Graph Representative Learning

19307130195 陈乐偲

使用图表示学习的方式，进行结点分类和链接预测任务，采用Cora数据集

数据集如下，

数据集	结点数	边数	结点特征数	标记比例	结点种类数
Cora	2708	5429	1433	-	7

结点分类(Node Classification)

- ChebConv
- GCNConv
- GATConv
- SplineCNN
- SSP（KFAC+GCN)

实验（Results: Accuracy）

模型	Accuracy（复现/论文）
ChebConv	81.40 / 81.20
GCNConv	81.10 / 81.50
GATConv	81.19 / 83.00
SplineCNN	81.49 / 89.43
SSP	92.19 / 90.160

链接预测（Link Prediction)

- GAE(GCN+InnerProducter)
- VGAE(GCN+InnerProducter)
- ARVGAE
- S-VGAE
- W-ARVGAE(自己弄的并自己起的名字，模型很垃圾，这个结果是硬跑出来的)

实验（Results: AUC-AP)

模型	**AUC（ours/offical)**	**AP(ours-offical)**
GAE	91.14/84.3	91.28/88.1
VGAE	90.56/84.0	91.56/87.7
ARVGAE	92.5/92.4	92.9/93.2
S-VGAE	92.88/94.1	93.1/94.1
W-ARVGAE	91.3/-	92.9/--

Theory

相关理论证明

ChebConv

给定图G,定义拉普拉斯矩阵$L=D-W$, 其中W为边权，对角阵D表示节点的度数，$D_{ii}=\sum_j W_{ij}$，给定图的结点特征X,

有$E=tr(X^TLX)=\frac{1}{2}\sum_{i,j}w_{ij}||x_i-x_j||_2$,$E$可以视作衡量$G$的能量,同时也衡量了图的平滑度等，当相邻两个结点的特征差距$||x_i-x_j||_2$越大时，图的能量E越大，

当边无边权的时候，如Cora数据集，边权可由邻接矩阵定义，即$W=A$,

由于L为正定对称矩阵，L存在谱分解，$L=U^T\Lambda U$, $E=tr(X^TU^T\Lambda UX)$, 令$Y=UX$,则$E=tr(Y^T\Lambda Y)$,

$Y = UX$，将信号X转换为另一个空间的信号Y,且Y对于衡量信号X的变化，也即频率有很大的作用，如果将X所在的子空间视为空间域，Y所在的空间视为频率域，则变换算子U,起到了类似于傅里叶变换的效果，将该变换方式称为图傅里叶变换。

由于U为正交矩阵，$U^TU = I$, 则傅里叶变换U对应的反傅里叶变换为$U^{-1} = U^T$.

对于在U对应的像空间中的向量y, y可以表示为一组标准正交基$\{u_i\}$的线性组合$y = \sum_i x_i u_i$, 则$y^T \Lambda y = \sum_i \lambda_i x_i^2$，如果$U$选择部分特征向量$\{u_k\}$对应的子空间，则获得的$y$只得到了$\{u_k\}$对应的特征$x$的信号,比如选取$U = [u_1, 0, \dots, 0]$, 则$y = \lambda_1 x_1^2$, 如果将小特征值$\lambda_1$视作低频信号的表征，则上述例子中选取的$U$相当于对$x$进行了一次低通滤波。

由傅里叶变换的卷积定理，在空间域的卷积等价于在频率域的$Hadama$乘积$\odot$,用该性质定义图卷积，

$$U(f * g) := (Uf) \odot (Ug)$$

~~该公式应该是直接类比定义的，这里左式的卷积和傅里叶变换的卷积应该不是一个东西~~

用可学习的卷积核

$$g_\theta(\Lambda) = diag(g(\theta)) = g(diag(\theta))$$

对空间域的X做卷积得到同样在空间域的另一信号Y。

上述卷积也等价于使用$Hadama$乘积对X对应的频率域信号$\tilde{X}$与卷积核进行逐点相乘得到同样在频率域的信号$\tilde{Y}$,用矩阵表示如下，

$$\begin{aligned} g_\theta(\Lambda)X &= \Theta \odot X \\ \tilde{Y} &= UY \\ \tilde{X} &= UX \end{aligned}$$

代入得，

$$\begin{aligned} \tilde{Y} &= g_\theta(\Lambda)\tilde{X} \\ UY &= g_\theta(\Lambda)UX \\ Y &= U^T g_\theta(\Lambda)UX \end{aligned}$$

其中卷积核的参数θ 为可学习参数，而该多项式本身有应该和图的特征值Λ 相关，才能得到图的频率特征等。比如，令$g_\theta(\Lambda) = \Lambda^{\frac{1}{2}}$,则

$$\begin{aligned} Y &= U^T \Lambda^{\frac{1}{2}} UX = L^{\frac{1}{2}} X \\ E &= tr(Y^T Y) = tr(X^T L X) \end{aligned}$$

此时的Y可以认为储存了能量信息，

用多项式表示卷积核，也即

$$g_\theta(\Lambda) = \sum \theta_k \Lambda^k$$

则结点分类任务可以转化为选取特定的θ ,使得图上的结点特征（原信号），经过上述图卷积运算后得到标签特征（目标信号），而θ 可以通过梯度下降算法迭代计算得到。同时，在实际中，可以使用多次图卷积操作堆叠得到最终的信号，而不一定只能进行一次图卷积。

但计算该多项式的复杂度很高，如果使用切比雪夫多项式定义,

$$g_\theta(\Lambda) = \sum_k \theta_k T_k(\tilde{\Lambda})$$
$$\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I$$

其中T_k为k阶切比雪夫多项式，上述使用$\tilde{\Lambda}$ 定义的原因是使得满足切比雪夫多项式的定义域$[-1, 1]$,

由于T_k可以递归计算，

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

所以用切比雪夫多项式定义的卷积核可以大大加速计算效率，

~~这里不知道使用切比雪夫多项式和其最佳逼近性质等有没有关系，论文也没讲~~

由此，定义了$ChebConv$ ， 本质我理解为一种可学习的图信号处理的手段。

GCN

利用$ChebConv$中的结论，我们来回顾一下，

首先归一化拉普拉斯矩阵L ,并不影响结果，

$$L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

又由$Lx = \Lambda x$,

$$\tilde{Y} = g_\theta(\tilde{\Lambda})UX = g_\theta(\tilde{L})X$$
$$\tilde{L} = 2L/\lambda_{max} - I$$

由于L的最大特征值不超过2，该结论可由随机矩阵的性质，或用反证法等推出，此处从略。

假设$\lambda_{max} \approx 2$,代入得

$$\tilde{L} = L - I = -D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

取$k = 2$,

$$g_\theta(\tilde{L})X = \sum_k \theta_k T_k(\tilde{L}) = \theta_0 I - \theta_1 D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

假设$\theta_0 = -\theta_1 = \theta$,则

$$g_\theta(\tilde{L})X = \theta(I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X$$

为了计算方便，改动该公式，相当于对邻接矩阵先添加自环I后再用度数D进行归一化操作，

改动的原因是，如果使用

$$\tilde{A} := I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

则该矩阵的特征值范围为$[0, 2]$, 如果多次迭代进行会导致数值不稳定的情况发生，若改为，

$$\tilde{A} := D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$$

则每次归一化保证矩阵的特征值属于$[0,1]$,避免了上述情况，

$$g_\theta(\tilde{L})X = \tilde{A}X\Theta$$
$$\tilde{A} := D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$$

使用两层GCN的话，为了增加神经网络拟合函数的非线性性，在每一层之间加入激活函数Relu，而最后一层采用Softmax函数使得输出为结点属于每一个类别的概率，得到最终的公式，

$$Z = f(X,\Theta) = Softmax(\tilde{A}\ Relu(\tilde{A}XW_0)\ W_1)$$

其实从ChebConv到GCN的推导中做了很多近似，但GCN的效果很好，个人理解是利用了GCN的强大的拟合能力去减小这些近似的影响，或者可以认为神经网络可以自动地学习到这些近似成立的条件。

GIN

GIN，图同构神经网络，致力于解决使用GNN对图进行分类的问题，我们知道采用GCN等GNN可以得到每个结点的嵌入向量表示（node embedding), 如果我们将一张图的所有节点的向量表示拼接起来，便可以得到整张图的向量表示（graph embedding)

~~文章看起来很高端，实际上似乎并没有那么难~~

首先文章说，WL测试是GNN判断图同构的上界，直观来讲，WL测试每次将一个多重集（一个结点的邻居集合）用一个哈希函数做了一个单射，而GNN聚合邻居信息的时候，虽然使用了各种各样的函数，但显然并不是单射，比如常用的$Max(x)Sum(x),Mean(x)$这些函数都不是单射，那么当然不可能有WL测试强了。文中的严谨证明用了反证法，证明了如果WL测试停止时不能判断出两个图是否同构，那么GNN也不能。

然后文章又给出了几个传统GNN失败的例子，比如使用mean的时候对相同分布就会有相同的结果啦，使用max的时候如果有多个最大值那当然也只剩下一个啦，使用sum的时候也有两个集合加起来相等的时候啦，就算使用了非线性激活函数，但如果假设结点特征都为正数的话，常用的Relu函数可能还是将结点特征变为正数，那实际上非线性作用并没有体现出来，那还是会出现什么什么 max，sum，mean失败的情形啦。

主要比较屌的地方是文章提出了一种新的图卷积方式，而且证明了该方式可以达到WL测试的上界，

那么首先我们知道，GNN中聚集一个结点的邻居信息，可以定义为一个多重集上的函数，那么要达到WL测试的上界，就要求这个函数是单射，多重集简单来说就是集合中元素可以重复的集合啦。

如果我们假设多重集X是可列集，而且集合的元素有限，那么显然存在函数h,

$$h(X) = \sum_{x\in X} f(x)$$

使得h这个单射可以表示为一组单射函数f的和，直观来讲，只要开一个n 位数组，n为集合元素个数的上界，而每一位的值域也是n, 存储每一个元素的出现次数，然后我们以看，这样的数组不就是一个n位n进制数嘛，那确实它可以写成上面的表示形式，$h(X)$表示这个n进制数的数值，而$f(x)$ 表示每一位代表的数值。

文中用公式定义了这个单射函数，但看来看去就是这个东西啦，不过写成公式就会显得比较高级啦。

那么这样子，我们就可以区分出每个结点的不同邻居结构了，但我们目的是区分每个结点的结点中心图$ego-graph$, 也就是我们还要考虑这个结点本身，那我们又需要一个单射函数，

$$h(c,X)=(1+\epsilon)f(c)+\sum_{x\in X}f(x)$$

其中,c表示结点c的特征，X表示该结点的邻居的特征的集合，ϵ 为任意的无理数，但x,c都为有理数，f也为有理函数

我们来证明这样的定义是一个单射，

$$(X,c)\neq(X',c')\Rightarrow h(c,X)\neq h(c',X')$$

若$X=X'\ or\ c=c'$ 显然，由f为单射，左式一定可以推出右式，

我们考虑最难的情况，也即,$X\neq X',c\neq c'$,

用反证法，假设$h(c,X)=h(c',X')$,

$$\epsilon(f(c)-f(c')=f(c)-f(c')+\sum f(x)-\sum f(x')$$

由于左端为无理数，右端为有理数，显然矛盾，故得证，

那么到了这里，还有一个问题，GNN的可学习参数放哪里呢，那再定义一个单射函数ϕ ,单射函数嵌套单射函数显然也是单射的嘛，

$$h(c,X)=\phi((1+\epsilon)f(c)+\sum_{x\in X}f(x))$$

利用MLP强大的拟合能力学习ϕ函数和f函数，并且令$MLP=f^{(l)}\circ g^{(l-1)}$，可以得到GIN最终的公式

DropEdge

本文首先阐述了GCN过平滑化的缺点，并且给出理论证明，然后证明了DropEdge的手段，也即每两层GCN之间随机Drop掉一些边可以减缓GCN的过平滑化的缺点。

所以我们得先证明一下，为什么GCN会有过平滑化的缺陷。

首先回顾GCN的定义，单层GCN的计算公式，

$$f(X)=\sigma(PXW)$$

其中，P表示和邻接矩阵相关的线性变换，W表示一个可学习的线性变换，σ为激活函数，

对比GCN的公式，

$$f(X)=Relu(\tilde{A}X\Theta)$$

也即$\Theta=W,\tilde{A}=P,\sigma(x)=Relu(x)=max(x,0)$,

论文中定义了P为$L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$的多项式函数，可以注意到，没有经过自环trick之前的GCN公式，可以表示为，

$$\tilde{A}XW = (2I - L)XW = g(L)XW$$
$$\tilde{A} := I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = 2I - L$$

也即原GCN的公式也满足该定义，下面使用该公式作为GCN的公式分析，

首先注意到，U为L的零空间，同时也是其最小特征值对应的特征子空间,因为

$$Le = (I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}})e = 0$$

其中$e = [1, 1, \dots, 1]^T$

此时满足U为线性变换P的不变子空间

$$\forall x \in U, Lx = 0$$

可以直接验证下式成立，

$$Lx = 0 \Rightarrow LPx = 0$$

同时，由于L为对称阵，P也为对称阵，则U的正交补空间U^C 也为P的不变子空间，

$$\forall u \in U, v \in U^C$$
$$< Pv, u >=< v, P^T u >=< v, Pu >= 0$$

因为$Pu \in U$,

定义子空间X相对子空间M之间的距离

$$d_M(X) := inf\{||X - Y||_F \mid Y \in M\}$$
$$M := U \otimes R^C = \{\sum_m^M e_m \otimes w_m | w_m \in R^C\}$$

其中R^C表示C维实数集，其中假设U属于M维子空间R^M，而$\{e_m\}$为R^N 空间中的标准正交基， 而其中的前M个为U 的标准正交基，

下面需要证明，GCN可以使得$d_M(X)$逐层递减，

证明分为两步，第一步为消息传递和聚合的过程使得距离递减，

$$d_M(PXW) \leq d_M(X)$$

第二步为证明激活函数使得距离也递减，

$$d_M(\sigma(X)) \leq d_M(X)$$

综合两式就可以得到，每一层GCN使得$d_M(X)$递减，

$$d_M(f(X)) \leq d_M(X)$$

由于$X \in R^N \times R^C$, X可以表示为

$$X = \sum_{m}^{N} e_m \otimes w_m, \exists w_m$$

由Krocker积的性质等可以推出，该结论也可以直观理解，

$$d_M(X) = \sum_{m=M+1}^{N} ||w_m||_2$$

推导如下，首先有，

$$d_M(X) = || \sum_{m=M+1}^{N} e_m \otimes w_m||_F$$

对该式进行化简，

$$\begin{aligned} d_M(X) &= \sum_{m=M+1}^{N} (e_m \otimes w_m)^T (e_m \otimes w_m) \\ &= \sum_{m=M+1}^{N} (e_m^T e_m) \otimes (w_m^T w_m) \\ &= \sum_{m=M+1}^{N} (w_m \otimes w_m) \\ &= \sum_{m=M+1}^{N} ||w_m||_2 \end{aligned}$$

同理，

$$\begin{aligned} d_M(PXW) &= \sum_{m=M+1}^{N} (Pe_m) \otimes (W^T w_m) \\ &= \sum_{m=M+1}^{N} (\lambda_m e_m) \otimes (W^T w_m) \\ &= \sum_{m=M+1}^{N} e_m \otimes (\lambda_m W^T w_m) \\ &= \sum_{m=M+1}^{N} ||\lambda_m W^T w_m||_2 \\ &\leq \lambda s \sum_{m=M+1}^{N} ||w_m||_2 \\ &\leq d_M(X) \end{aligned}$$

其中，定义(λ_m, e_m) 为 P的一个特征对，而λ, s 分别为P的最大特征值和W的最大奇异值，

在$\lambda s \leq 1$的假设下，得到最后一个不等式，从而证明了消息传递的过程使得距离递减的性质，

~~文中好像并没有详细说明为什么$\lambda s \leq 1$的假设是OK的，相反文中证明了在这个假设下收GCN的渐进收敛性质的，个人感觉这是显然的结论，感觉这么证没有说明问题~~

下面需要证明激活函数也使得$d_M(X)$递减，

个人觉得这个结论是直观的，因为$Relu$激活函数只保留了正部，所以$d_M(X)$不可能会增大，当然文中的证明非常严谨，此处从略，大致证明思路也是从$Relu$函数入手，具体证明的过程中用到了e_m正交性质以及$||.||_F$ 的排列不变性质，此处从略。

有了以上的结论，可以定义$\epsilon - soomth$的定义，其实就是说，GCN堆叠到了一定的层数之后，$d_M(f^{(l)}(X)) \to 0$

用分析的语言表达为，

$$\exists l^*, \forall l \geq l^*, d_M(f^{(l)}(X)) \leq \epsilon$$

若要求

$$d_M(f^{(l)}(X)) \leq (s\lambda)^l d_M(X) \leq \epsilon$$

则只需要，

$$l \geq \frac{\log \frac{\epsilon}{d_M(X)}}{\log s\lambda}$$

也即当GCN的层数达到一定数目后，就会陷入过平滑化的危险之中。

搞了这么久，好像还没有进入DropEdge，DropEdge为什么有效呢，原因是DropEdge可以减缓过平滑化的趋势，那么又是如何减缓的呢，可以证明使用DropEdge的技术，可以提高上面推出的l的下界，也就是说GCN需要更多的层数才会达到同样的平滑化效果。

但在证明DropEdge的神奇作用之前，还得先证明一些也很神奇的公式，

好像又有点说来话长了，首先要从图上的随机游走说起.....

首先我们需要引入一些些概念....

图上的随机游走定义为每个结点随机地选择一个邻居转移，也即转移矩阵由如下公式表示

$$M = D^{-1}A$$

且有

$$Me = e, e_i = 1,$$
$$M^T\pi = \pi, \pi_i = \frac{d_i}{2m}$$

上述得到的两个向量即为转移矩阵M的左右特征向量，直接代入验算可得。

但M不是实对称矩阵，使用起来不方便，使用

$$N := D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = D^{\frac{1}{2}}MD^{-\frac{1}{2}}$$

则N存在正交谱分解，

$$N = \sum_i \lambda_i v_i v_i^T$$
$$v_1 = (\frac{D}{2m})^{\frac{1}{2}}e$$

代入得，

$$M = D^{-\frac{1}{2}}ND^{\frac{1}{2}} = Q + \sum_{i=2}^{n} D^{-\frac{1}{2}}\lambda_i v_i v_i^T D^{\frac{1}{2}}$$

$$M^t = D^{-\frac{1}{2}}N^t D^{\frac{1}{2}} = Q + \sum_{i=2}^{n} D^{-\frac{1}{2}}\lambda_i^t v_i v_i^T D^{\frac{1}{2}}$$

其中Q为π组成的矩阵，由于$\forall i > 1, \lambda_i < 1$，右式的第二项收敛至0，

$$M \to Q, t \to \infty$$

下面定义结点之间的平均距离，$H(i,j)$定义为结点i走到结点j的期望时间，由于结点i到结点j的期望时间等于其邻居结点到j的平均时间加1，用矩阵表示为，其中J为全1矩阵，当$i \neq j$ 时有下式成立，

$$H = J + MH$$

所以已知

$$F = J + MH - H$$

为对角阵，下面求其对角元素，利用下式，

$$F^T\pi = J^T\pi + H^T(M^T - I)\pi = J\pi = e$$

则计算得到，

$$F_{ii} = \frac{2m}{d_i}$$
$$F = 2mD^{-1}$$
$$(M - I)H = 2mD^{-1} - J$$

但由于根据Perron定理，M的最大特征值为1，因此$M - I$不可逆，无法通过直接求逆的方式求得H,但注意到，

$$(M - I)e = 0$$

因此，如果H为该方程的解，则$\forall a, H + ea^T$ 也为该方程的解, 因此可以通过选取合适的a，使得该方程可解，

取$a = \pi$, 则代入后可以求出H，此时$ea^T = Q$,

~~但此处计算有点麻烦，我并没有算出来~~

这里采用另一种方式的证明，

$$N = Q^T\Lambda Q$$
$$L = D - A$$
$$L^+ = D^{-\frac{1}{2}}Q(I - \Lambda)^+ D^{-\frac{1}{2}}$$

根据上式，

$$(I - M)H = J - 2mD^{-1}$$
$$LH = DJ - 2mI$$

又由，

$$L^+L = I - \frac{1}{n}ee^T, e = [1, 1, \ldots 1,]^T$$

两边同乘L^+并移项，

$$H = L^+J - 2mL^+ + eu^T, \exists u$$

展开每个元素得，

$$H_{ij} = \sum_k L_{ik}^+ d_k - 2mL_{ij}^+ + u_j$$

我们知道$H_{ii}=0$,因此代入可以求出u_j,

$$u_j=\sum_k L_{jk}^+ d_k+2mL_{jj}^+$$

定义图中两个结点的通讯距离,

$$\kappa(i,j)=H(i,j)+H(j,i)=2m(L_{ii}^+ +L_{jj}^+ -2L_{ij}^+)$$

再代入之间得到的,

$$L^+ =D^{-\frac{1}{2}}Q(I-\Lambda)^+ D^{-\frac{1}{2}}$$

可以得到最终的公式,其中v表示Q的列向量,

$$\kappa(i,j)=2m\sum_{k=2}\frac{1}{1-\lambda_k}(\frac{v_{ik}}{\sqrt{d_i}}-\frac{v_{jk}}{\sqrt{d_j}})^2$$

由$\lambda=\lambda_2\geq\lambda_k,\forall k\geq 2$,以及$Q$为正交矩阵,$v_{ik}^2+v_{jk}^2\leq 1$,

以及显然上式的最大值不超过$v_{ik}^2+v_{jk}^2=1$ 的情况,

但在$v_{ik}^2+v_{jk}^2=1$ 的情况下可以通过不等式放缩证明,

$$(\frac{v_{ik}}{\sqrt{d_i}}-\frac{v_{jk}}{\sqrt{d_j}})^2\leq\frac{1}{d_i}+\frac{1}{d_j}$$

故

$$m(\frac{1}{d_i}+\frac{1}{d_j})\leq\kappa(i,j)\leq\frac{2m}{1-\lambda}(\frac{1}{d_i}+\frac{1}{d_j})$$

~~好了终于证到了这里,下面还有更神奇的,但是两只羊打发我去睡觉了,所以我们明天见朋友们。~~

下面的证明,建立起了图上的随机游走和电路之间的关系,个人觉得非常美妙,先看如下公式,

$$\phi(v)=\frac{1}{d_v}\sum_{u\in\Gamma(v)}\phi(u)$$

其中,$\Gamma(v)$ 表示v的出度邻居集合,

考虑图上的两个结点s,t。

如果$\phi(u)$ 定义为一个从u出发的随机游走,在到达t之间经过了结点s的概率,显然这样定义的ϕ 满足上式,由于要经过一个结点必先到达其某个入度邻居。

如果将图视作一个电路,图中的每一条边视作一个单位电阻,考虑一个从结点s流向t的电流,定义$\phi(u)$ 为结点u上的电压,显然由伏安定律,ϕ也满足上式。

所以上述两个定义本质上是等价的,同时,注意到在上述两个定义中均满足$\phi(s)=1,\phi(t)=0$,

同时,由伏安定律,s与t之间大的电阻表示为,

$$R_{st}=\frac{1}{\sum_{u\in\Gamma(t)}\phi(u)}=\frac{1}{\phi(t)d_t}$$

此时$\phi(t)$的含义是,从t出发的随机游走,在返回t之前经过了s的概率。

在证明最终结论之前，需要再看一下图上随机游走的性质，代入易验证上述平稳分布实际上为细致平稳分布，

$$p_{ij}\pi_i = p_{ji}\pi_j = \frac{1}{2m}$$

该式子实际上说明了，到达平稳分布后，停留在每一条边都是等概率的。

那么，如果我们从一条边出发，回到该边的期望步数为$2m$,

同理，停留在每一个结点的概率等于π_i, 从该节点出发，重新回到该节点的期望步数为$\frac{1}{\pi_i} = \frac{2m}{d_i}$,

$$E(\sigma) := E(t \to s \to t) = H(t,s) + H(s,t) = \kappa(s,t)$$
$$E(\tau) := E(t \to t) = \frac{2m}{d_t}$$

其中，$E(s \to t)$ 定义为从s出发到达t的期望步数，又有

$$E(\sigma) - E(\tau) = (1-q)E(\sigma)$$

因为$P(\sigma = \tau) = q$, 而若以$1-q$的概率该事件没有发生，那么需要走的期望步数为，

$$(1-q)E(\sigma)$$

又因为该事件没有发生正好意味着$\tau < \sigma$, 那么此时已经走了$E(\tau)$步，只需要再走$E(\sigma) - E(\tau)$ 步就可以使得事件σ发生，也即期望步数为

$$E(\sigma) - E(\tau)$$

故

$$q = \phi(t) = \frac{E(\tau)}{E(\sigma)} = \frac{1}{\pi_t\ \kappa(s,t)}$$

则有，

$$R_{st} = 2m\kappa(s,t)$$

总结一下推出的几个公式，

$$\lambda \geq 1 - \frac{1}{\kappa(i,j)}(\frac{1}{d_i} + \frac{1}{d_j})$$
$$\kappa(i,j) = \frac{1}{2mR_{ij}}$$
$$l \geq \frac{\log \frac{\epsilon}{d_M(X)}}{\log s\lambda}$$

那么DropEdge的效果是什么呢?

我们从电路的角度看整张图，将边看作单位电阻，DropEdge等价于删去一个电阻（将其断路），电路中的总电阻值应该增大，

由，

$$\lambda \geq 1 - \frac{2m}{\kappa(i,j)}(\frac{1}{d_i} + \frac{1}{d_j}) = 1 - \frac{1}{R_{ij}}(\frac{1}{d_i} + \frac{1}{d_j})$$

我们知道这相当于增加了λ的下界，

而且，在极限情况下，不断执行DropEdge操作将会得到完全的断路，也即，

$$\exists s,t, R_{st} = \infty$$

此时有$\lambda \geq 1$但我们又已知$\lambda \leq 1$,故极限情况下一定会达到$\lambda = 1$, 故在未达到$\lambda = 1$之间λ的确会增加而不可能一直保持不变状态。

其实该结论等价于，增加了l特征值的重数。

那么，又由于，

$$l \geq \frac{\log \frac{\epsilon}{d_M(X)}}{\log s\lambda}$$
$$when\ \log \frac{\epsilon}{d_M(X)} < 0$$

l与λ成正相关关系，故l的值也会增加，

也即使用DropEdge的确可以减缓$\epsilon - smooth$的速度，

同时，从另一个角度，当图达到不连通状态时，空间M的维数至少增加1，那么$dim(M)$也增大了，相当于可以在更高维的空间中进行特征处理。从信息的角度上，允许我们在更高维的空间中考虑问题，可能可以获得更多的信息，从而有助于GNN的学习。

~~非常喜欢该文章，全文读来酣畅淋漓，在各种REF的帮助的前提下~~

GCN2

GCN的改版，公式长得有点像PageRank，然后证明了改进后有更好的收敛性质。

~~但发现证明中用了一些比较大的近似，并不十分认同，觉得整个公式也没有很好玩，这里就懒得写了~~

KFAC

KFAC(Kronecker-factoredApproximate Curvature) 是一种基于Kronecker-分解的二阶优化算法，一般神经网络的额优化不使用二阶优化算法，因为计算Hesson矩阵什么的实在太慢了，但KFAC利用一些近似实现了该算法。

首先，该优化算法定义在自然梯度上，自然梯度即为使用Fisher信息矩阵定义的梯度。

$$F = E[\nabla \log p(x) \nabla \log p(x)^T]$$

注意，为了表示方便，上式做了简化，其实表达的意思是，

$$\nabla \log p(x) := \nabla_\theta log(p|\theta)$$
$$E[p(x)] = E_{x \sim p(x|\theta)}$$

首先，我们知道，

$$
\begin{aligned}
E[\nabla \log p(x)] &= \int p(x) \frac{\nabla p(x)}{p(x)} \\
&= \int \nabla p(x) \\
&= \nabla \int p(x) \\
&= \nabla 1 \\
&= 0
\end{aligned}
$$

且当损失为负对数似然概率时，

$$
L = -\log p(x)
$$

对L 求二阶导，

$$
\begin{aligned}
H &= \nabla^2 L \\
&= -\nabla^2 \log p(x) \\
&= -\nabla \frac{\nabla p(x)}{p(x)} \\
&= \frac{\nabla p(x)}{p(x)}^T \frac{\nabla p(x)}{p(x)} - \frac{p(x)^2}{p(x)} \\
&= \nabla \log p(x)^T \nabla \log p(x) - \frac{\nabla^2 p(x)}{p(x)}
\end{aligned}
$$

对上式取期望，

$$
\begin{aligned}
E[H] &= E[\nabla \log p(x)^T \nabla \log p(x)] - E[\frac{\nabla^2 p(x)}{p(x)}] \\
&= F - \int p(x) \frac{\nabla^2 p(x)}{p(x)} \\
&= F - \int \nabla^2 p(x) \\
&= F - \nabla^2 \int p(x) \\
&= F - \nabla^2 1 \\
&= F
\end{aligned}
$$

回顾牛顿法的更新公式，

$$
x_{t+1} = x_t - H^{-1} \nabla L(x)
$$

那么，可以用F 代替H ,相当于用H的期望代替了H,该更新方式就是自然梯度下降。

其实，自然梯度也和KL散度在流形上的东东等相关，这里就不多说了。

首先，回顾F,

$$
F = E[\nabla \log p(x) \nabla \log p(x)^T] = E[\nabla L(x) \nabla L(x)^T]
$$

由于θ 本质为所有层W 的拼接，

$$
\begin{aligned}
d\theta &:= \nabla_\theta L(x) \\
F &= E[\nabla L(x) \nabla L(x)^T] = E[d\theta \, d\theta^T] \\
\theta &= [vec(W_0)^T, vec(W_1)^T, \dots vec(W_n)^T]^T
\end{aligned}
$$

代入展开得到，

$$
F_{ij} = E[vec(dW_i) vec(dW_j)^T]
$$

令a_i, g_i 分别为第i层的前向输入和反向传播梯度，由反向传播算法，其实就是链式求导法则，

$$dW_i = g_i a_i^T$$

代入，

$$vec(dW_i) = vec(g_i a_i^T) = a_i \otimes g_i$$

那么，

$$\begin{aligned} F_{ij} &= E[vec(dW_i)vec(dW_j)^T] \\ &= E[(a_i \otimes g_i)(a_j \otimes g_j)^T] \\ &= E[(a_i a_j^T) \otimes (g_i g_j^T)] \\ &\approx E[a_i a_j^T]E[g_i g_j^T] \end{aligned}$$

最后一个$\approx$为什么成立呢，他等价于说，

$$E[a^{(1)}a^{(2)}g^{(1)}g^{(2)}] \approx E[a^{(1)}a^{(2)}]E[g^{(1)}g^{(2)}]$$

其中$a^{(1)}$表示向量a的一个分量，

这个近似的误差其实可以用公式写出来，也即我们要计算下式，

$$E[a^{(1)}a^{(2)}g^{(1)}g^{(2)}] - E[a^{(1)}a^{(2)}]E[g^{(1)}g^{(2)}]$$

首先，由累积量和矩的关系，可以将$E[a^{(1)}a^{(2)}g^{(1)}g^{(2)}]$展开为一系列累积量之和，由于我们知道，

$$\begin{aligned} E[g^{(i)})] &= E[-\nabla_g \log p(x)] \\ &= -\int p(x)\frac{\nabla_g p(x)}{p(x)} \\ &= -\nabla_g \int p(x) \\ &= \nabla_g 1 \\ &= 0 \end{aligned}$$

那么，

$$\begin{aligned} Cov(a^{(i)}, g^{(j)}) &= E[(a^{(i)} - E[a^{(i)}])(g^{(i)} - E[g^{(i)}])] \\ &= E[(a^{(i)} - E[a^{(i)}])g^{(i)}] \\ &= E[a^{(i)}g^{(i)}] - E[a^{(i)}]E[g^{(i)}] \\ &= E[a^{(i)}g^{(i)}] \end{aligned}$$

又类似地，

$$\begin{aligned} E[a^{(i)}g^{(i)}]E[g^{(i)})] &= E[-a\nabla_g \log p(x)] \\ &= -\int a\, p(x)\frac{\nabla_g p(x)}{p(x)} \\ &= -\nabla_g \int a\, p(x) \\ &= \nabla_g a \\ &= 0 \end{aligned}$$

又由于一阶或二阶累积量和矩是等价的，代入可以消去很多项为0的累积量，为了简便，下面用下标代替，

$$\begin{aligned} E[a_1, a_2, g_1, g_2] =& \kappa(a_1, a_2, g_1, g_2) \\ &+\kappa(a_1)\kappa(a_2, g_1, g_2) + \kappa(a_2)\kappa(a_1, g_1, g_2) \\ &+\kappa(a_1, a_2)\kappa(g_1, g_2) + \kappa(a_1)\kappa(a_2)\kappa(g_1, g_2) \end{aligned}$$

又，

$$\begin{aligned}\kappa(a_1,a_2)\kappa(g_1,g_2)+\kappa(a_1)\kappa(a_2)\kappa(g_1,g_2) &= Cov(a_1,a_2)Cov(g_1,g_2)+E(a_1)E(a_2)Cov(g_1,g_2)) \\ &= (Cov(a_1,a_2)+E(a_1)E(a_2))Cov(g_1,g_2) \\ &= E(a_1,a_2)E(g_1,g_2)\end{aligned}$$

移向就得到了，

$$E[a_1,a_2,g_1,g_2]-E(a_1,a_2)E(g_1,g_2)=\kappa(a_1,a_2,g_1,g_2)+\kappa(a_1)\kappa(a_2,g_1,g_2)+\kappa(a_2)\kappa(a_1,g_1,g_2)$$

只要假设所有的右端的累积量都较小，那么我们的近似误差是较小的，

$$E[a^{(1)}a^{(2)}g^{(1)}g^{(2)}]\approx E[a^{(1)}a^{(2)}]E[g^{(1)}g^{(2)}]$$

那么，只需要不断地计算$E[a^{(1)}a^{(2)}]E[g^{(1)}g^{(2)}]$ 就可以估计出Fisher矩阵，下面解决F^{-1}不存在的情况，

其实也很简单，虽然F 不一定可逆，但F 作为协方差矩阵，是半正定的，加上一个小量就好啦嘿嘿。

$$F^{-1}\approx(F+\epsilon)^{-1}$$

把$KFAC$用在 GCN上面，效果的确很好，嘿嘿。