大作业链路预测实验报告

姓名：\_\_\_\_\_王涛\_\_\_ 学号：\_\_\_\_\_\_2020300400\_\_ 成绩：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

###### 前言

研究链路预测问题可以帮助我们理解网络演化的内在机制。比如针对一个特定的含时演化网络，七七八八的人提出了若干模型（若干机制），都可以生成一些人工网络。里面有一些簇系数接近，有一些平均距离接近，有一些模块结构类似，有一些度分布更像（模块结构和度分度哪个更像，还真不是个简单问题），那么怎么办呢，什么机制更可信呢，直接从网络拓扑特征来看不好比较。链路预测提供了一种比较简单的方法，只需要根据时间先后，把最后M条边移走不要（一般是几百到几千），然后根据各自的机制从M条边以前演化生长，看看谁能把那M条边长出来，至少谁能长得最准确，就是个判断标准了。

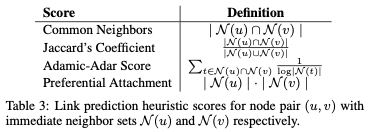
第二链路预测可以用来评估各种节点相似性的度量指标。节点相似性是个基本问题，聚类啊，群落啊，都和这个有关系。在很多情况下，可以假设越相似的节点越容易相连（不完全，比如性关系网络，男的和男的连，毕竟是少数），那么利用这个相似性指标进行链路预测，有的效果好有的效果坏，优劣就出来了（当然不是绝对的，但是有参考价值）。

第三应用起来有前景。比如说蛋白质折叠网络、新陈代谢网络、基因调控网络等等生物网络，一条边是否存在，完全靠实验。由于实验很贵，如果能够在实验前先作预测，而且预测精度还行，可以省钱。这个就叫做missing link prediction——预测本来存在但是我们还不知道的。还有比如在线的社区网络，可以通过这个方法进行预测，根据预测结果推荐朋友。这个叫做link prediction for evolving networks——预测将来可能存在的连接。这两种应用研究都大有可为。

# **链路预测相关方法**

 1.[启发式算法](https://so.csdn.net/so/search?q=%E5%90%AF%E5%8F%91%E5%BC%8F%E7%AE%97%E6%B3%95&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/adreammaker/article/details/_blank)（基于打分的算法）

参考文献：Link prediction Based on Graph Neural Network (NeurIPS 2018)  
首先，链路预测的启发式算法（也可以称之为启发式得分方法、基于打分的方法）都有一个前提假设就是更“相似”的节点更容易产生链接。（这里的相似是人为定义的，例如可以定义为共同邻居越多节点越相似）。  
Node2vec的作者在这里根据一对节点(u,v)的邻居来定义启发式得分heuristic scores（也可理解为相似度得分）如下所示：

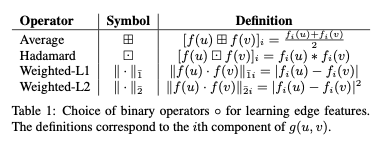


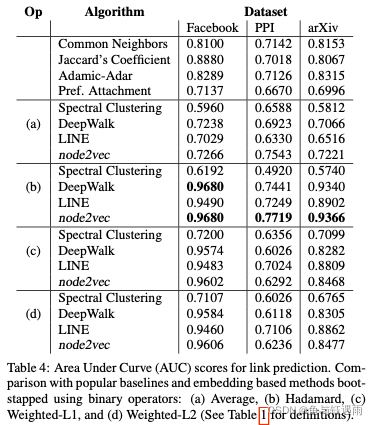
根据相似度得分的高低，设定一个阈值，我们就可以对节点对之间的关联进行预测。

2.基于图表示学习-图嵌入技术的方法

Node2vec(2018)作者指出since none of feature learning algorithms have been previously used for link prediction. （以前没有特征学习算法被用于链路预测）

如何将表示学习用于链路预测呢？实现思路是什么？  
首先，从网络中学习每个节点的特征表示（[node](https://so.csdn.net/so/search?q=node&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/adreammaker/article/details/_blank) feature)，然后在一对节点(u,v)上，可以根据节点的特征通过一个二元操作o(u,v)得到边的特征，有了边的的特征就可以作为而分类问题来求解了。可以对边特征应用各种分类方法来进行关联预测。

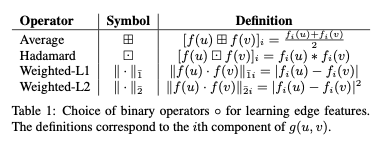
下图是作者采用的一些计算边特征的二元操作o:  
  
下图是作者的实验结果：和基于打分的方法和其他一些网络特征学习方法进行比较。



相关论文参考：DEMLP

3.基于图表示学习-图[深度](https://so.csdn.net/so/search?q=%E6%B7%B1%E5%BA%A6&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/adreammaker/article/details/_blank)学习技术的方法

图深度学习的方法技术的典型代表就是GCN，通过图卷积等图深度学习方法来进行节点的特征表示，而且与图嵌入技术不同，图深度学习技术通常可以结合下游任务进行端到端的训练），即学到的节点对特征直接在模型中进行拼接（相加）等融合操作操作（这些操作的目的可以等价理解为获取边的特征），并把节点对的特征用于下游任务-例如链路预测或者节点分类。

这里的融合方法除了拼接和相加，还可以用下面这些操作，类似于得到变的特征：  
  
然后讲这些特征投入端到端的分类模块中。融合的过程也需要在模型内完成。

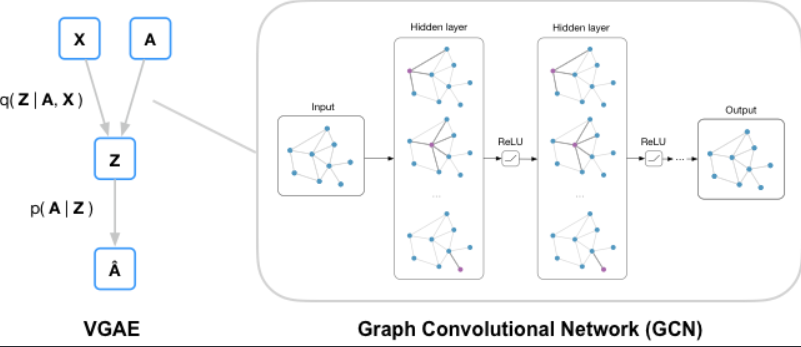
本次使用基于图表示学习-图[深度](https://so.csdn.net/so/search?q=%E6%B7%B1%E5%BA%A6&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/adreammaker/article/details/_blank)学习技术的方法

模型主要使用图[神经网络](https://so.csdn.net/so/search?q=%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/newlw/article/details/_blank)，如gae、vgae等

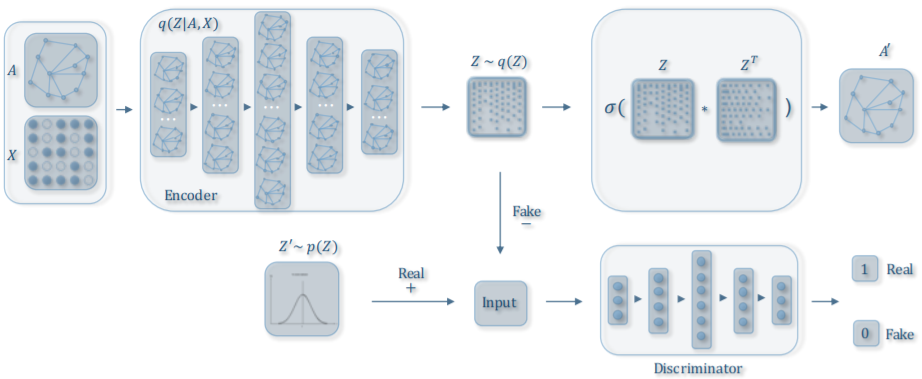
VAE 通过构建两个神经网络来分别学习均值和方差 ，这样我们便能得到样本  的专属均值和方差了，然后从专属分布中采样出 ，然后通过生成器得到 ，并通过最小化重构误差来进行约束 。

但隐变量是通过采样得到的，而不是经过编码器算出来的。这样的重构过程中免不了受到噪声的影响，噪声会增加重构的难度，不过好在这个噪声的强度可以通过方差反应，方差可以通过一个神经网络得到计算，所以最终模型为了更好的重构会尽量让模型的方差为零，而方差为零时，就不存在随机性了，这样模型最终会是一组均值，便退化成了普通的 AutoEncoder。

###### 方法

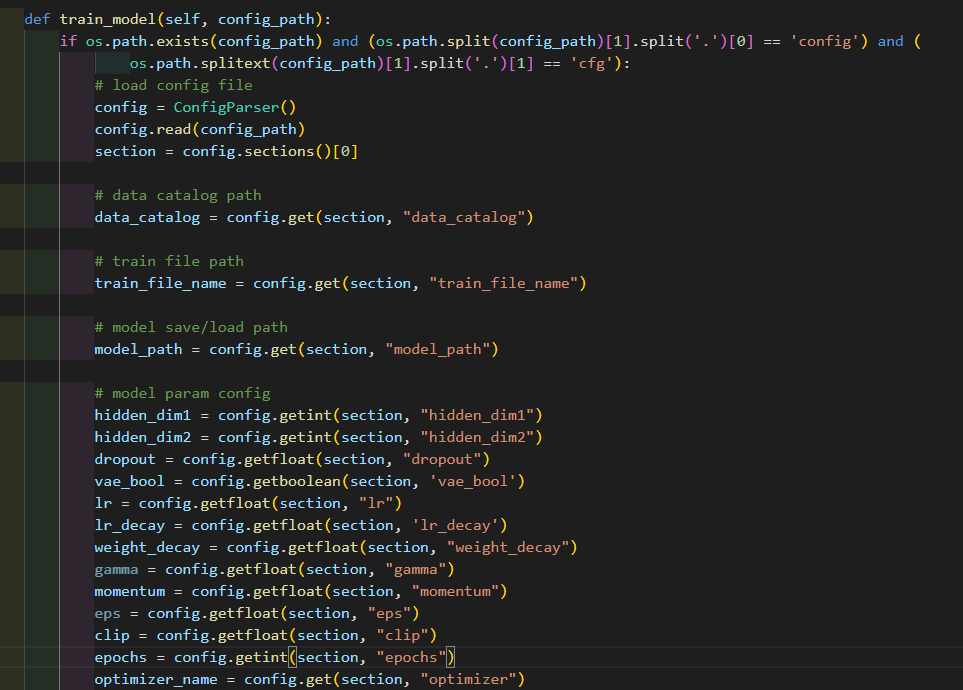
1.GCNModelVAE(src/vgae)：图卷积自编码和变分图卷积自编码(config中可配置使用自编码或变分自编码)，利用gae/vgae作为编码器，InnerProductDecoder作解码器。 [Variational Graph Auto-Encoders](https://arxiv.org/pdf/1611.07308.pdf) 。

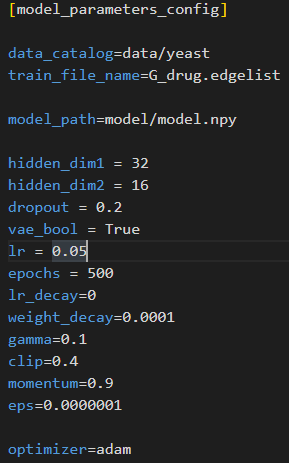
2.GCNModelARGA(src/arga)：对抗正则化图自编码，利用gae/vgae作为生成器；一个三层前馈网络作判别器。 [Adversarially Regularized Graph Autoencoder for Graph Embedding](https://arxiv.org/pdf/1802.04407v2.pdf) 。



总体来说GCN的前向过程就是实体嵌入索引查找->图卷积第一层->ReLU->drop\_out->图卷积第二层->用于链路预测或者实体分类的特征向量。

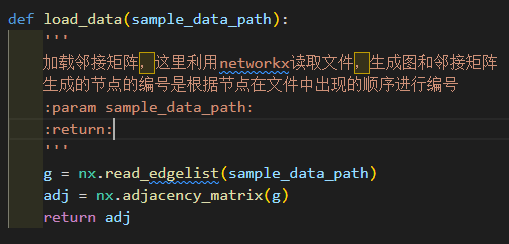
第一步：导入参数





第二步：加载数据

将所给数据（领接矩阵）改为src形式便于读取与表示



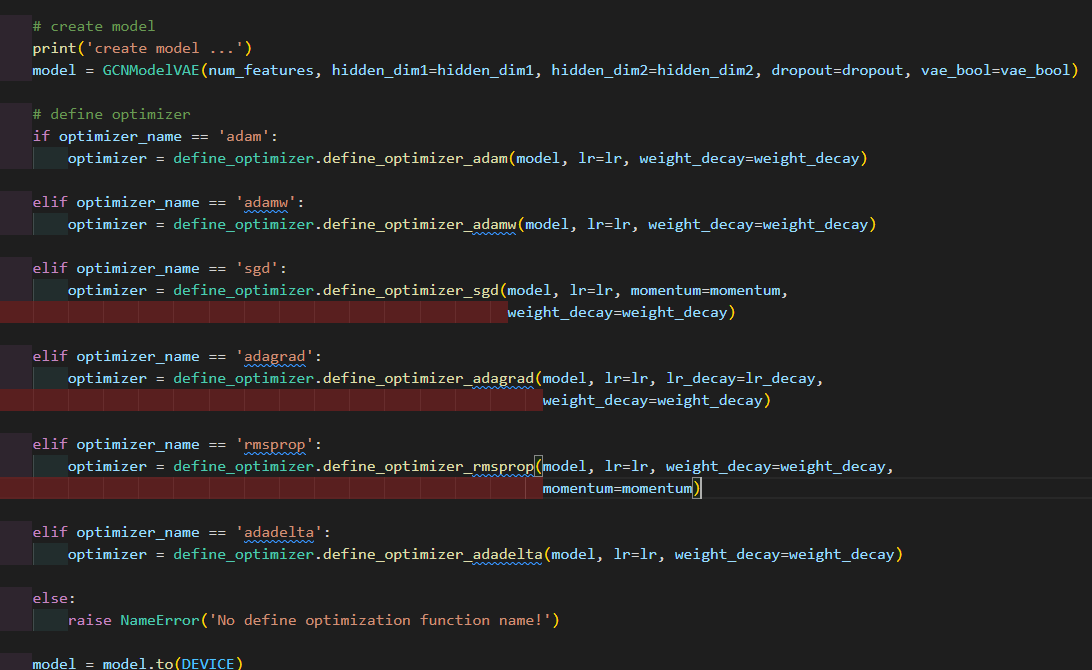
加载邻接矩阵，这里利用networkx读取文件，生成图和邻接矩阵生成的节点的编号是根据节点在文件中出现的顺序进行编号

第三步：数据处理

去除对角线元素，返回初始数据（adj\_orig）的对角元素（一维），并增加一维，抽出adj\_orig的对角元素并构建只有这些对角元素的对角矩阵

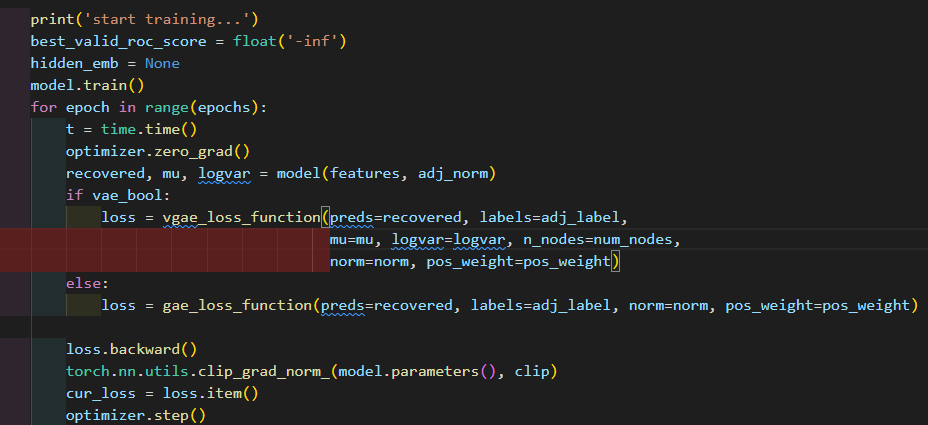
注意，adj的每个元素非1即0。pos\_weight是用于训练的邻接矩阵中负样本边（既不存在的边）和正样本边的倍数（即比值），这个数值在二分类交叉熵损失函数中用到， 如果正样本边所占的比例和负样本边所占比例失衡，比如正样本边很多，负样本边很少，那么在求loss的时候可以提供weight参数，将正样本边的weight设置小一点，负样本边的weight设置大一点，此时能够很好的平衡两类在loss中的占比，任务效果可以得到进一步提升。参考：https://www.zhihu.com/question/383567632  负样本边的weight都为1，正样本边的weight都为pos\_weight

第四部：建立模型”



 稀疏张量被表示为一对致密张量：一维张量和二维张量的索引。可以通过提供这两个张量来构造稀疏张量

第五步：开始训练

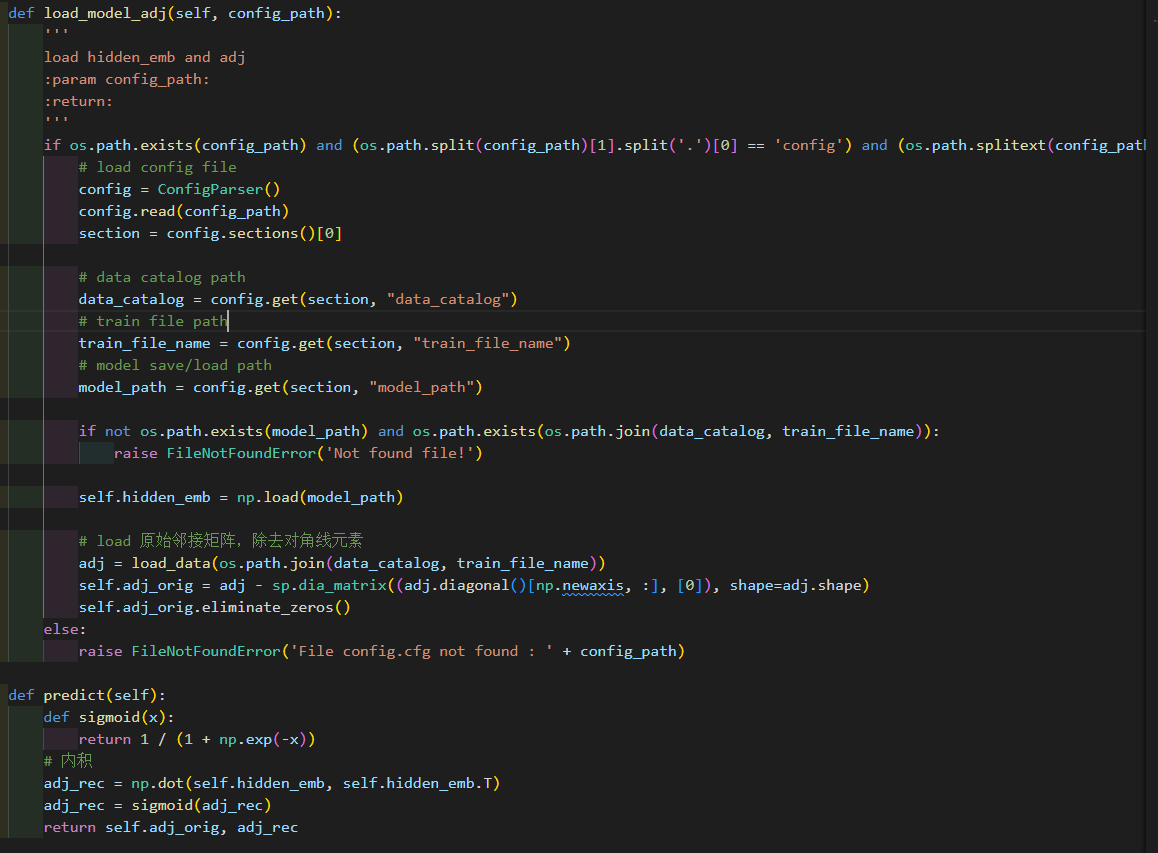


第六步：评估验证集

保存最好的roc score

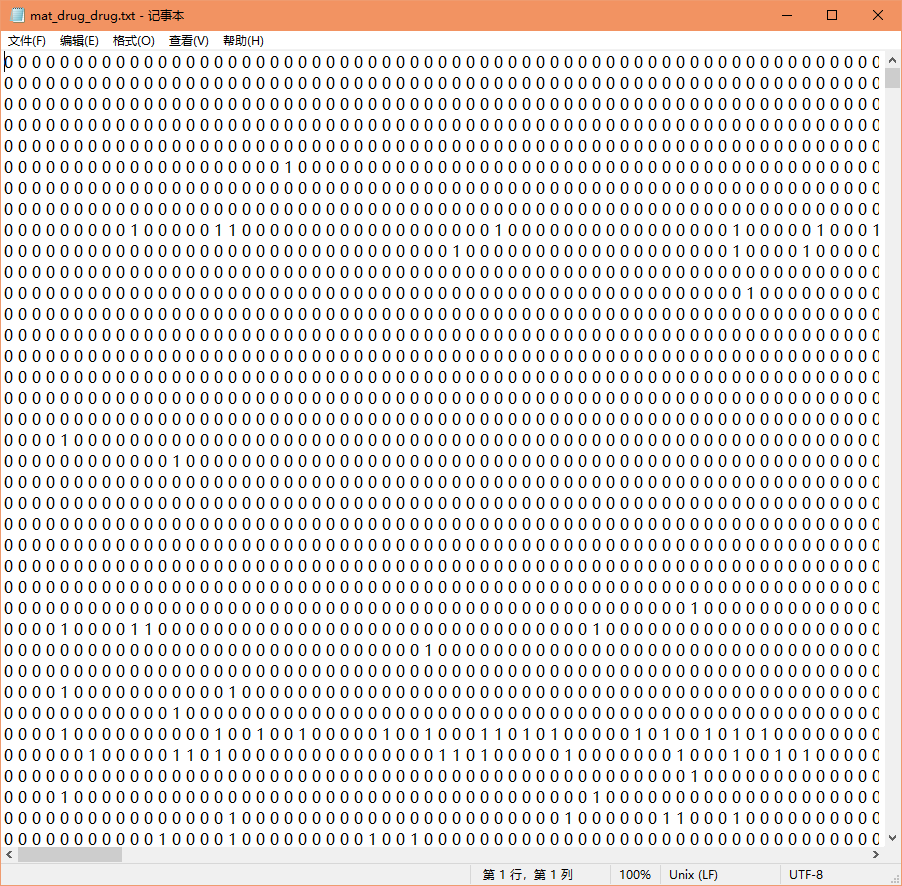
不需要保存整个model，只需保存hidden\_emb，因为后面的解码是用hidden\_emb内积的形式作推断

第七步：预测

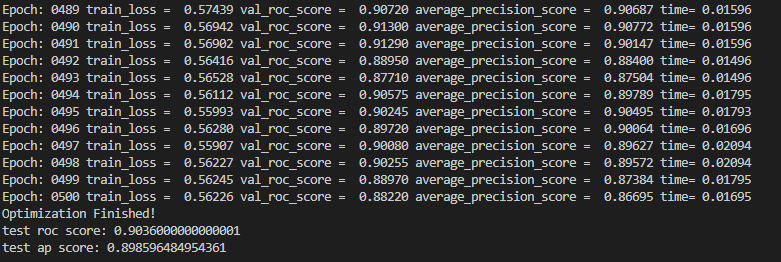


###### 结果

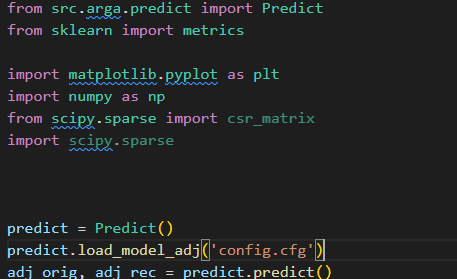
本次使用药物-药物相互作用网络(708个药物节点)



开始训练：epoch==500

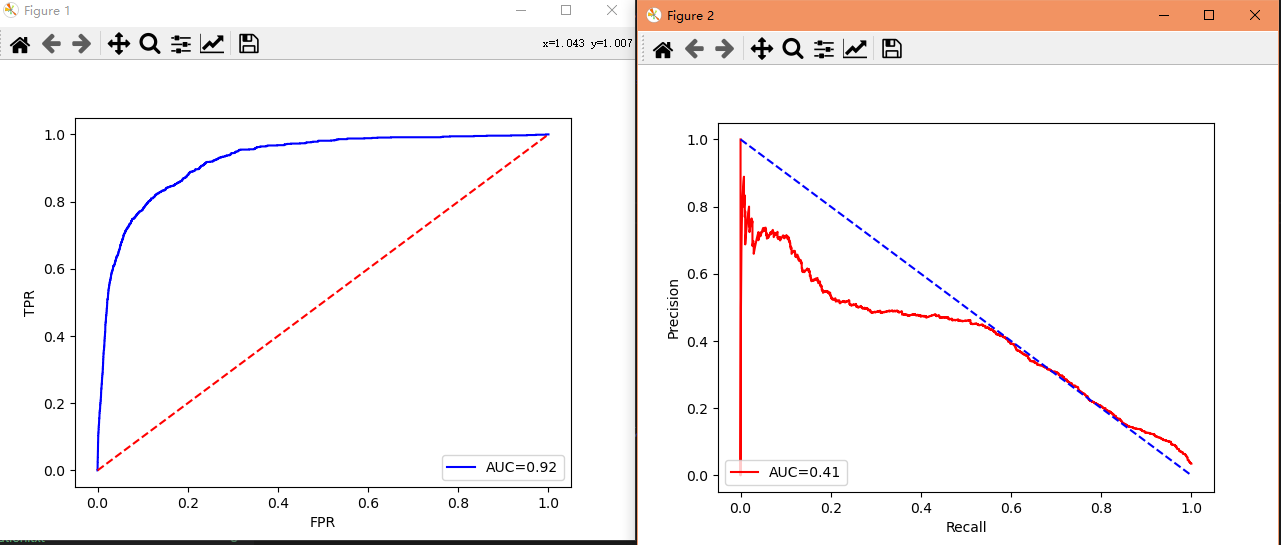


开始预测：



会返回原始的图邻接矩阵和经过模型编码后的hidden embedding经过内积解码的邻接矩阵，可以对这两个矩阵进行比对，得出link prediction.

绘图：



生成原矩阵和预测矩阵txt文件

预测结束之后按照模型推断出来的edge score来做链路预测，概率阈值选为0.9，超过0.9预测有链接，小于0.9无连接。



可以看出来使用GCN的方法链路预测表现远远比使用传统节点相似度方法的表现好。我个人认为主要是受益于这两个数据集的样本量都比较大，可以把深度学习模型的节点嵌入优化得特别好。而相比之下传统方法尤其是节点相似度方法在这个场景下最大的不足就是无法考虑边的类型，这会导致明明是不同边传递的信息会被一视同仁，从而丢失了不同边的独特性，最后使得节点间相似度计算时漏掉了这些重要信息。传统方法其他的问题还有可能就是以前大家提出这些相似度计算特征的时候往往都是从小规模的数据集上归纳出的方法，数据集规模大了之后会有一些以前没注意到但是很重要的规律被忽略了。

Top:

1:0.971468

2:0.969304

3:0.96898

4:0.967797

5:0.967525

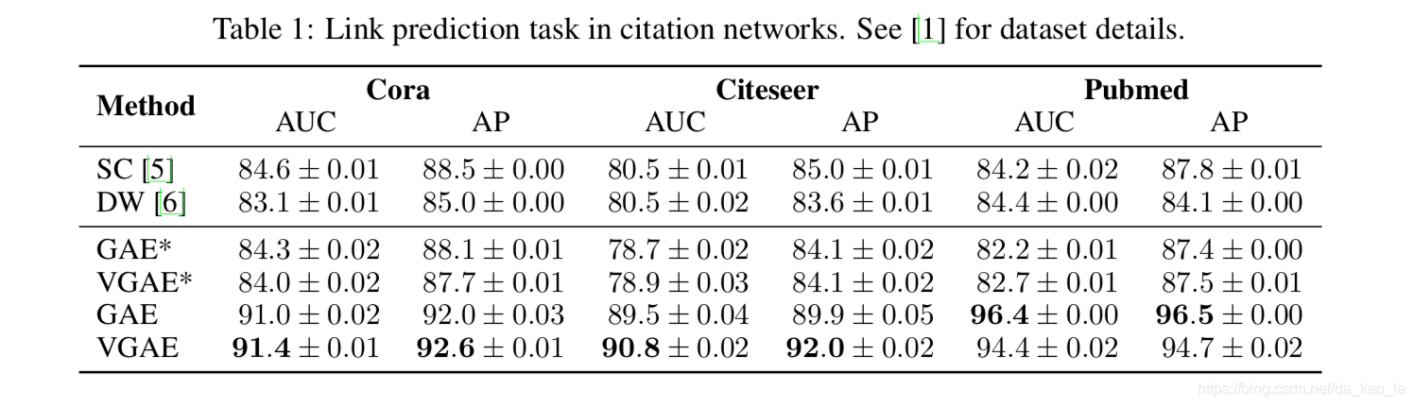
6:0.96701

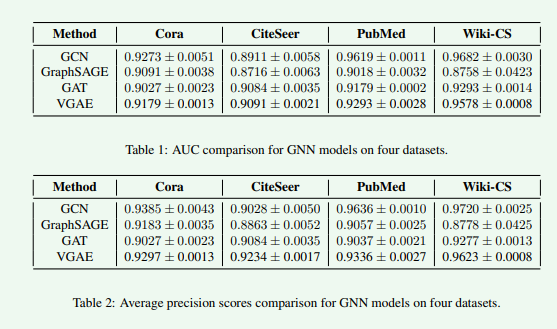
7:0.965493

8:0.96531

9:0.965187

10:0.96442

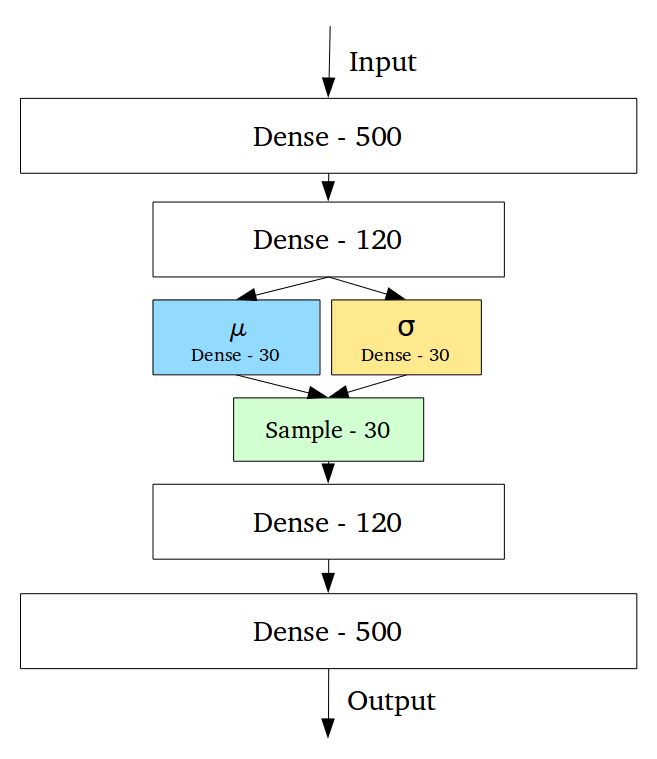




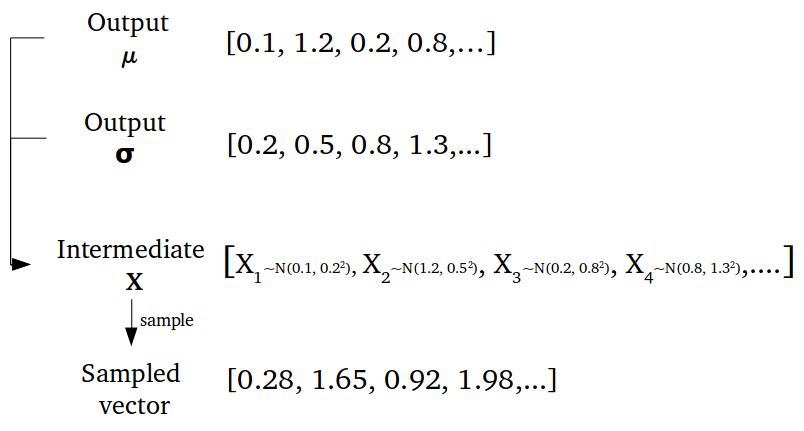
###### 结论

变分自动编码器（VAEs）具有一个独特的性质，可以将它们与vanilla自动编码器分离开来，正是这种特性使其在生成建模时非常有用：它们的潜在空间在设计上是连续的，允许随机采样和插值。

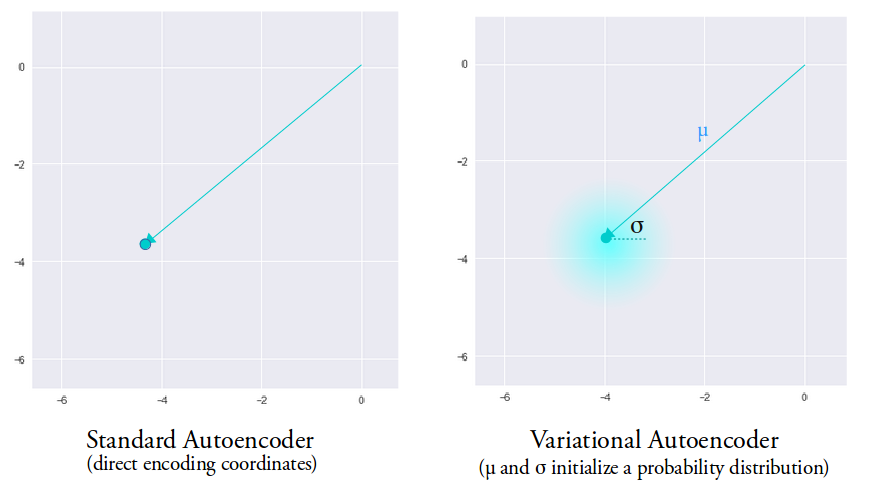
它通过做一些约束来达到这个目的：使编码器不输出大小为n的编码矢量，而是输出两个大小为n的矢量：平均矢量μ和另一个标准偏差矢量σ。



它们构成了长度为n的随机变量向量的参数，μ的第i个元素和σ是我们抽样的第i个随机变量X i的平均值和标准差，并用它们获得采样编码，然后传给解码器：

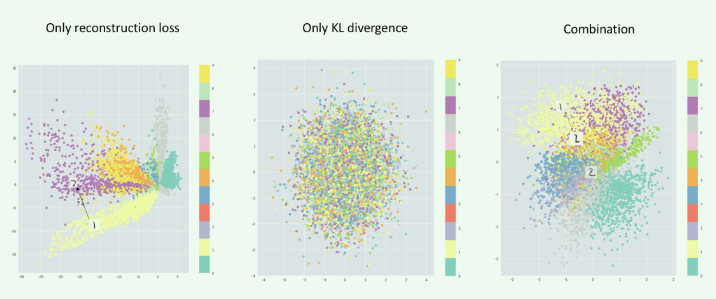


这种随机生成意味着，即使对于相同的输入，虽然平均值和标准偏差保持不变，但是实际编码会在采样过程中发生些许变化。



直观地，平均向量控制输入的编码的中间位置，而标准偏差控制“区域”，即编码可以改变多少。 由于编码是从“圆”（分布）内的任意位置随机产生的，因此解码器不仅可以获得指向该类样本的潜在空间中的单个点，而且所有附近的点也都是相同的。 这使得解码器不仅能够解码潜在空间中的单个特定编码（使可解码的潜在空间不连续），而且还能够稍微改变，因为解码器作用在相同输入编码的一系列变化上。

变分自编码器的主要优点是我们能够学习输入数据的平滑潜在状态表示。对于标准的自编码器，我们只需要学习一个编码，它允许我们重现输入。如左图所示，只关注重构损失允许我们分离出类（在这种情况下是MNIST数字），这使我们的解码器模型能够重现原始手写数字，但是它的潜在空间内的数据分布不均匀。换句话说，潜在空间中有一些区域不代表我们观测到的任何数据。



一直以来我都非常惊叹这种划时代的研究成果是怎么搞出来的，明明每个人都是两个肩膀扛一个脑袋，为啥人家就能想到把深度学习用到计算机视觉或者图类型数据上面。我这里自己想了一种可能合理的想到这种用法的猜想，可能GCN的提出者心里就在想，深度学习当前已经证明了大数据量情况下比任何手动特征工程还要好，那么除了计算机视觉以外还有什么场景是数据量大但是我们人工手动特征表现还有待提高的呢？有了，很多边的图网络数据集。然后接下来就是靠着自己高超的理论和编程功底扫清实现GCN应用的障碍。