随机变量 X 的分布，则描述 X 需要 H(X) 的信息量，那么要描述 Y 还需要多少额外信息呢，我们用条件熵来描述在已知 X 的条件下随机变量 Y 的不确定性：

$$\begin{aligned}\mathrm{H}(Y|X) &\equiv \sum_{x\in\mathcal{X}} p(x)\,\mathrm{H}(Y|X=x) \\ &= -\sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y|x)\,\log\,p(y|x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\,\log\,p(y|x)\end{aligned}$$

实际上就是 P(Y|X)的熵在 X 分布下的期望，联合熵和条件熵服从链式法则：H(X,Y) = H(X)+H(Y|X)

知道这些概念后我们给出互信息（Mutual Information）的定义，随机变量 X 和 Y 的互信息定义为：

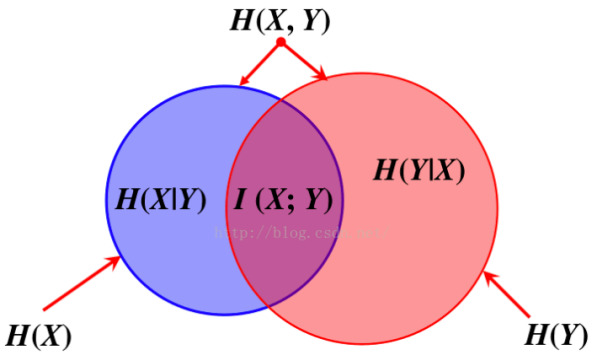
$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log\left(\frac{p(x,y)}{p(x)\,p(y)}\right)$$

互信息通过计算两个变量联合分布与边缘分布之积的 KL 散度来描述两个变量“不独立”的程度：

$\mathrm{I}(x,y) = \mathrm{KL}(p(x,y)\,||\,p(x)p(y))$

互信息是非负的、对称的，互信息和条件熵之间的关系为：

$\mathrm{I}(x,y) = \mathrm{H}(x) - \mathrm{H}(x\,|\,y) = \mathrm{H}(y) - \mathrm{H}(y\,|\,x)$

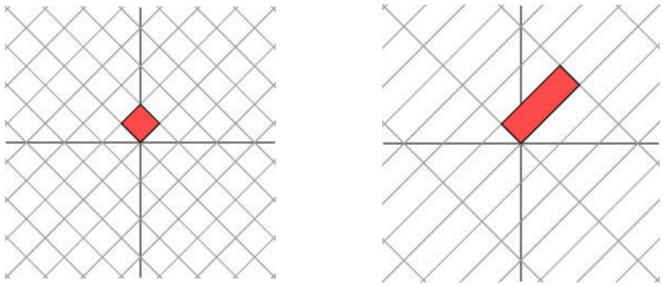


点互信息（PMI）是当 X 和 Y 有具体的取值时，这两个值的相关性判断，互信息是点互信息的期望值：

$$PMI(x;y) = log\frac{p(x,y)}{p(x)p(y)} = log\frac{p(x|y)}{p(x)} = log\frac{p(y|x)}{p(y)}$$

4.1.2 SVD 分解

首先看一下 EVD 分解的物理意义。对于 n 阶矩阵 A，如果数 λ 和向量 x 使得 Ax = λx，那么 x 是 A 的特征向量，也就是说在 x 的方向上施加初等变换 A，那么变换后不会改变方向，而是相当于将 x 放缩了特征值 λ 倍。如果 A 有 n 个线性无关且相互正交的特征向量，那么这些向量组成了 n 维空间中的一组正交基，这组基有一个性质：对其施加线性变换 A，基只改变大小（拉伸变化），不改变方向（不做旋转变化）。



然而不是所有的矩阵都存在 n 个线性无关的特征向量，我们取一种比较特殊的矩阵：对称矩阵，对称矩阵一定可以相似对角化，也就是说对于对称矩阵 A，一定能找到一个对角矩阵 Λ，使得 $A = P\Lambda P^{-1}$（相似）。

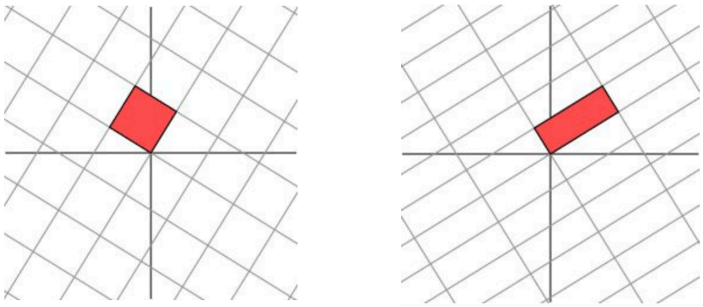
两个矩阵相似，意味着特征值相同，对角矩阵的特征值就是对角线上所有的值，而 P = [p1,p2…pn]实际上就是 A 的特征向量，与 Λ 对角线上每个值一一对应，这是因为：

A[p1,p2…pn] = [p1,p2…pn]Λ

则 A p1 = λ p1

所以如果 A 有 n 个线性无关的特征向量，那么矩阵 P 可逆，从而 $A = P\Lambda P^{-1}$。我们已经知道对称矩阵一定可以相似对角化，其实对于对称矩阵来说，P 不仅是可逆矩阵，还是正交矩阵（行和列都是正交的单位向量，$P^T = P^{-1}$），P 就是上图中的正交基，对应的特征值是该方向上拉伸的倍数。$A = P\Lambda P^{-1}$ 就是矩阵的 EVD 分解。

对于不对称的矩阵 M，我们不能确定一定能找到这样的正交基，使 M 作用在基上只有拉伸。但是我们可以找到一组正交基，经过 M 的作用有拉伸和旋转变换，但是变换后仍然是一组正交基（相互垂直）。



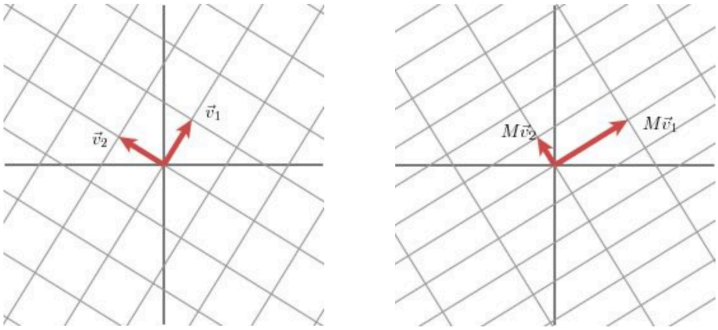
对于 m*n 阶的矩阵 M，我们要找到一组标准正交基 V（即 V 为正交矩阵），作用在 V 中每个向量上，得到的 Mv1，Mv2 仍是正交的，令新的方向上的标准正交基为 U，那么：

M[v1,v2…vn] = [u1,u2…um]Σ

即 M v1 = σ1 u1

Σ 矩阵是 m*n 阶的，主对角线上的值为奇异值，意味着 Mv1 在 u1 方向上的长度，其余的值都为 0。V 是一个 n*n 的矩阵，U 是 m*m 的矩阵。由于

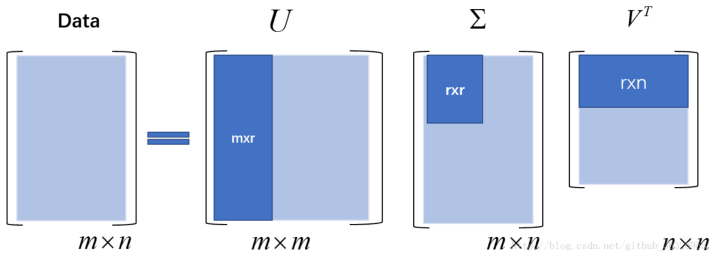
V 可逆，$M = U\Sigma V^{-1}$，这便是矩阵的奇异值分解。



奇异值中保存着矩阵的信息，如果我们将奇异值从大到小排列，那么前 r 个奇异值就保存了矩阵大部分的信息，这样：

$$Data_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n} \approx U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}$$

m*n 的大矩阵就可以用三个小矩阵相乘来近似描述，可以用于数据压缩，降维等。



4.2 基于负采样的 skip-gram 模型的本质

skip-gram with negative sampling（SGNS）的训练方法是一种经典的 word embedding 方法，通过将在数据集中观测到的单词-上下文对所对应的矢量内积最大化，随机选择的单词-上下文对的矢量内积最小化，得到了很好的单词表示，然而算法实际上优化的是什么量呢？本节将详细描述，SGNG 方法实际上是加权矩阵分解，优化目标是隐式分解平移后的 PMI 矩阵。

将来自同一个语料库单词表记为 Vw、上下文表记为 Vc，#(w,c)是(w,c)对出现次数，#(w)是单词 w 出现次数，#(c)是上下文出现次数。对于观测到的单个(w,c)对，目标函数为：

$$\log\sigma(\vec{w}\cdot\vec{c}) + k\cdot\mathbb{E}_{c_N\sim P_D}\left[\log\sigma(-\vec{w}\cdot\vec{c}_N)\right]$$

P_D 是一个与#(c)有关的分布，比如 $P_D(c) = \frac{\#(c)}{|D|}$，或者 $P_D(c) \propto \#(c)^{3/4}$。对语料库中的所有(w,c)对，全局目标函数为：

$$l = \sum_{w\in V_W}\sum_{c\in V_C}\#(w,c)\left\{\log\sigma(\vec{w}\cdot\vec{c}) + k\cdot\mathbb{E}_{c_N\sim P_D}\left[\log\sigma(-\vec{w}\cdot\vec{c}_N)\right]\right\}$$

SGNS 的目的是得到单词矩阵 W，上下文矩阵 C，由于目标函数中存在 w*c，这个算法也可以看做是对矩阵 M(Mij = wi*ci)进行了分解，下面我们来看 M 是什么矩阵。

考虑全局目标函数，当维度 d 足够大时，M 可以被完美重构，也就是说每个内积 w*c 都可以是与其他内积独立的取值，于是我们寻找 l 中只与某个内积 w*c

有关的那部分函数（l 则是与每一个内积有关的部分的加和），而优化目标是使 l 关于 w*c 最大化。

重写目标函数：

$$\ell = \sum_{w\in V_W}\sum_{c\in V_C} \#(w,c)\left(\log\sigma(\vec{w}\cdot\vec{c})\right) + \sum_{w\in V_W}\sum_{c\in V_C} \#(w,c)\left(k\cdot\mathbb{E}_{c_N\sim P_D}\left[\log\sigma(-\vec{w}\cdot\vec{c}_N)\right]\right)$$

$$= \sum_{w\in V_W}\sum_{c\in V_C} \#(w,c)\left(\log\sigma(\vec{w}\cdot\vec{c})\right) + \sum_{w\in V_W} \#(w)\left(k\cdot\mathbb{E}_{c_N\sim P_D}\left[\log\sigma(-\vec{w}\cdot\vec{c}_N)\right]\right)$$

展开期望的那一项：

$$\mathbb{E}_{c_N\sim P_D}\left[\log\sigma(-\vec{w}\cdot\vec{c}_N)\right] = \sum_{c_N\in V_C}\frac{\#(c_N)}{|D|}\log\sigma(-\vec{w}\cdot\vec{c}_N)$$

$$= \frac{\#(c)}{|D|}\log\sigma(-\vec{w}\cdot\vec{c}) + \sum_{c_N\in V_C\setminus\{c\}}\frac{\#(c_N)}{|D|}\log\sigma(-\vec{w}\cdot\vec{c}_N)$$

则对于特定的一个内积 w*c：

$$\ell(w,c) = \#(w,c)\log\sigma(\vec{w}\cdot\vec{c}) + k\cdot\#(w)\cdot\frac{\#(c)}{|D|}\log\sigma(-\vec{w}\cdot\vec{c})$$

记 w*c 为 x，令 l(x)对 x 求导值等于 0，得：

$$\vec{w}\cdot\vec{c} = \log\left(\frac{\#(w,c)\cdot|D|}{\#(w)\cdot\#(c)}\right) - \log k$$

而其中的第一项刚好是 w 和 c 的点互信息！则 SGNS 算法分解的矩阵 M 是单词与上下文的互信息矩阵（PMI）平移 log k 后的矩阵，当 k=1 时，平移的距离为 0：$M_{ij}^{\text{SGNS}} = W_i\cdot C_j = \vec{w}_i\cdot\vec{c}_j = PMI(w_i,c_j) - \log k$

以上是维度 d 足够高的情况，而对于较小的维度 d，无法用 W 和 C 矩阵完美重构 M，部分 w*c 会偏离理想值，在 l(w,c)中我们可以看出，#(w,c)和 k*#w*#c 越大，内积 w*c 的权重越大，所以 SGNS 问题可以看做是一个加权矩阵分解的问题。

在 NLP 中，使用 PMI 矩阵存在的问题是很多(w,c)对并没有观测到，#(w,c) = 0，导致对应的互信息是无穷，常用的一种解决方式是使用正定 PMI 矩阵（PPMI）：

$$PPMI(w,c) = \max\left(PMI\left(w,c\right),0\right)$$

去掉了无穷，同时也去掉了负关联，使矩阵变得更加稀疏（SGNS 不要求矩阵稀疏，而 SVD 分解是无法计算稠密的大矩阵的）。

4.3 谱方法（SVD）与随机梯度下降（SGNS）

最后，我们在考虑一下用 SVD 直接对 PMI（或 PPMI）矩阵进行分解的情况，如第一部分所述，取 Σ 前 d 个奇异值，那么矩阵$M_d = U_d\cdot\Sigma_d\cdot V_d^\top$是秩为 d 的矩阵里对原矩阵 M 最近似的：

$$M_d = \arg\min_{Rank(M')=d}\|M'-M\|_2$$

由于 $M^TM = V(\Sigma^TU^TU\Sigma)V^T$，可知 M 中两行的内积与 UΣ 矩阵两行的内积相等（由于 V 是正交矩阵），在这个意义上 W = UΣ 矩阵是 M_d 矩阵的低维代替。如果对 PMI 矩阵进行 SVD 分解，令 W 作为单词的 embedding，C = V 作为上下文的

embedding，那么我们就可以得到要的表示，但是在一些语义任务上的实验上，使用 SVD 的效果不如 SGNS。有人提出这是由于得到的单词和上下文表示是不对称的，提出了一种对称 SVD 分解的方法：

$$W^{\mathrm{SVD}_{1/2}} = U_d \cdot \sqrt{\Sigma_d} \qquad C^{\mathrm{SVD}_{1/2}} = V_d \cdot \sqrt{\Sigma_d}$$

该方法虽然没有理论上的依据，但是确实效果比 SVD 更好。