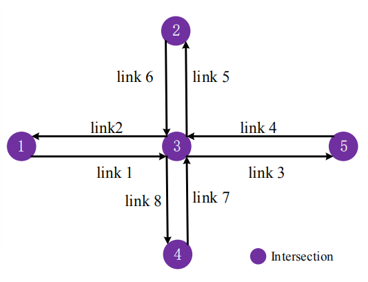
**Thinking1 GCN/Graph Embedding 都有哪些应用场景**

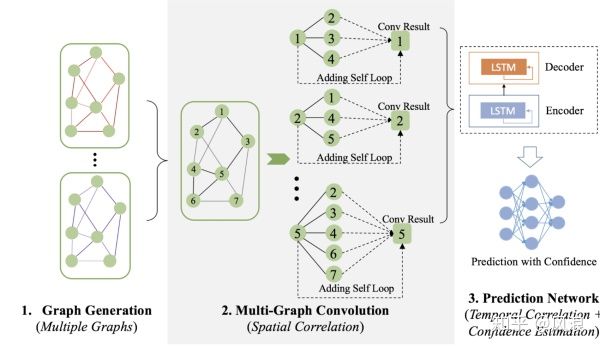
可以应用于NLP中的文本分类、商品推荐、 金融风控模型、聊天机器人语义分析及意图识别等等。

**Thinking2 在交通流量预测中，如何使用Graph Embedding，请说明简要的思路**



整个框架分成三个模块：

* 构建和融合Graph：定义路口间不同关系并融合得到Graph。
* GCN+encoder-decoder 模型：生成包含时间、空间、流量等信息的embedding
* MLP：结合embedding和节假日、天气等其他特征去预测路口流量。



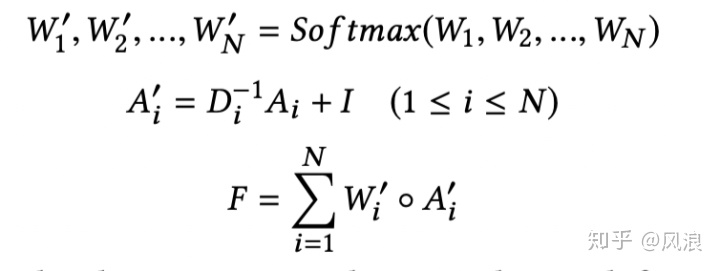
一. 构建和融合Graph

首先定义三种路口之间的关系（边权重）来表示路口间相似度，权重越大越相似

1. 路口间距离：考虑地理位置，距离越小越相似， 权重=路口间距离的倒数
2. 路口间行车记录：考虑交互，行车记录数越多越相似，权重=路口间记录数
3. 路口间流量相似度：考虑出入流量模式，权重=路口间出入流量的相关系数。

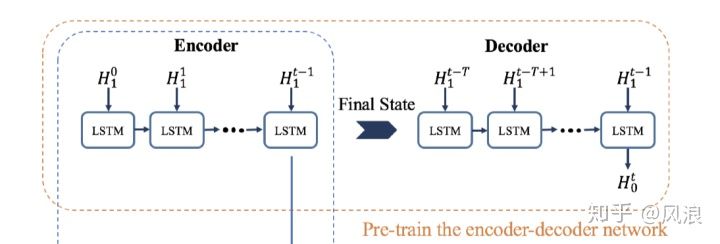
若有N种关系：

1. 定义N个参数矩阵，并归一化得到
2. 对每种关系对应的邻接矩阵进行GCN标准化处理：归一化、增加self-loop
3. N个 相乘求和合并权重，得到最终经过标准化、合并权重的graph表达F



》如果有N种关系, 路口间有|E|条边，则有个参数。因为本文的graph规模小，这种方式参数不会爆炸。如果规模较大，最好采用近似的方式，即每个边共享N个参数

二. GCN+Encoder-Decoder 模型

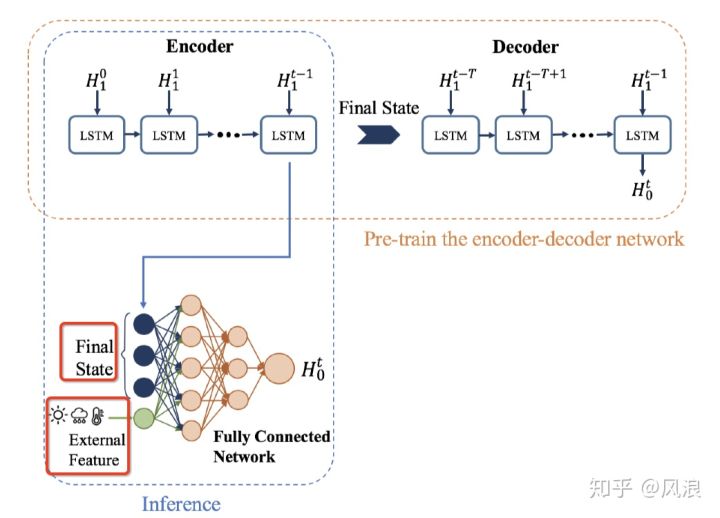


细节

1. 节点特征定义：时段每个路口流量  为流出量, 为流入量
2. 单层GCN操作：
3. 在Decoder层输入可能是为了提高离t较近时刻的特征权重。

三. MLP模型

以GCN+Encoder-Decoder 模型输出的final state向量 和人工设计的特征（如节假日、天气）做为MLP模型的输入特征，进行训练。



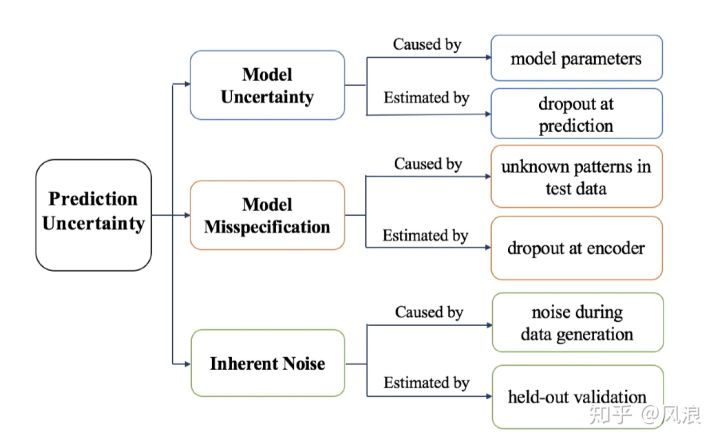
四. 估计置信区间

除了预测流量外，还额外输出一个置信区间，比较有实际意义。

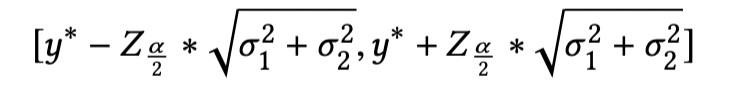
比如有两个路口：学校和小区，学校路口置信区间比较大，小区路口置信区间比较小（早出晚归）。我们可以把精力集中在学校这一类置信区间较大路口上；另后续提升算法的方向可以是深挖这类路口的模式；甚至可以用于异常检测，在置信区间以外的是异常的。

估计方法

首先定义了模型不确定性由以下三个因素决定



然后通过dropout方法估计前两个因素的方差, 通过划分出测试集估计得到第三个因素的方差, 最终得到预测结果  的置信区间如下



* 为标准正态分布下1-a点



**Thinking3 在文本分类中，如何使用Graph Embedding，请说明简要的思路**

01文本如何构建图

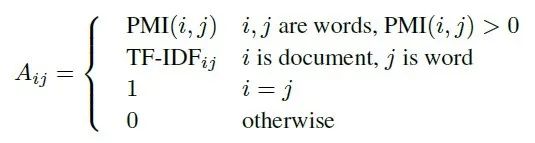
我们要构建一个具有定义好n个节点，m条边的图。

例如，我抽屉里有5本不同的机器学习书，里面一共有a个章节，同时所有书里面一共有b种不同的单词(不是单词个数，是所有的单词种类)。然后我们就可以给a个章节和b个单词标记唯一的id，一共n=a+b个节点，这是我们图的节点。

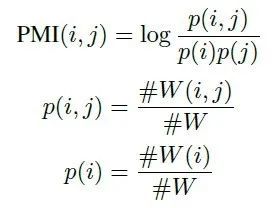
-边的创建-

我们有两种节点，章节和单词。然后边的构建则来源于章节-单词 关系和 单词-单词 关系。对于边章节-单词 来说，边的权重用的是单词在这个章节的TF-IDF算法，可以较好地表示这个单词和这个章节的关系。这个算法比直接用单词频率效果要好。单词-单词 关系的边的权重则依赖于单词的共现关系。我们可以用固定宽度的滑窗对5本书的内容进行平滑，类似于word2vector的训练取样本过程，以此计算两个单词的关系。具体的算法则有PMI算法实现。

point-wise mutual information（PMI）是一个很流行的计算两个单词关系的算法。我们可以用它来计算两个单词节点的权重。节点 i 和节点 j 的权重计算公式如下：



PMI(i, j)的计算方式如下：



#W(i) 表示所有的滑窗中包含单词节点 i 的个数。

#W(i; j) 表示所有的滑窗中同时包含单词节点 i 和单词节点 j 的个数。

#W 是总的滑窗次数

PMI值为正则说明两个单词语义高度相关，为负则说明相关性不高。因此最后的图构造过程中只保留了具有正值的单词节点对组成的边。

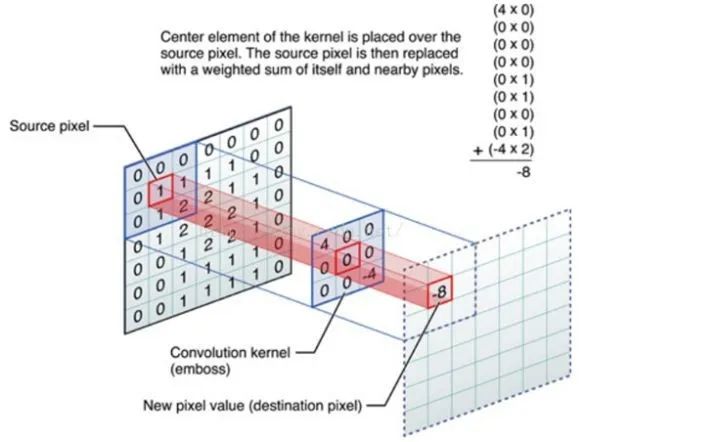
图的节点和边确定了，接下来就是用图卷积神经网络进行一些学习应用。

02 图卷积神经网络

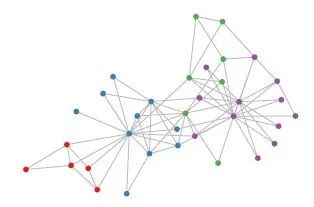
图卷积神经网络(Graph Convolutional Network, GCN)是一类采用图卷积的神经网络，在图网络领域的地位正如同卷积操作在图像处理里的地位。

什么是卷积

离散卷积的本质是一种加权求和。

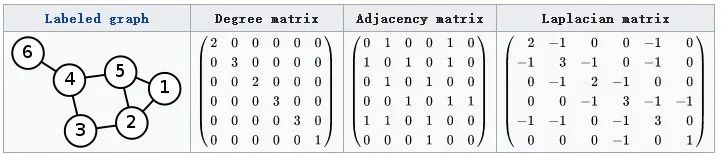
卷积过程示意图

CNN中卷积的本质就是利用共享参数的过滤器 kernel，通过计算中心像素点及相邻像素点的加权和来构成feature map实现空间特征的提取，加权系数就是卷积核的权重系数。卷积核的权重系数通过BP算法得到迭代优化。卷积核的参数正是通过优化才能实现特征提取的作用，GCN的理论很重要一点就是引入可以优化的卷积参数来实现对图结构数据特征的获取。

社交网络中图结构

图卷积的目的类似，寄希望学习到一种节点表示，该节点表示依赖于每个节点及其周边相邻的节点。然后该节点表示就可以输出用作分类任务了，就是我们常说的节点分类。

那么有什么东西来度量节点的邻居节点这个关系呢？拉普拉斯矩阵。举个简单的例子，对于下图中的左图而言，它的度矩阵 ，邻接矩阵 和拉普拉斯矩阵 分别如下图所示，度矩阵（degree matrix） 只有对角线上有值，为对应节点的度，其余为0；邻接矩阵只有在有边连接的两个节点之间为1，其余地方为0；拉普拉斯矩阵 为 。这是比较简单的拉普拉斯矩阵。

图结构数据的各种表示

以下是重点

图卷积网络(GCN)第一层的传播公式如下：



ρ是激活函数，比如ReLU。

上式的我们暂时理解等同于邻接矩阵A，代表图的拓扑结构，维度N\*N，N表示节点个数;

X是第一层输入的特征矩阵，维度N\*M，M表示每个节点的特征向量维度;

Wo是权重参数矩阵，维度M\*K，K代表转给下一层的向量维度。

因此第一层输出L1的向量维度就是 N\*K。

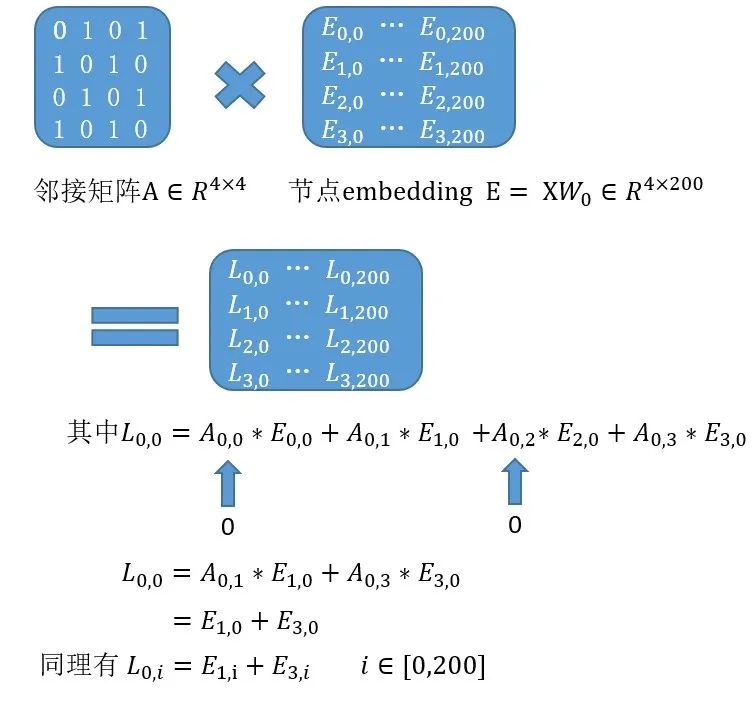
在上面介绍的文本分类任务中，

X是原始输入，我们用对角线为1的单位矩阵来表示，维度N\*N；可以理解为是对节点的one-hot表示。Wo采用的参数是N\*K随机初始化(K=200)。

XWo 的维度就是N\*200，相当于对每个输入节点做了embedding，维度为200。

A \* XWo 这个矩阵乘法怎么理解才是理解图卷积的关键。复习一下矩阵乘法公式，发现新生成的L1这个N\*K矩阵的每一个节点的K个维度，都是对应该节点的相邻节点邻接权重乘以相邻节点在这个维度上的值的累加和。从而实现了通过一次卷积，GCN可以让每个节点都拥有其邻居节点的信息。（不准确的讲，图的邻接矩阵乘以图节点embedding，就相当于是做一次卷积)

下面是一个示意图

图卷积的实现过程

结论：新生成的0号节点的向量全部由相邻的1号节点和3号节点的向量等加权求和得到。从而实现了周边节点卷积(加权求和)得到新的自身的目的。

(邻接矩阵A第一行0 1 0 1表示0号节点和1,3号节点相连，和2号不连接)

如果要让节点拥有周边更广泛的节点信息，可以多次进行卷积。



上面的如果用邻接矩阵替代的话有两个缺点。

* 没有考虑节点自身对自己的影响，因为邻接矩阵对角线为0；
* 邻接矩阵没有被规范化，这在提取图特征时可能存在问题，比如邻居节点多的节点倾向于有更大的影响力。

因此更常用的公式是：



又称为规范化对称邻接矩阵(normalized symmetric adjacency matrix)。关于这个公式理解，可以参考

03 pytorch代码实现

下面介绍了两层图卷积网络的模型定义：

class gcn(nn.Module):

def \_\_init\_\_(self, X\_size, A\_hat, args, bias=True): # X\_size = num features

super(gcn, self).\_\_init\_\_()

self.A\_hat = torch.tensor(A\_hat, requires\_grad=False).float()

self.weight = nn.parameter.Parameter(torch.FloatTensor(X\_size, args.hidden\_size\_1))

var = 2./(self.weight.size(1)+self.weight.size(0))

self.weight.data.normal\_(0,var)

self.weight2 = nn.parameter.Parameter(torch.FloatTensor(args.hidden\_size\_1, args.hidden\_size\_2))

var2 = 2./(self.weight2.size(1)+self.weight2.size(0))

self.weight2.data.normal\_(0,var2)

if bias:

self.bias = nn.parameter.Parameter(torch.FloatTensor(args.hidden\_size\_1))

self.bias.data.normal\_(0,var)

self.bias2 = nn.parameter.Parameter(torch.FloatTensor(args.hidden\_size\_2))

self.bias2.data.normal\_(0,var2)

else:

self.register\_parameter("bias", None)

self.fc1 = nn.Linear(args.hidden\_size\_2, args.num\_classes)

def forward(self, X): ### 2-layer GCN architecture

X = torch.mm(X, self.weight)

if self.bias is not None:

X = (X + self.bias)

X = F.relu(torch.mm(self.A\_hat, X))

X = torch.mm(X, self.weight2)

if self.bias2 is not None:

X = (X + self.bias2)

X = F.relu(torch.mm(self.A\_hat, X))

return self.fc1(X)

# 第一层权重维度 args.hidden\_size\_1取200，

# 第二层权重维度args.hidden\_size\_2取20；

# args.num\_classes=5

最开始介绍的5本书的章节和单词构成的图，一共有100个章节节点和5000个单词节点。每个章节节点的标签是属于哪本书。一共五类。希望通过对其中50个章节的标签进行标记和训练，让网络学会剩下50个章节属于哪本书。属于半监督学习。