

MGC-RM

GAT

GAT 的核心思想

GAT 的数学描述

1. 初始化
2. 注意力机制
3. 聚合
4. 更新

点积表示相关性

解释各个组成部分

为什么可以用来表示相关度

公式的意义

激活函数的作用

GAT的基本训练流程:

训练的模型架构:

MSE

相似度系数

Reference indicators

1. Natural Connectivity
2. MSE
3. AEC

Test Results

Similarity Score

5. GP : 75 (shanshili)

Reference indicators

LOG

241106

241105

241104

241103

241102

241101

241025

241024

241023※

241022

241017

241016

241015

241014

241013

241012

MGC-RM

GAT

GAT (Graph Attention Networks, 图注意力网络) 是一种图神经网络 (GNN) 的变体, 它通过引入自适应权重分配机制来改进传统的图卷积网络 (GCN)。GAT 通过学习邻居节点之间的注意力系数来动态地聚合邻居的信息, 从而更好地捕捉图中节点之间的关系 (某种节点表示, 对自己的表示中有权重的考虑邻居节点)。

GAT 的核心思想

GAT 的核心思想在于通过注意力机制为每个节点的不同邻居分配不同的权重, 这样可以更加灵活地处理不同节点的重要性。具体来说, GAT 的关键步骤包括:

1. **初始化**: 每个节点都有自己的特征表示 (通常是特征向量)。
2. **注意力机制**: 通过计算注意力系数 (attention coefficients) 来决定邻居节点的重要性。
3. **聚合**: 根据注意力系数加权聚合邻居节点的信息。
4. **更新**: 使用聚合的信息来更新每个节点的特征表示。

GAT 的数学描述

以下是 GAT 的数学描述:

1. 初始化

假设有图 ($G = (V, E)$), 其中 (V) 是节点集, (E) 是边集。每个节点 (v_i) 有一个特征向量 (\mathbf{h}_i)。

2. 注意力机制

定义注意力系数 ($\alpha_{i,j}$), 它是节点 (v_i) 和 (v_j) 之间的关系强度的度量。GAT 使用共享的注意力机制 (α) 来计算注意力系数:

$$[e_{ij} = \text{LeakyReLU}(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j])]$$

单层的全连接层, 表示一种: 一对向量到一个标量的映射, 表示两个向量的相关度

其中:

- (\mathbf{a}) 是一个可学习的向量, 用于计算注意力分数。
- (\mathbf{W}) 是一个权重矩阵, 用于变换节点特征。
- ($||$) 表示特征向量的拼接 (concatenation)。
- (e_{ij}) 是未归一化的注意力分数。

接下来, 对每个节点 (v_i) 的邻居节点计算注意力系数:

$$[\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})}] \quad \text{针对一个节点, 其邻居对其的权重系数。所有邻居归一化处理, 分配权重 softmax 用于归一化}$$

其中 ($\mathcal{N}(i)$) 是节点 (v_i) 的邻居集。

3. 聚合

使用注意力系数加权聚合邻居节点的信息:

$$[\mathbf{h}'_i = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j \right)]$$

其中 (σ) 是激活函数, 如 $ReLU$ 。

4. 更新

更新节点(v_i)的特征表示:

$$[\mathbf{h}'_i = \text{Concat}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}, \dots, \mathbf{h}_i^{(K)})]$$

这里(K)是注意力头的数量，通常使用多头注意力机制来增强模型的表达能力。

内积表示向量的相似度

点积表示相关性

在GAT(Graph Attention Networks)中， $\text{LeakyReLU}(\mathbf{a}^T [\mathbf{Wh}_i || \mathbf{Wh}_j])$ 用于计算两个节点(v_i)和(v_j)之间的注意力系数。这个表达式实际上是一个自定义的函数，用于衡量两个节点之间的相似度或相关性。下面详细解释一下各个组成部分及其意义：

解释各个组成部分

1. (\mathbf{Wh}_i) 和 (\mathbf{Wh}_j) :
 - (\mathbf{W})是一个权重矩阵，用于将原始节点特征向量(\mathbf{h}_i)和(\mathbf{h}_j)转换成新的特征表示。
 - (\mathbf{Wh}_i) 和 (\mathbf{Wh}_j) 分别表示节点(v_i)和(v_j)在经过线性变换后的特征向量。
2. $([\mathbf{Wh}_i || \mathbf{Wh}_j])$:
 - ($||$)表示向量拼接(concatenation)操作，即将两个向量合并成一个新的向量。这样做是为了保留两个节点的特征信息，并通过拼接后的向量来计算它们之间的关系。
 - 例如，如果 (\mathbf{Wh}_i) 和 (\mathbf{Wh}_j) 都是(d)-维向量，则 $([\mathbf{Wh}_i || \mathbf{Wh}_j])$ 将是一个($2d$)-维向量。
3. (\mathbf{a}^T) 和 (\mathbf{a}) :
 - (\mathbf{a})是一个可学习的向量，用于衡量两个节点特征向量的相似度或相关性。
 - (\mathbf{a}^T) 表示(\mathbf{a})的转置，使用转置后可以进行点积运算。
4. $(\mathbf{a}^T [\mathbf{Wh}_i || \mathbf{Wh}_j])$:
 - 这个表达式计算了(\mathbf{a})向量与拼接后的特征向量($([\mathbf{Wh}_i || \mathbf{Wh}_j])$)的点积。
 - 点积的结果是一个标量，用于衡量两个节点特征向量的相似度或相关性。
5. $(\text{LeakyReLU}(x))$:
 - LeakyReLU是一个激活函数，用于引入非线性。LeakyReLU的定义如下:
$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases}$$
 - 其中(α)是一个小的正数(例如0.01)，用于防止梯度消失问题。

为什么可以用来表示相关度

1. **衡量相似度**:
 - $(\mathbf{a}^T [\mathbf{Wh}_i || \mathbf{Wh}_j])$ 计算的结果是一个标量，这个标量反映了节点(v_i)和(v_j)之间的相似度或相关性。
 - 如果两个节点的特征向量相似，则它们的拼接向量与(\mathbf{a})向量的点积也会较大。
- 假设(\mathbf{Wh}_i)和(\mathbf{Wh}_j)相似，意味着它们在向量空间中的位置接近，即它们之间的角度较小。在这种情况下：
 1. **点积的大小**: 如果(\mathbf{Wh}_i)和(\mathbf{Wh}_j)相似，它们在相同方向上的投影较长，因此(\mathbf{Wh}_i)和(\mathbf{Wh}_j)的拼接向量($([\mathbf{Wh}_i || \mathbf{Wh}_j])$)在相同方向上的投影也较长。因此，当与(\mathbf{a})向量进行点积运算时，结果较大。
 2. **向量(\mathbf{a})的作用**: 向量(\mathbf{a})是一个可学习的向量，用于衡量两个节点特征向量的相似度。如果(\mathbf{a})在($([\mathbf{Wh}_i || \mathbf{Wh}_j])$)的某些维度上有较大的权重，则这些维度上的相似性会对最终的点积结果贡献更大。
2. **引入非线性**:

- LeakyReLU 作为激活函数，可以进一步增强模型的表达能力，使得注意力系数更具区分性。
- LeakyReLU 保留了负值部分的信息，使得模型在处理负数特征时也能保持一定的敏感度。

公式的意义

综合起来，公式 $(\text{LeakyReLU}(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j]))$ 的意义在于通过一个自定义的函数来衡量两个节点之间的相似度或相关性，并通过非线性激活函数进一步增强这种衡量的效果。

理解为什么“如果两个节点的特征向量相似，则它们的拼接向量与 (\mathbf{a}) 向量的点积也会较大”可以从向量空间的角度来解释。这里的关键在于理解点积运算的性质以及特征向量相似性的含义。

在 GAT (Graph Attention Networks) 中，激活函数在计算注意力系数 (attention coefficients) 的过程中起着重要的作用。具体来说，在 GAT 的注意力机制中，使用了 LeakyReLU 激活函数来处理节点特征向量的点积结果。下面我们详细探讨激活函数在注意力系数公式中的意义。

激活函数的作用

1. 引入非线性：

- 激活函数的主要作用之一是引入非线性，使得模型能够拟合更复杂的函数关系。如果没有激活函数，整个模型就相当于一个线性模型，其表达能力有限。
- LeakyReLU 是一种常用的激活函数，它在正数部分保持线性，而在负数部分引入了一个小的斜率，避免了 ReLU 的“死区”问题。

2. 标准化输出：

- 激活函数可以对输出进行标准化，使其落在一个特定的范围内。在 GAT 中，LeakyReLU 可以确保注意力系数 (e_{ij}) 的值不会过大或过小，从而影响后续的归一化操作。
- 通过 LeakyReLU，即使输入为负数，输出也不会完全为零，而是保留了一定的信息。

3. 增强区分性：

- 激活函数可以增强不同节点之间的区分性。LeakyReLU 可以放大或缩小输入信号的幅度，从而使得不同节点之间的注意力系数更有区分性。
- 例如，对于两个相似的节点，它们的注意力系数可能会非常接近，但在经过 LeakyReLU 激活之后，它们的差异会被放大，从而更好地捕捉节点之间的关系。

4. 数值稳定性：

- 激活函数可以帮助提高数值稳定性。LeakyReLU 可以避免 ReLU 在负数区域的梯度消失问题，从而有助于训练过程中的梯度传播。

GAT (Graph Attention Networks) 是一种基于图结构数据的深度学习模型。它主要用于处理图数据中的节点分类、边预测等问题。GAT的设计目的是为了在图中捕捉局部结构信息，并且赋予不同的邻居节点不同的权重，这通过注意力机制来实现。

GAT的基本训练流程：

1. 数据准备：

- 首先需要准备好图数据，包括 **节点特征矩阵**、**邻接矩阵**（表示节点之间的连接关系）等。
- 对于监督学习任务，还需要准备标签数据。

2. 模型构建：

- 定义GAT层：每一层中，节点会根据其邻居的信息以及它们之间的关系进行信息传递，并通过注意力机制计算出不同邻居的重要性权重。
- 可以堆叠多个GAT层来构建深层的网络结构。

3. 损失函数定义：

- 根据具体任务选择合适的损失函数，如交叉熵损失用于分类任务。

4. 优化器设置：

- 选择一个优化算法，如Adam或SGD，并设定学习率等超参数。

5. 训练过程:

- 使用前向传播计算输出。
- 计算损失值，并使用反向传播更新模型参数。
- 重复上述步骤直到达到预设的训练轮数或满足停止条件。

6. 评估与调整:

- 在验证集上评估模型性能，根据结果调整超参数或模型结构。
- 使用测试集评估最终模型性能。

训练的模型架构:

GAT通常包含以下组件：

- 输入层：接收节点特征。
- 多个GAT层：每个GAT层都会为输入的节点特征计算一个注意力系数矩阵，然后利用这个矩阵对邻居节点的信息进行加权求和，得到新的节点特征表示。
- 输出层：根据任务需求，可能是分类层或其他类型的层。

在实际应用中，GAT可以和其他神经网络层结合使用，比如全连接层、池化层等，以适应更复杂的数据结构和任务需求。此外，为了防止过拟合，还可以在模型中加入正则化项或使用dropout技术。

MSE

均方误差 (Mean Squared Error, MSE) 是一个非常常用的损失函数，特别是在回归任务中。它也可以用于其他类型的机器学习任务，比如自编码器中的重构误差、强化学习中的价值函数估计等。以下是MSE作为损失函数的一些原因：

1. 易于理解和计算：

MSE公式简单直观，计算也非常直接。对于给定的一组预测值 (\hat{y}) 和真实值 (y)，MSE定义为：

$$[\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2]$$

这里 (n) 是样本数量，(y_i) 是真实值，(\hat{y}_i) 是预测值。

2. 对异常值敏感：

由于MSE是预测误差的平方和，因此它会对较大的预测误差（即异常值）更加敏感。这意味着模型会被驱动去减小大的误差，从而可能提高总体的预测精度。

3. 导数简单：

MSE的导数也很容易计算，这对于使用梯度下降法优化参数非常重要。MSE的导数为：

$$[\frac{\partial \text{MSE}}{\partial \hat{y}} = 2(\hat{y} - y)]$$

这意味着优化过程中计算梯度比较容易。

4. 平滑性：

MSE是一个连续可微的损失函数，这使得它非常适合使用梯度下降法进行优化。相比之下，像绝对误差这样的损失函数在某些点上不可微，可能会导致优化过程中的问题。

总之，MSE作为一种损失函数具有许多优点，使得它在多种机器学习任务中都是一个有效的选择。然而，需要注意的是，MSE对于异常值的敏感性有时可能会成为缺点，特别是当数据集中存在很多异常值时。在这种情况下，可能需要考虑使用其他损失函数，如均值绝对误差 (MAE) 或Huber损失等。

相似度系数

$$\beta_{mj} = \frac{\cos(\mathbf{h}_m, \mathbf{h}_j)}{\sum_{k=1}^N \cos(\mathbf{h}_m, \mathbf{h}_k)}$$

您提供的公式看起来像是一个基于余弦相似度的加权版本，用于衡量两个向量 (\mathbf{h}_m) 和 (\mathbf{h}_j) 之间的相似度。在这个公式中，(N) 表示一组参考向量的数量，(\mathbf{h}_k) 是这一组参考向量中的第 (k) 个向量。让我们逐步解析这个公式：

1. 分子部分：

- ($\cos(\mathbf{h}_m, \mathbf{h}_j)$) 是向量 (\mathbf{h}_m) 和 (\mathbf{h}_j) 之间的余弦相似度。余弦相似度衡量了这两个向量的方向上的相似性，而不是大小。它的取值范围在 [-1, 1] 之间，其中 1 表示完全相同的方向，0 表示方向完全不同，-1 表示完全相反的方向。

$$(\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}| |\mathbf{B}|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}})$$

2. 分母部分：

- ($\sum_{k=1}^N \cos(\mathbf{h}_m, \mathbf{h}_k)$) 是向量 (\mathbf{h}_m) 与所有 (N) 个参考向量的余弦相似度之和。这个和反映了 (\mathbf{h}_m) 与整个参考集的平均相似度。

3. 整体意义：

- 这个公式实际上是对 (\mathbf{h}_m) 和 (\mathbf{h}_j) 之间的余弦相似度进行了加权，权重由 (\mathbf{h}_m) 与其他所有参考向量的相似度决定。换句话说，如果 (\mathbf{h}_m) 与大多数参考向量都高度相似，那么 (\mathbf{h}_m) 和 (\mathbf{h}_j) 之间的相似度就会降低，反之亦然。

Reference indicators

1. Natural Connectivity

自然连通性 (Natural Connectivity) 是一种衡量网络或图的鲁棒性和连通性的指标。它不仅考虑了网络的连通性，还考虑了节点之间的路径多样性。自然连通性越高，表明网络在面对节点或边的删除时更加鲁棒。

自然连通性的定义

自然连通性 ($\phi(G)$) 定义为：

$$\phi(G) = \frac{1}{n} \sum_{k=1}^n \log(\lambda_k)$$

其中：

- (G) 是一个图。
- (n) 是图的节点数。
- (λ_k) 是图的拉普拉斯矩阵的第 (k) 个特征值 (不包括零特征值)。

计算自然连通性

要计算自然连通性，你需要以下几个步骤：

- 构建图的邻接矩阵：**表示图中节点之间的连接关系。
- 计算图的拉普拉斯矩阵：**拉普拉斯矩阵 (L) 定义为 ($L = D - A$)，其中 (D) 是度矩阵 (对角线上的元素是每个节点的度)，(A) 是邻接矩阵。
- 计算拉普拉斯矩阵的特征值：**特征值反映了图的结构特性。

4. **计算自然连通性**: 根据上述公式计算自然连通性。

示例代码

以下是一个使用 Python 和 NetworkX 库来计算自然连通性的示例:

```
import networkx as nx
import numpy as np

# 创建一个示例图
G = nx.Graph()
G.add_edges_from([(1, 2), (1, 3), (2, 3), (2, 4), (3, 4)])

# 计算图的拉普拉斯矩阵
L = nx.laplacian_matrix(G).todense()

# 计算拉普拉斯矩阵的特征值
eigenvalues = np.linalg.eigvals(L)

# 去除零特征值
non_zero_eigenvalues = eigenvalues[eigenvalues > 1e-10]

# 计算自然连通性
n = G.number_of_nodes()
natural_connectivity = (1 / n) * np.sum(np.log(non_zero_eigenvalues))

print(f"Natural Connectivity: {natural_connectivity}")
```

解释

1. **创建图**: 使用 `networkx` 库创建一个简单的无向图。
2. **计算拉普拉斯矩阵**: 使用 `nx.laplacian_matrix` 函数计算图的拉普拉斯矩阵。
3. **计算特征值**: 使用 `numpy.linalg.eigvals` 函数计算拉普拉斯矩阵的特征值。
4. **去除零特征值**: 自然连通性的定义中不包括零特征值，因此需要过滤掉接近零的特征值。
5. **计算自然连通性**: 根据公式计算自然连通性。

注意事项

- **数值稳定性**: 在计算特征值时，可能会遇到数值稳定性问题，特别是当特征值非常接近零时。因此，通常会设置一个小阈值（如 `1e-10`）来过滤掉这些值。
- **图的连通性**: 自然连通性适用于连通图。如果图不是连通的，可能需要分别计算每个连通分量的自然连通性。

希望这些信息对你理解自然连通性及其计算方法有所帮助！如果有其他问题或需要进一步的解释，请随时提问。

衡量网络或图的鲁棒性和连通性的指标

自然连通性 (Natural Connectivity) 作为一种衡量网络或图的鲁棒性和连通性的指标，具有以下几个关键特点和优势，使其成为一种有效的度量工具：

1. 考虑了所有路径

自然连通性不仅考虑了图的连通性，还考虑了节点之间的路径多样性。这意味着它不仅关注图是否连通，还关注连通的程度和路径的丰富性。这种全面的视角使得自然连通性能更准确地反映网络的结构特性。

2. 基于谱理论

自然连通性的定义基于图的拉普拉斯矩阵的特征值。拉普拉斯矩阵的特征值包含了图的大量结构性信息，如连通性、聚类性、路径长度等。通过分析这些特征值，可以更深入地了解图的拓扑结构。

3. 鲁棒性

自然连通性能够有效衡量网络在面对节点或边的删除时的鲁棒性。具体来说，自然连通性较高的网络在删除某些节点或边后，仍然能够保持较好的连通性。这是因为高自然连通性意味着网络中存在多条路径连接节点，即使某些路径被破坏，网络仍能通过其他路径保持连通。

4. 数学性质

自然连通性的定义具有良好的数学性质，使其易于计算和分析。通过特征值的对数平均，自然连通性提供了一个平滑且稳定的度量，能够在不同规模和结构的网络之间进行比较。

5. 反映全局和局部特性

自然连通性既反映了网络的全局连通性，也反映了局部连通性。全局连通性是指整个网络的连通程度，而局部连通性是指网络中各个子区域的连通性。自然连通性通过特征值的组合，能够同时捕捉这两种特性。

6. 对随机扰动的敏感性

自然连通性对网络中的随机扰动（如节点或边的随机删除）具有较高的敏感性。这意味着它可以有效检测网络在受到小概率事件影响时的性能变化，这对于评估网络的可靠性和稳定性非常重要。

7. 与其他度量的互补性

自然连通性可以与其他网络度量（如平均路径长度、聚类系数、介数中心性等）互补使用，提供更全面的网络特性描述。通过结合多种度量，可以更准确地评估网络的鲁棒性和连通性。

数学解释

自然连通性 ($\phi(G)$) 的定义为：

$$\phi(G) = \frac{1}{n} \sum_{k=1}^n \log(\lambda_k)$$

其中：

- (G) 是一个图。
- (n) 是图的节点数。
- (λ_k) 是图的拉普拉斯矩阵的第 (k) 个特征值（不包括零特征值）。

为什么特征值重要？

- **特征值的物理意义：**拉普拉斯矩阵的特征值反映了图的结构性质。例如，**最小非零特征值**（代数连通性）反映了图的连通性强度，而**较大的特征值反映了图的局部结构特性**。
- **对数平均：**通过对特征值取对数平均，自然连通性能够平滑地反映图的整体连通性，避免了单一特征值的极端影响。

总结

自然连通性通过综合考虑图的路径多样性和结构特性，提供了一种有效的度量网络鲁棒性和连通性的方法。它不仅能够反映网络的全局连通性，还能捕捉局部连通性，使其成为评估网络性能的重要工具。希望这些解释能帮助你更好地理解自然连通性的重要性。如果有其他问题或需要进一步的解释，请随时提问。

特征值log的求和是什么含义

特征值的对数求和在自然连通性的定义中起着重要的作用。为了更好地理解这一点，我们需要从几个角度来解释其含义：

1. 数学背景

特征值的意义

- **拉普拉斯矩阵**：对于一个图(G)，其拉普拉斯矩阵(L)定义为($L = D - A$)，其中(D)是度矩阵(对角线上的元素是每个节点的度)，(A)是邻接矩阵。
- **特征值**：拉普拉斯矩阵的特征值($\lambda_1, \lambda_2, \dots, \lambda_n$)包含了图的大量结构性信息。特别是，最小非零特征值(代数连通性)反映了图的连通性强度，而较大的特征值反映了图的局部结构特性。

对数的性质

- **对数函数**：对数函数($\log(x)$)是一个单调递增函数，且($\log(1) = 0$)。对数函数的一个重要性质是它可以将乘法转换为加法，即($\log(a \cdot b) = \log(a) + \log(b)$)。
- **对数平均**：通过对特征值取对数再求平均，可以平滑地反映特征值的分布，避免了单一特征值的极端影响。

2. 对数求和的含义

平滑化

- **平滑化**：对数函数可以将特征值的乘积转换为对数的和，从而平滑地反映特征值的分布。这意味着即使某些特征值非常大或非常小，对整体结果的影响也会被平滑化。
- **避免极端值**：对数函数可以避免特征值中的极端值对结果产生过大的影响，从而使自然连通性更加稳定和可靠。

信息熵

- **信息熵**：对数求和可以看作是一种信息熵的形式。在信息论中，信息熵用于衡量信息的不确定性。类似地，特征值的对数求和可以看作是图的结构信息的不确定性的度量。
- **多样性**：对数求和反映了特征值的多样性。如果特征值分布较为均匀，对数求和的结果会较大，反之则较小。这表明图的结构较为均匀，路径多样性较高。

3. 物理意义

连通性

- **连通性**：特征值的对数求和反映了图的连通性。较大的对数求和值意味着图的连通性较强，路径多样性较高。
- **鲁棒性**：自然连通性较高的图在面对节点或边的删除时，能够更好地保持连通性。这是因为高自然连通性意味着图中存在多条路径连接节点，即使某些路径被破坏，网络仍能通过其他路径保持连通。

4. 总结

特征值的对数求和在自然连通性的定义中起到了平滑化和信息熵的作用，能够更全面地反映图的结构特性。具体来说：

- **平滑化**：避免了单一特征值的极端影响，使自然连通性更加稳定和可靠。
- **信息熵**：反映了特征值的多样性，从而衡量图的结构信息的不确定性。
- **连通性**：反映了图的连通性和路径多样性，高自然连通性意味着图在面对节点或边的删除时具有较强的鲁棒性。

特征值的连乘积

特征值的连乘积在数学和图论中有着重要的意义，特别是在分析图的结构和性质时。特征值的连乘积通常与图的拉普拉斯矩阵相关联，可以提供关于图的一些深刻见解。以下是特征值连乘积的一些主要含义和应用：

1. 行列式的值

对于一个 ($n \times n$) 的矩阵 (A)，其特征值的连乘积等于该矩阵的行列式 (determinant)。

对于图的拉普拉斯矩阵 (L)，其行列式具有特殊的意义：

$$\det(L) = \prod_{i=1}^n \lambda_i$$

其中，(λ_i) 是拉普拉斯矩阵的特征值。对于连通图，拉普拉斯矩阵的一个特征值总是 0，因此行列式实际上等于非零特征值的连乘积。

2. 图的生成树数量

对于一个无向图 (G)，其生成树的数量可以通过拉普拉斯矩阵的非零特征值的连乘积来计算。具体来说，生成树的数量 ($T(G)$) 可以通过以下公式计算：

$$T(G) = \frac{1}{n} \prod_{i=1}^{n-1} \lambda_i$$

其中，($\lambda_1, \lambda_2, \dots, \lambda_{n-1}$) 是拉普拉斯矩阵的非零特征值，(n) 是图的节点数。

3. 图的谱半径

图的谱半径 (spectral radius) 是拉普拉斯矩阵的最大特征值。虽然谱半径本身不是特征值的连乘积，但它与特征值的连乘积一起可以提供关于图的结构和性质的更多信息。谱半径可以反映图的连通性和扩张性。

4. 图的鲁棒性和连通性

特征值的连乘积可以用于评估图的鲁棒性和连通性。高连乘积值通常意味着图的连通性较强，路径多样性较高，因此在网络受到攻击或节点故障时，图能够更好地保持连通性。

5. 图的稳定性

在控制理论和动力系统中，特征值的连乘积可以用于评估系统的稳定性。对于一个线性系统，特征值的连乘积可以反映系统的动态行为和稳定性。

2. MSE

(1) 均方误差 (MSE): MSE 用于比较选择出的节点经重构后的特征和原始特征之间的差异。MSE 越低，说明通过所选节点重构原始信息的能力越强。

$$MSE = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2 \quad (3-41)$$

其中， x_i 为节点 v_i 的原始特征， \hat{x}_i 为重构特征。

3. AEC

(2) 平均能量消耗 (Average Energy Consumption, AEC): AEC 度量了传感网的总体能量消耗。AEC 越低, 说明网络中的能量效率越高, 有利于网络的可持续性。

$$AEC = \frac{1}{N} \sum_{i=1}^N \left(E_0 - L \sum_{N_i} E_T - L \sum_{N_i} E_R \right) \quad (3-42)$$

其中, E_0 为传感器节点的初始能量, L 为网络寿命, E_T 为发送能耗, E_R 为接收能耗。根据无线传感网的无线传输能耗模型 (Radio Energy Dissipation Model, REDM) [93], 每个传感器节点在单次通信中的发送能耗和接收能耗分别为:

$$E_T = k(E_{elec} + \beta d^\alpha) \quad (3-43)$$

$$E_R = kE_{elec} \quad (3-44)$$

其中, E_{elec} 为单位信息的电子损耗, k 为信息量, β 为单位信息的功放损失, α 为传播衰减指数, d 为通信距离。

Test Results

Similarity Score

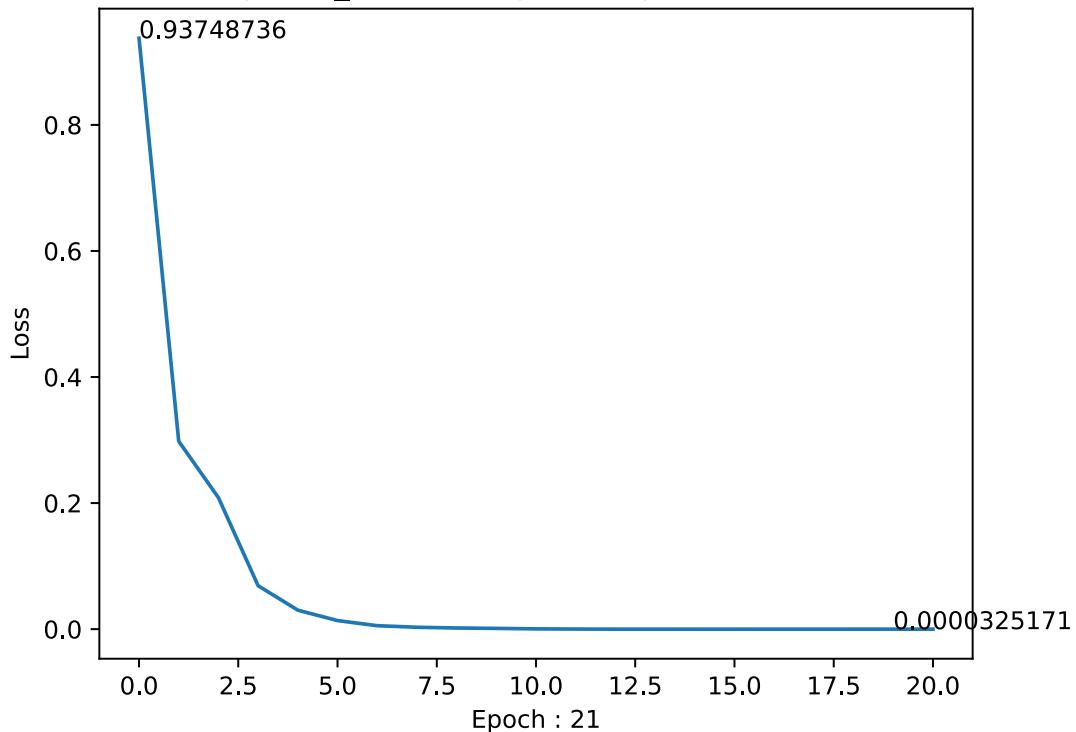
n450_d2000_Training_Loss_epoch_21_lr_0.0001

prediction_loss越低的点, 越适用于训练模型, (该点具有普适性, 越相同)

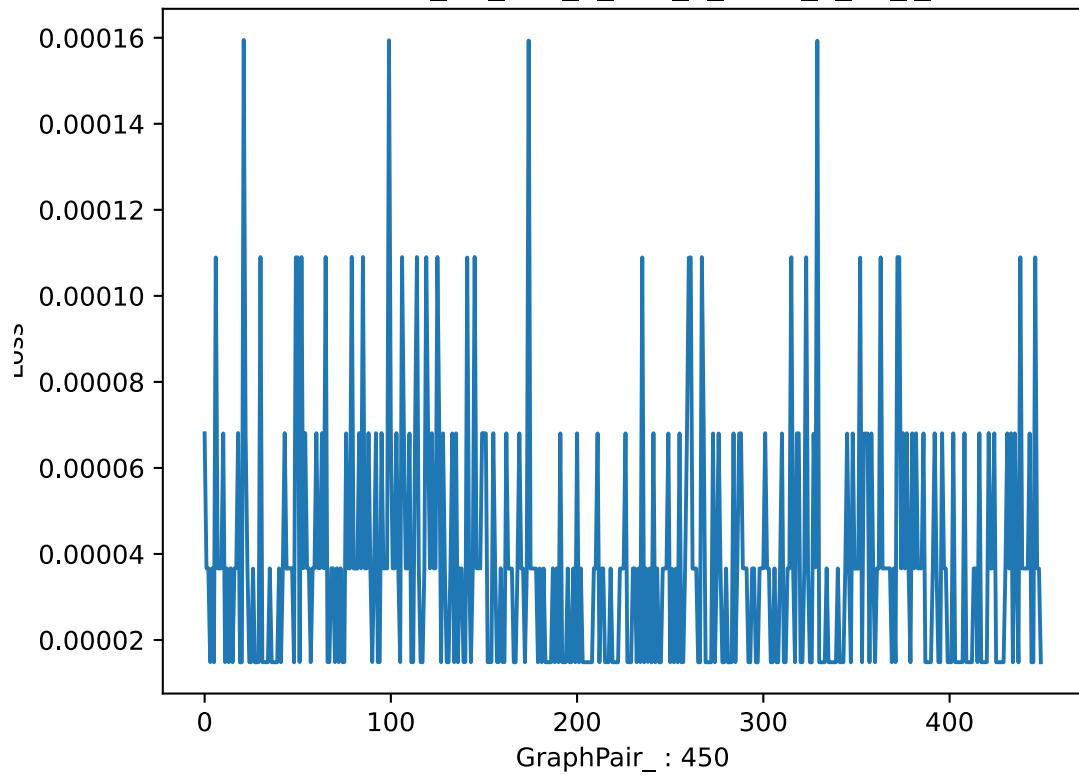
1. GP : 159 (lab)



GraphPair_159 Training Loss: epoch21 lr0.0001

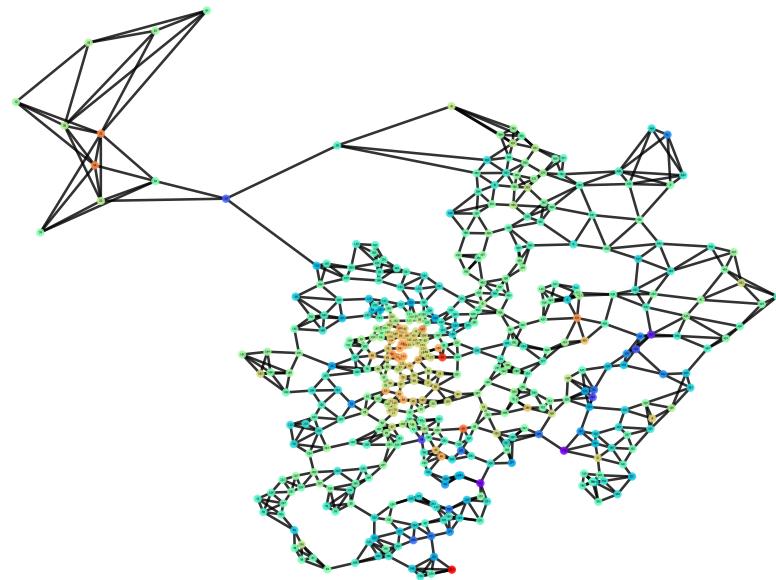


Similarity Loss_GP_449_n_450_d_2000_e_21_l_0.0001



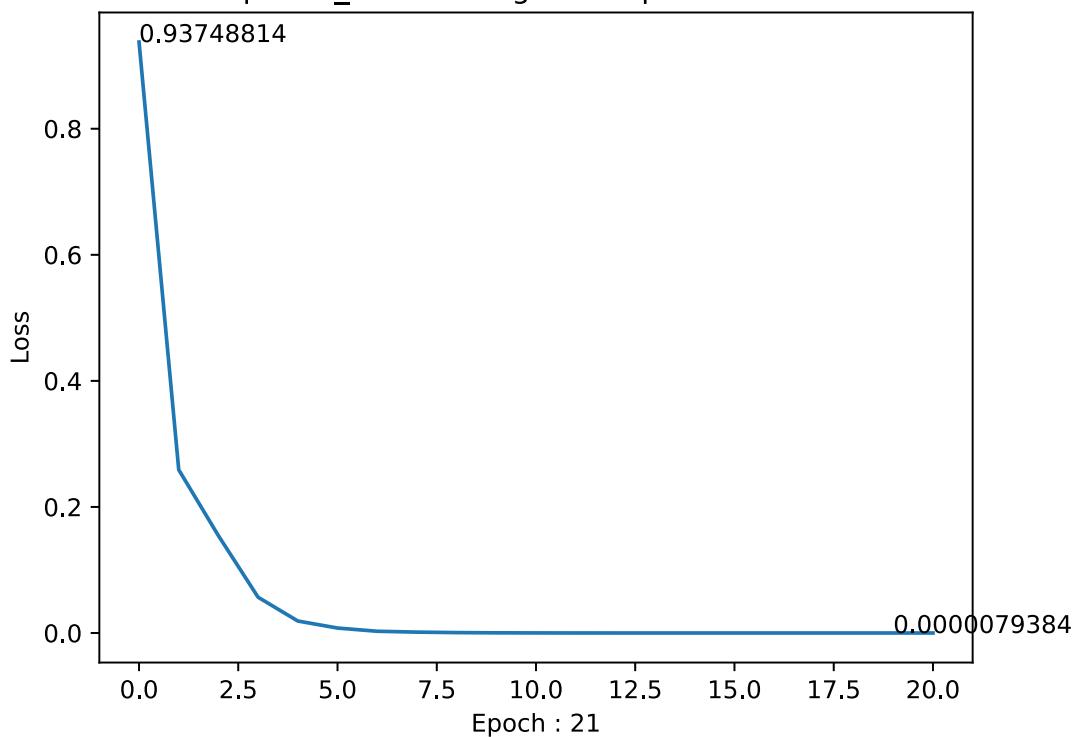
328异常: data[328] - 1.949527258053421974e-05 (借用75数据)

Graph Pair: 159

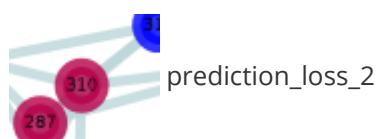


2. GP : 159 (508)

GraphPair_159 Training Loss: epoch21 lr0.0001

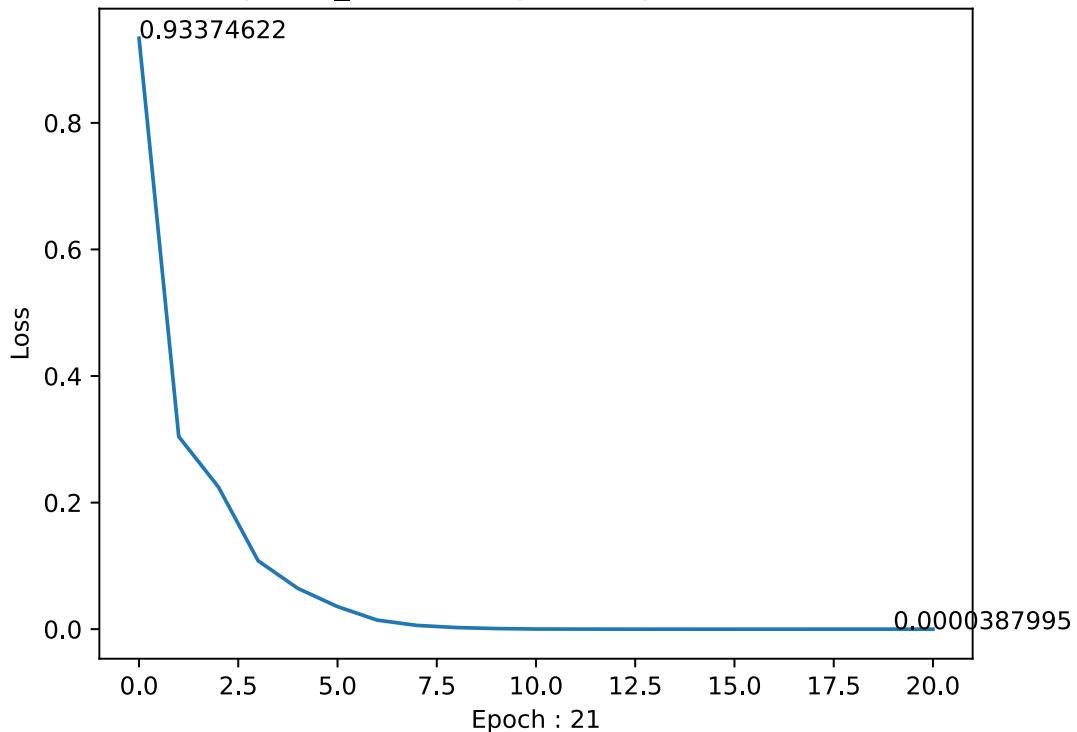


3. GP : 310 (shanshili)

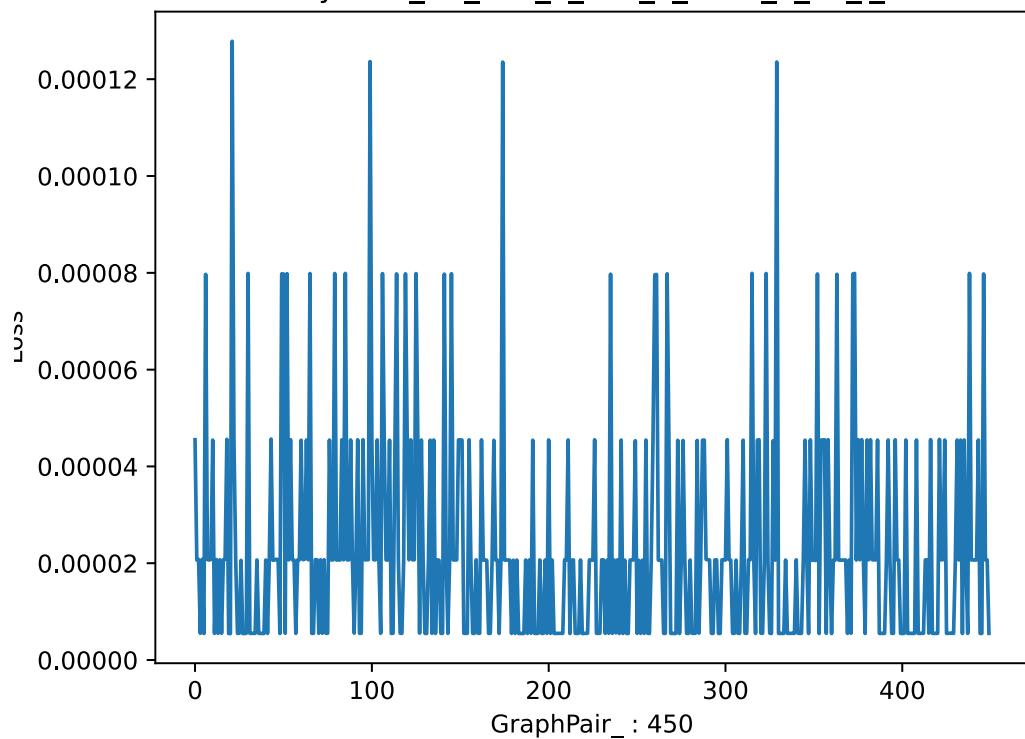


整体损失低一点

GraphPair_310 Training Loss: epoch21 lr0.0001

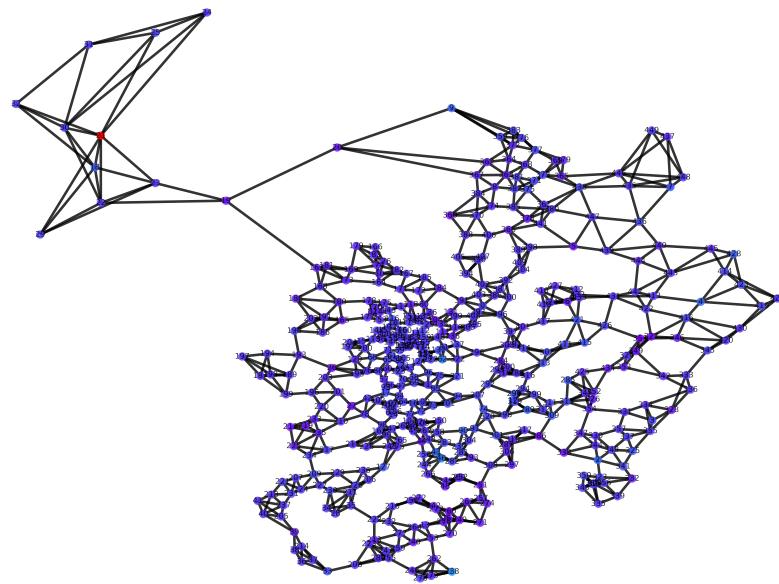


Similarity Loss_GP_449_n_450_d_2000_e_21_l_0.0001

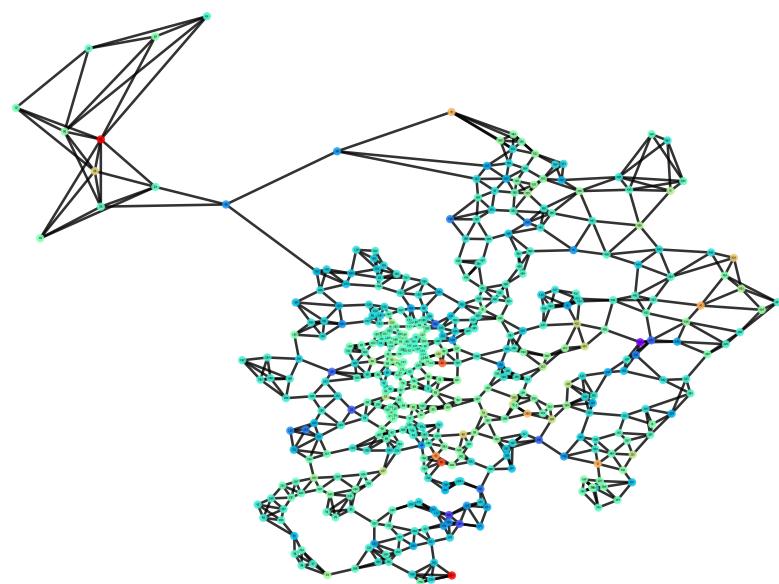


21异常 data[21] - 1.664528177166357636e-04 (调试数据)

Graph Pair: 1



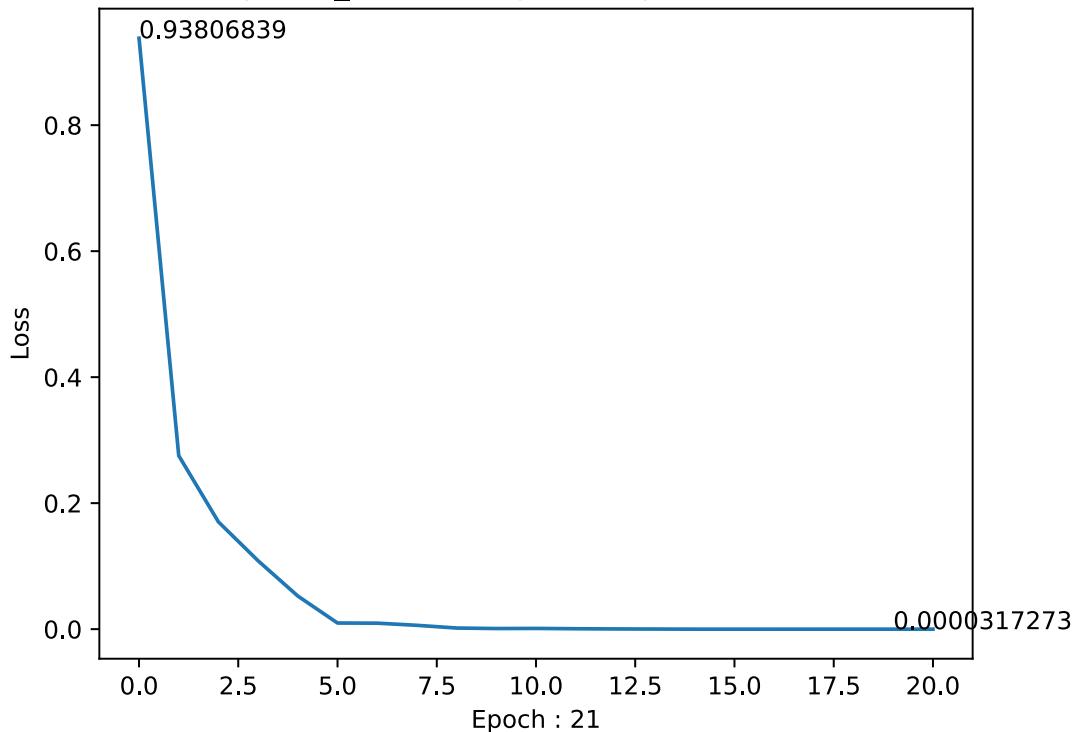
Graph Pair: 310



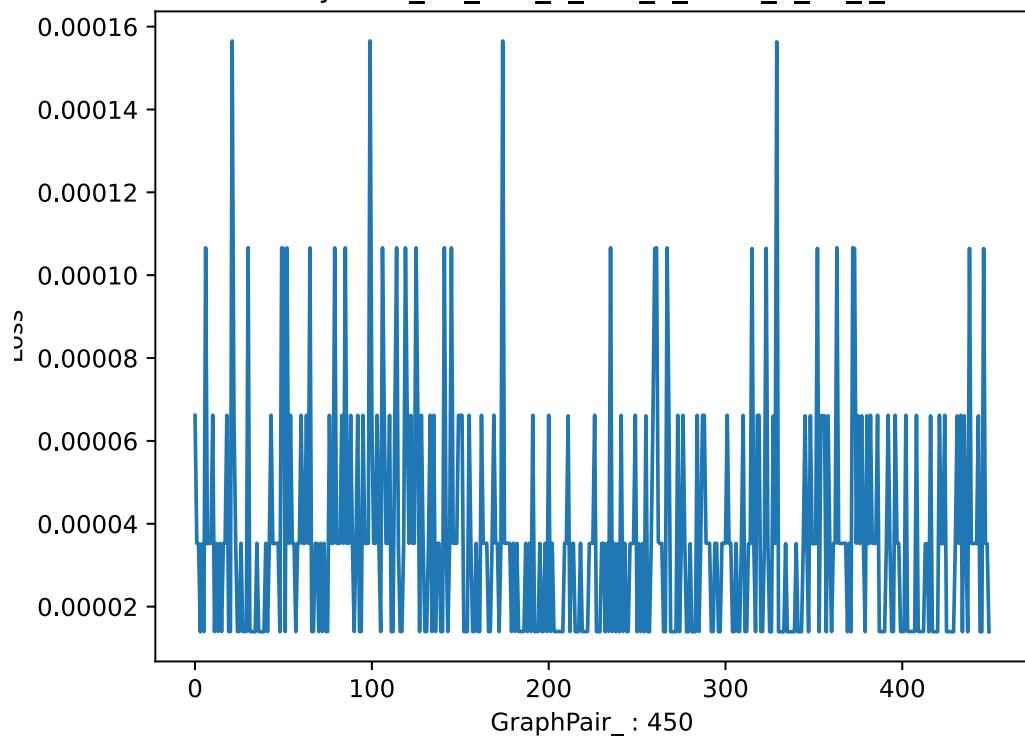
4. GP : 432 (508)

更平滑

GraphPair_432 Training Loss: epoch21 lr0.0001



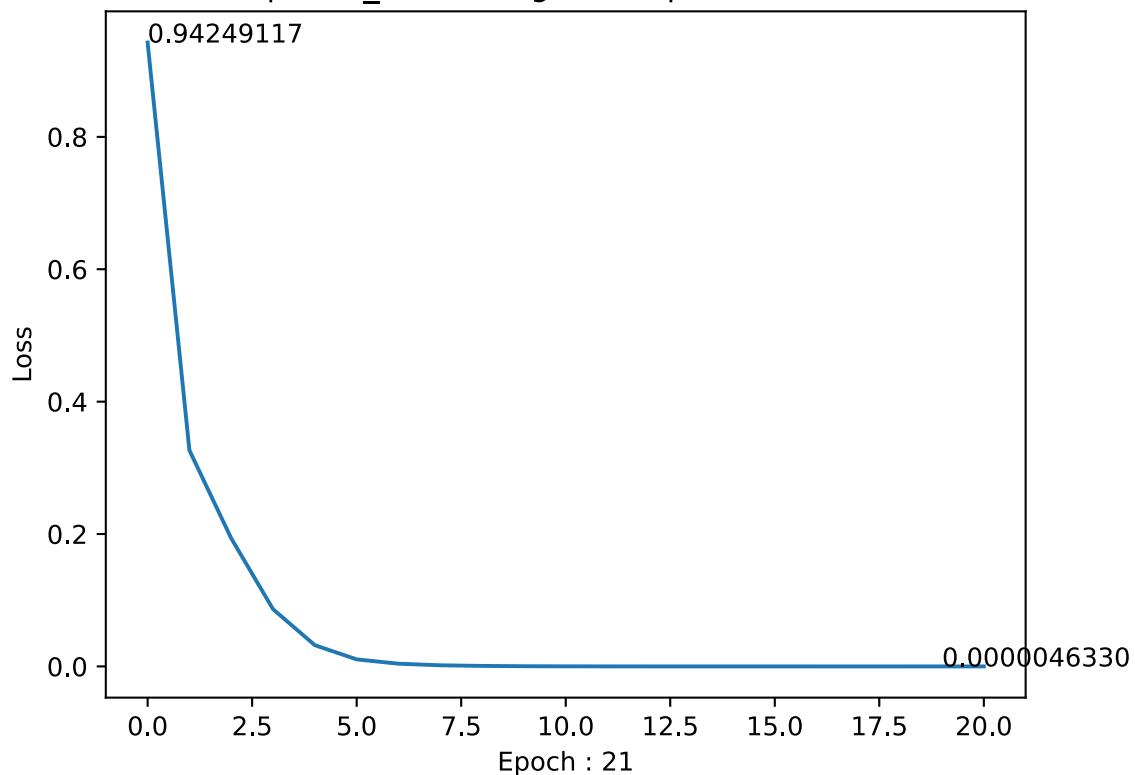
Similarity Loss_GP_449_n_450_d_2000_e_21_l_0.0001



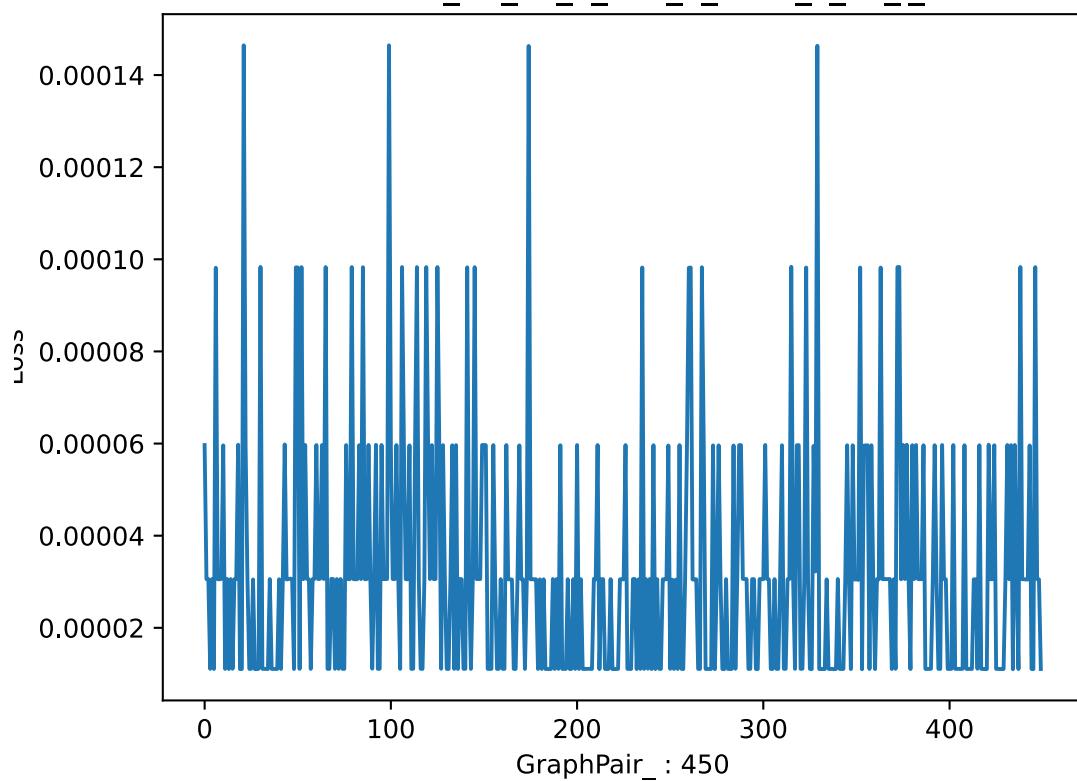
5. GP : 75 (shanshili)

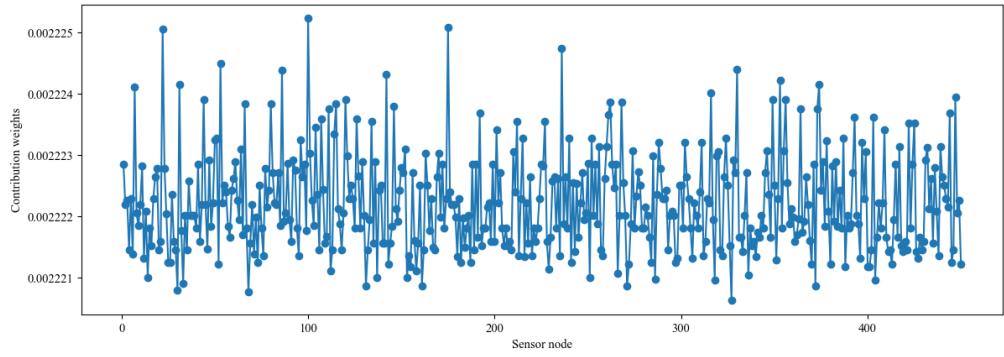


GraphPair_75 Training Loss: epoch21 lr0.0001



Prediction Loss_GP_75_n_450_d_2000_e_21_l_0.0001



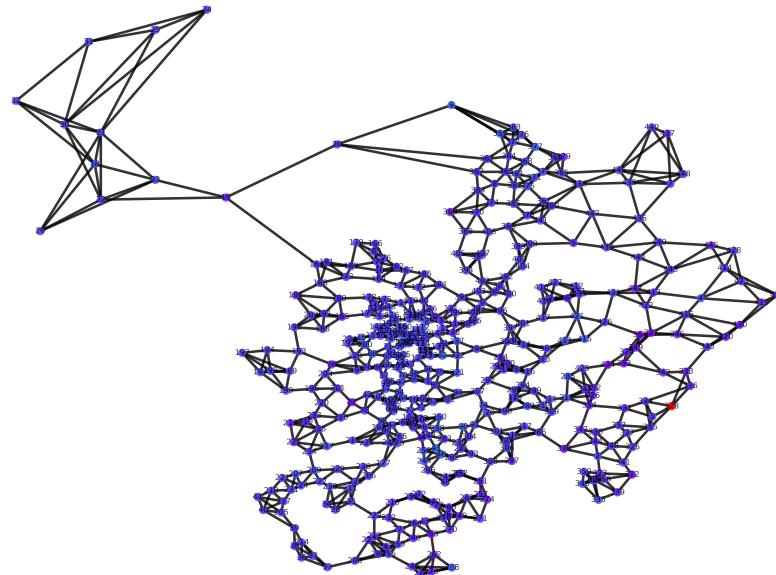


328异常 1.464528177166357636e-04 (收集数据)

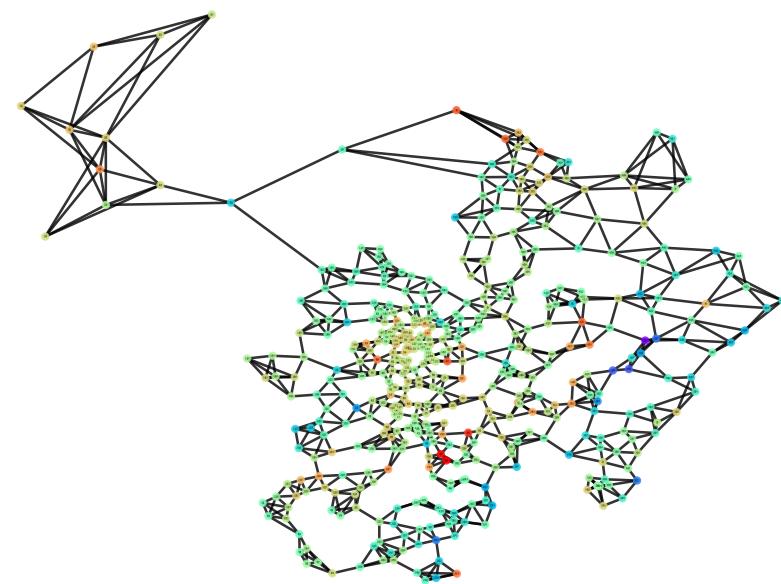
但是要用data[329]的数据修正data[328]

为什么异常点不同

Graph Pair: 2

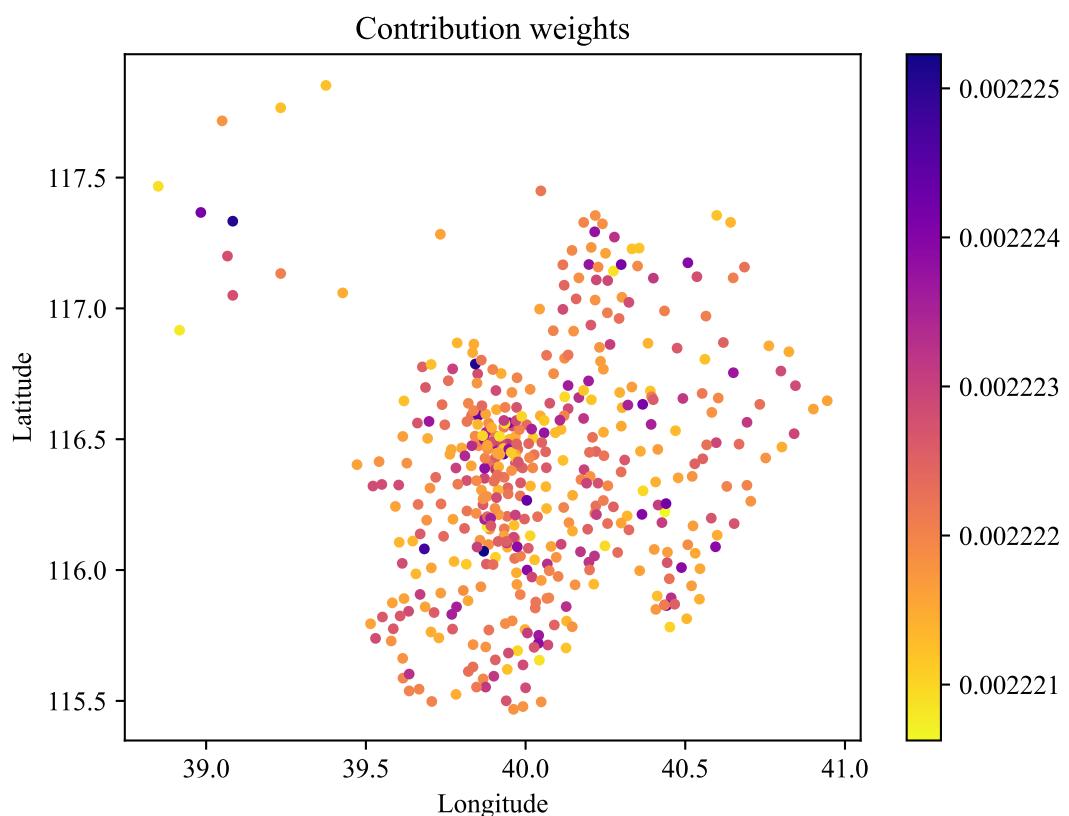


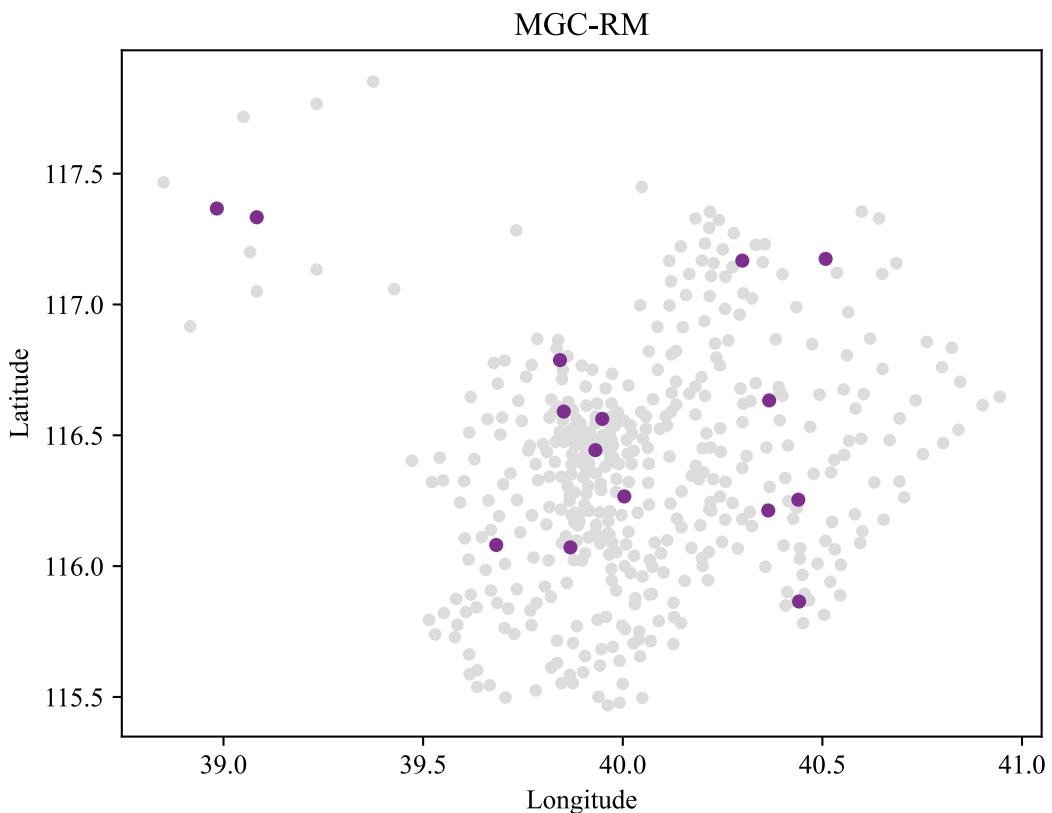
Graph Pair: 75



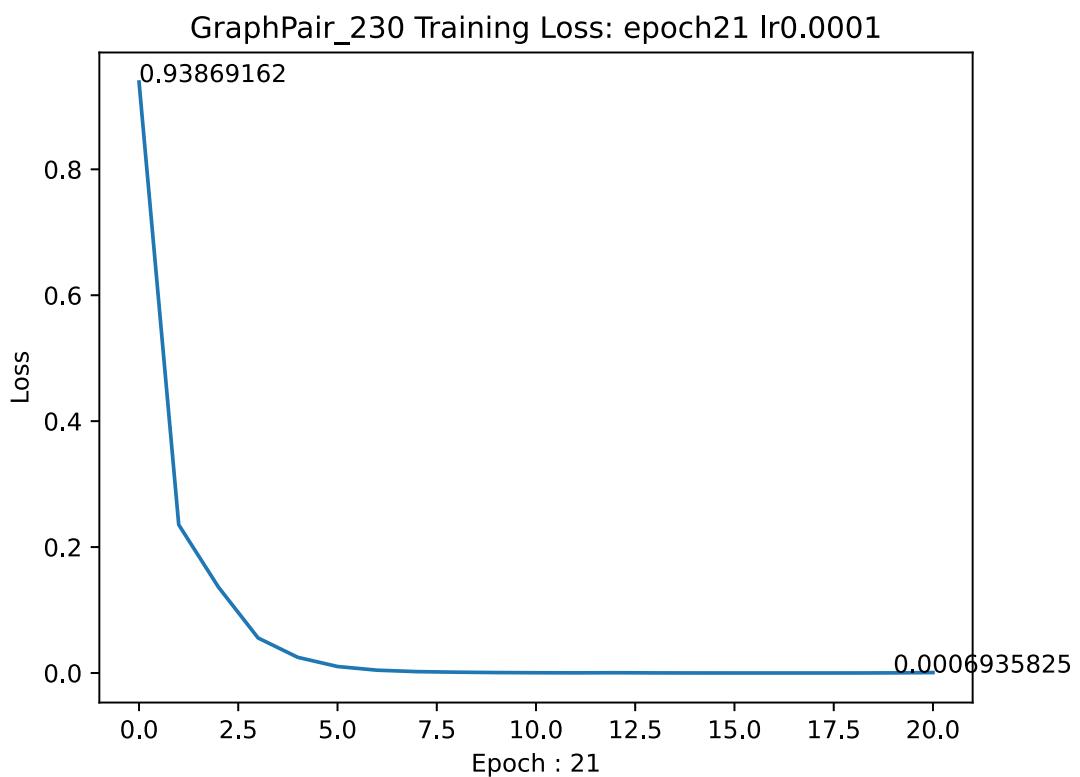
异常点不影响排序

Similarity Score—>(PageRank)—>Contribution weights





6. GP : 230 (lab)



7. GP : 100 (lab)

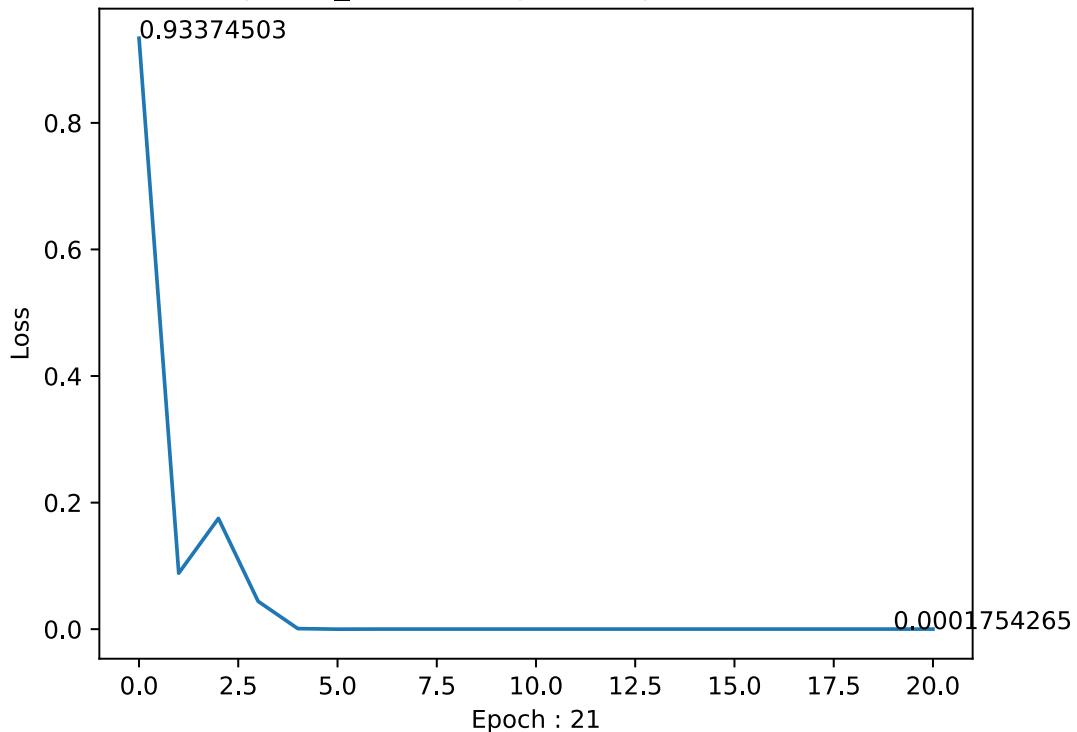
prediction loss 较大的点拿来训练似乎会出现尖刺。

但是每次会有不同。

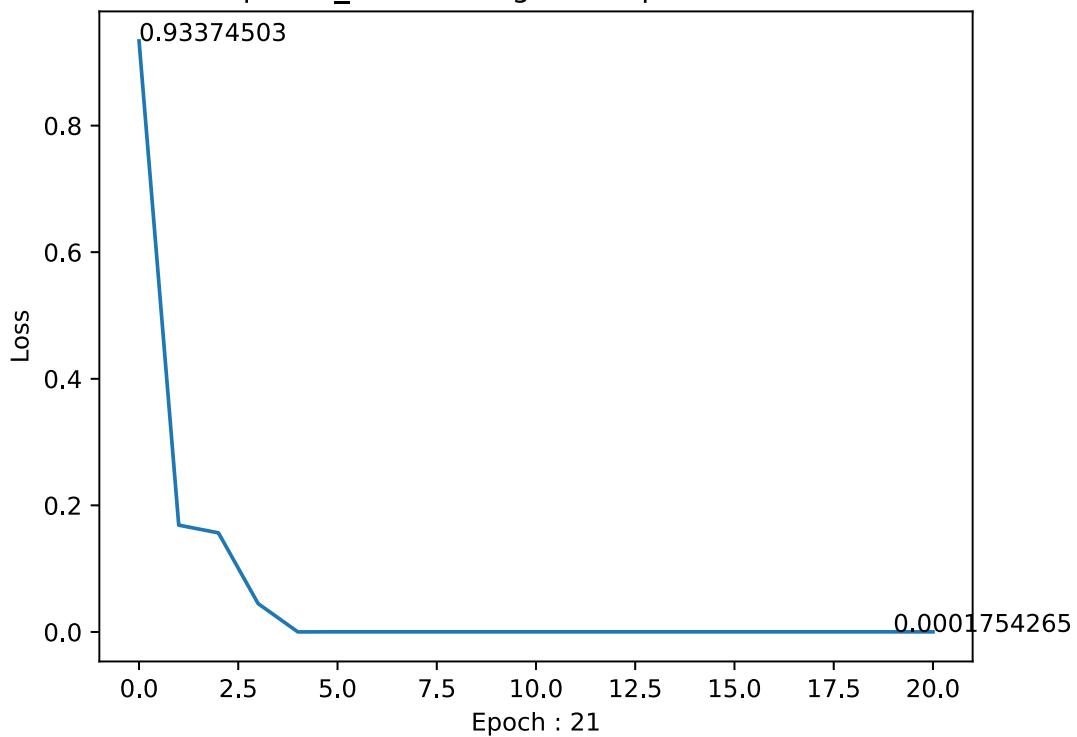
预测分数也总是1

GPU是不是需要单独设置种子,越跑越平滑了还

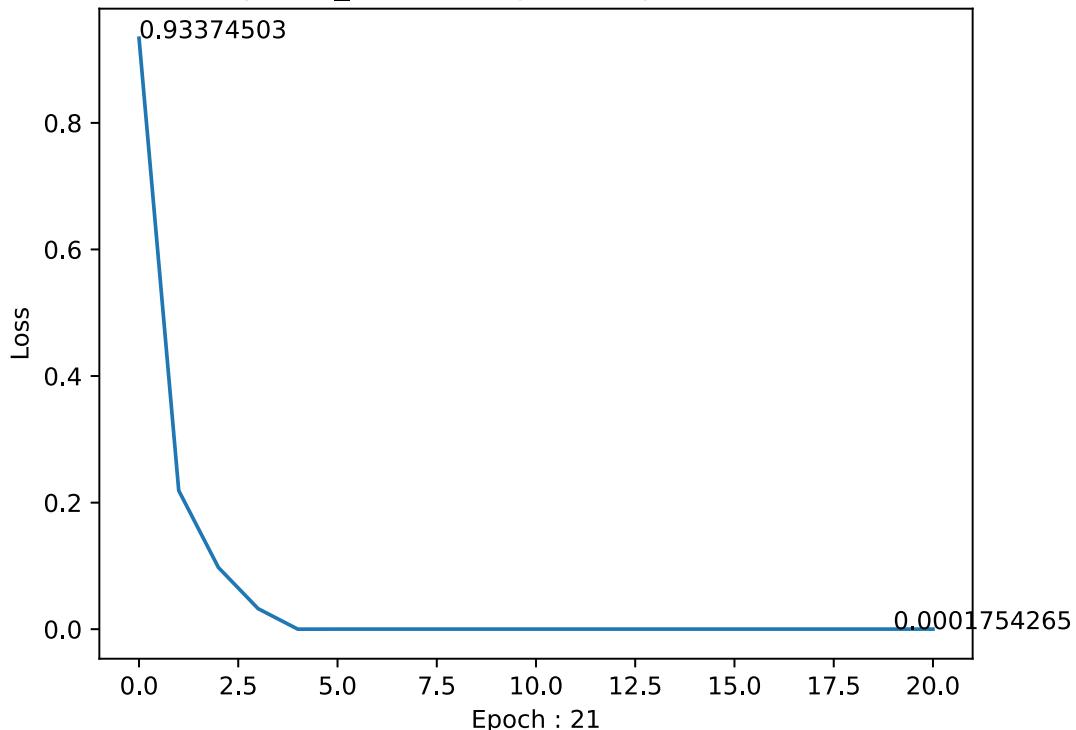
GraphPair_100 Training Loss: epoch21 lr0.0001



GraphPair_100 Training Loss: epoch21 lr0.0001



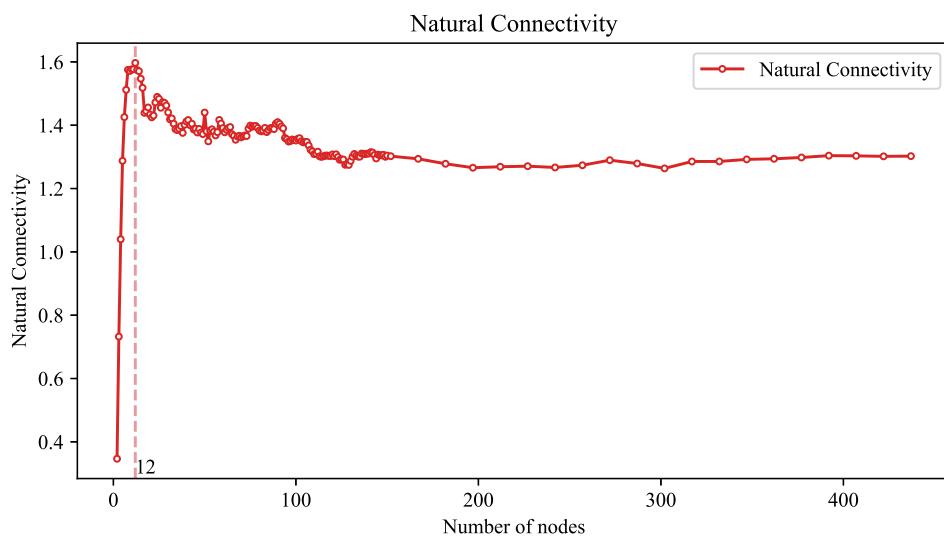
GraphPair_100 Training Loss: epoch21 lr0.0001



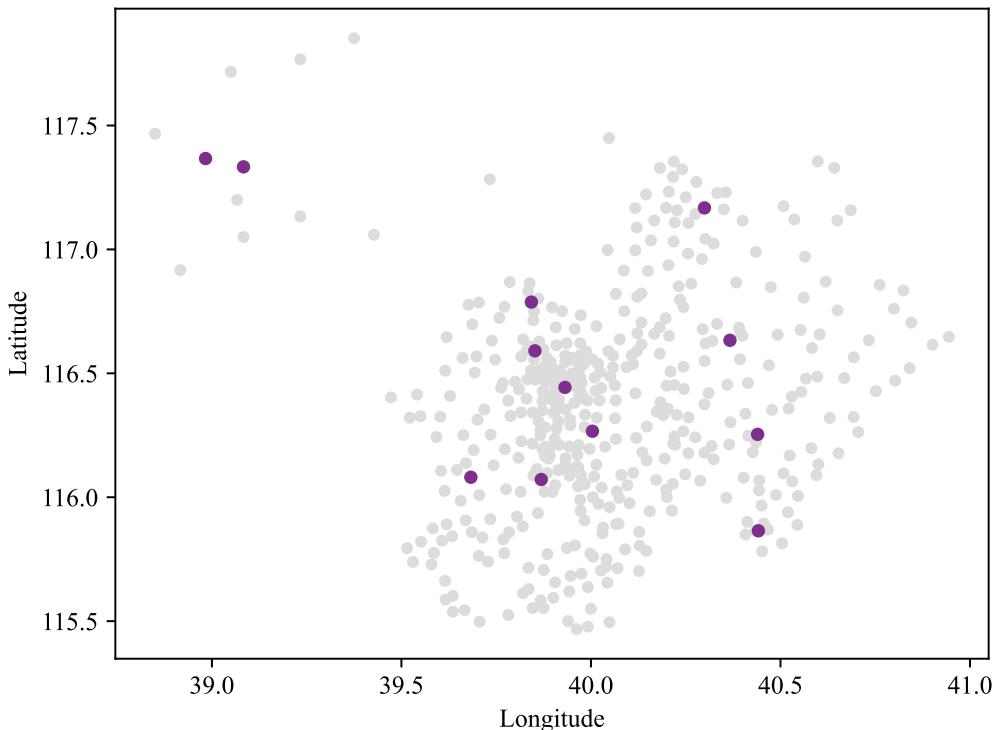
Reference indicators

1. Natural Connectivity

selectrd_node = 12



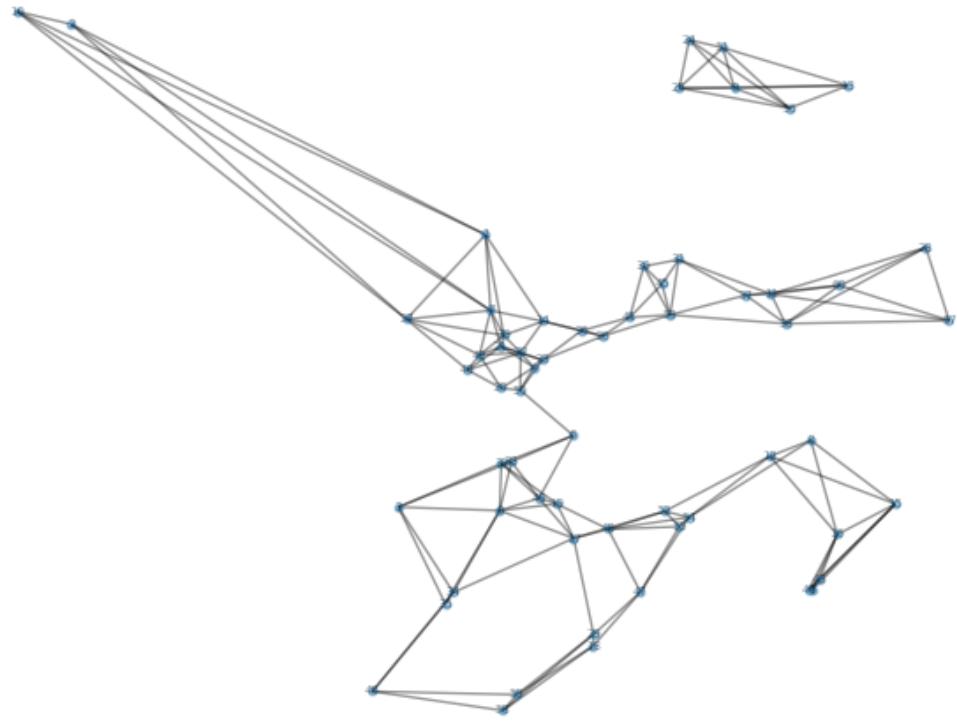
MGC-RM



LOG

241106

1. 指标的选择：多个不同维度的图进行比较
 1. 节点特征聚合为图特征进行衡量：图特征聚合器训练
 2. 比较过程中是否固定图的维度。（选出的节点重新构图/未选节点从原图中移除边、掩蔽特征）
1. 使用 NetworkX 库计算图的半径时，如果图不是连通的，`nx.radius` 函数会抛出 `NetworkXError`，因为半径的定义要求图必须是连通的。对于非连通图，你需要分别计算每个连通子图的半径。



2. 平均剩余能量是什么含义

241105

1. (1) 均方误差 (MSE): MSE 用于比较选择出的节点经重构后的特征和原始特征之间的差异。MSE 越低，说明通过所选节点重构原始信息的能力越强。

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2 \quad (3-41)$$

其中， x_i 为节点 v_i 的原始特征， \hat{x}_i 为重构特征。

MSE 衡量某个节点的预测特征与原始特征的均方误差？

怎么用来评估整个图？

对于全图而言的特征是？？拓扑特征？节点特征？

什么意思，选出来的点通过特征预测后的特征与原特征的差距？

数据重建

2.

(2) 平均能量消耗 (Average Energy Consumption, AEC): AEC 度量了传感网的总体能量消耗。AEC 越低, 说明网络中的能量效率越高, 有利于网络的可持续性。

$$AEC = \frac{1}{N} \sum_{i=1}^N \left(E_0 - L \sum_{N_i} E_T - L \sum_{N_i} E_R \right) \quad (3-42)$$

其中, E_0 为传感器节点的初始能量, L 为网络寿命, E_T 为发送能耗, E_R 为接收能耗。根据无线传感网的无线传输能耗模型 (Radio Energy Dissipation Model, REDM) [93], 每个传感器节点在单次通信中的发送能耗和接收能耗分别为:

$$E_T = k(E_{elec} + \beta d^\alpha) \quad (3-43)$$

$$E_R = kE_{elec} \quad (3-44)$$

其中, E_{elec} 为单位信息的电子损耗, k 为信息量, β 为单位信息的功放损失, α 为传播衰减指数, d 为通信距离。

3. 评价指标在第三章的三个指标的基础上, 本章从信息角度增加两个指标来评估原始网络的特征和优化后网络的特征所包含语义信息的差异: 互信息 (Mutual Information, MI) 和 KL 散度。MI 用于衡量变量之间共享信息的数量, 而 KL 则衡量概率分布之间的相似性。较高的 MI 和较低的 KL 表明优化方法在过程中有效地保留了相关信息。

$$MI = \sum_x \sum_{x'} p(X, X') \log \frac{p(X, X')}{p(X)p(X')} \quad (4-40)$$

$$KL = \sum_x p(X) \log \frac{p(X)}{p(X')} \quad (4-41)$$

其中, X 为原始网络特征, X' 为优化后网络特征, $p(\square)$ 表示特征分布密度。

4. `.twinx()` 方法用于在同一张图中绘制两个具有不同 y 轴刻度的曲线。`.twinx()` 方法会创建一个新的 Axes 对象, 共享同一个 x 轴, 但有两个不同的 y 轴。
5. `.get_label()` 方法用于获取由 `.plot()` 或其他绘图方法设置的图例标签。这些标签通常用于图例 (legend) 来标识图表中的不同数据系列。
6. `.axhline()` 方法用于在图表中绘制一条水平线。这条水平线可以用来标记特定的 y 值, 例如平均值、阈值等。
7. `.legend()` 方法用于添加图例
8. 传感器覆盖面积作为衡量?
9. KL散度里的概率分布要怎么获得:
连续特征: 对于连续特征, 可以使用直方图或核密度估计 (KDE) 来估计概率分布。
嵌入向量: 使用图嵌入方法 (如 Node2Vec、GraphSAGE 等) 生成节点的嵌入向量, 可以计算嵌入向量的分布。

```
k1_divergence = entropy(degree_freq_G1, degree_freq_G2)
```

10. 自然连通性 (Natural Connectivity) 是一种衡量图的鲁棒性的指标，它反映了图在面对节点或边的删除时保持连通性的能力。自然连通性越高，图的鲁棒性越好。
11. `find_value_according_index_list(aim_list, index_list):` # 索引转换 从排序索引找对应节点

但是aim_list 是? 节点特征, position

12. 图鲁棒性归纳学习器Inductive Learner for Graph Robustness (ILGR)
13. 需要把不同数量节点的图构建出来
14. 按照重要性排序删点及其边，但是维度不变？这样会影响比较吧，在一些孤立点上会有影响吧
15. 确保不会出现“list assignment index out of range”错误。特别是，`selected_node` 列表的初始化应该在读取文件并填充 `node_list` 之后进行，以确保 `selected_node` 有足够的空间来存储每个 `select_node` 的值。

计划

1. 指标的分析：

1. 自然连通性具体代表什么含义(√)
2. 还有哪些可用的指标，针对优化和鲁棒性

2. 指标的代码构建

3. 弹性部分的展开逻辑

241104

1. gat权值共享怎么体现

2. 马尔科夫链？

3. 1. `np.sum(A, axis=1):`

- 这个函数计算矩阵 AA 每一行的元素之和。结果是一个一维数组，其中每个元素对应于 AA 中相应行的元素总和。

2. `np.power(..., -1):`

- 这个函数将上一步得到的一维数组中的每个元素取倒数（即，对每个元素应用幂 -1）。如果某行的和为0，则该操作会导致一个除以零的错误，因此在实际使用时需要确保没有行和为0，或者适当地处理这种异常情况。

3. `np.diag(...):`

- 最后，`np.diag` 函数用于从上述步骤的结果创建一个对角矩阵。这意味着它会将输入的一维数组转换成一个二维数组，其中输入数组的元素位于矩阵的主对角线上，而所有其他位置的值都为0。

4. `np.dot(A, D)` 可以计算这两个矩阵的点积（矩阵乘法）。具体来说，这个操作将矩阵 A 与对角矩阵 D 相乘，其中 DD 的对角线元素是 AA 每一行元素之和的倒数。

5. `np.ones(N)` 是 NumPy 库中的一个函数，用于创建一个包含全1的数组。具体来说，这个函数接受一个整数 N 作为参数，并返回一个长度为 N 的一维数组，其中所有元素都是1。

6. `np.argsort` 是 NumPy 库中的一个函数，用于返回数组元素排序后的索引。具体来说，它返回一个索引数组，这些索引按照原数组中元素的排序顺序排列。

4. 相似度分数的异常值不影响排序

5. 整理师兄的评估指标代码，绘制评估图的性能

241103

早上一来，有图没数据，机器好像重启了，果然不能寄希望于没有被看着的机子

好在我电脑上的数据保住了，我的宝贝电脑此刻价值得到了极大的发挥

服务器的GPU价值好还是很大的

1. 看起来prediction 误差很小，但是similarity score之间的误差也很小，这样的误差足以变换次序了

2. 预测误差为什么是一致的，在特定点上有一致的误差

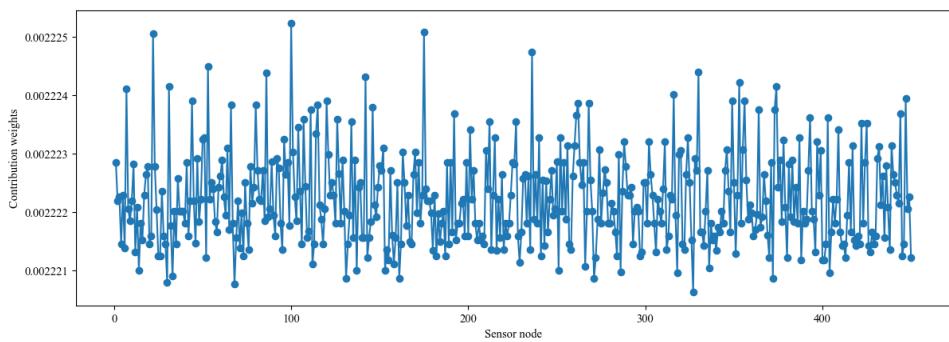
3. 328异常 1.464528177166357636e-04 (收集数据)

但是要用data[329]的数据修正data[328]

为什么不同数据，异常点不同？

4. colorbar?

5. 预测损失的规律性，与重要性得分之间存在什么关联？似乎是因此而影响的



计划

1. 细看论文，解析代码

241102

GPU确实训练部分很快！！

但是模型评估部分运行特别慢

1. cuGraph缺少wheel，安装了Docker，但不知道方法是否正确，还缺少什么环境

2. 最慢的部分不是math.exp，是NX.graph_edit_distance，但是没有找到NX在cuda的办法

3. **cuda的编程使用是个待研究的课题**

4. 每次获得全图的数据都很慢，一个小失误，一天白干。**注意小数据测试**

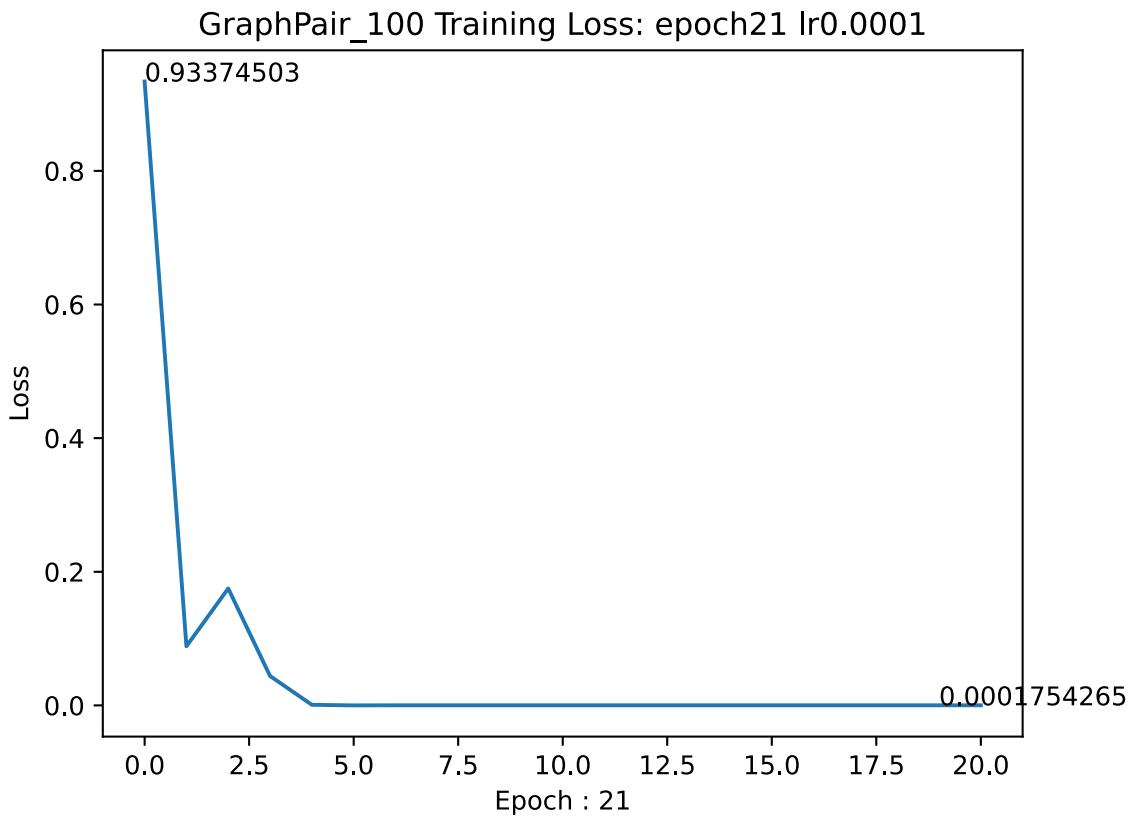
```
for i in range(perturbed_a.shape[0]-445):
```

5. 还是我的电脑跑的快一点

6. GPU同一个机器跑出来还会不同？**GPU是不是需要单独设置种子**，越跑越平滑了，但是预测分数依然总是1

prediction loss 较低的点会不会训练的模型更好？230 75

prediction loss 较大的点拿来训练似乎会出现尖刺。但是每次会有不同。**预测分数也总是1**



```

epoch:17, loss:0.00017542651039548218
z: 1.0
epoch:18, loss:0.00017542651039548218
z: 1.0
epoch:19, loss:0.00017542651039548218
z: 1.0
epoch:20, loss:0.00017542651039548218
0.000175426510395
GP: 0 =====
loss: 0.00017542651039548218
score: 1.0
GP: 1 =====
loss: 0.00012209427950438112
score: 1.0
GP: 2 =====
loss: 0.00012209427950438112
score: 1.0
GP: 3 =====

```

计划:

1. 对分数做处理，画出分数的彩色图
2. pagerank

241101

长时间休眠和睡眠，期间打开主机进行打印，然后系统崩溃，修复系统时文件损坏，不能修复，于是重装了系统。

1. 写readme文件

在服务器重新安装了pytorch，可以运行cuda，但是好像还是很慢

先用效果好的小图导出模型，计算数值往后做

241025

1. 仅用一个图对训练出模型

1. 不需要训练的数据怎么快速通过模型获得结果
2. 他的模型没有考虑每个扰动图之间的关系
3. 图对能够把所有扰动图考虑进去吗？
4. 不同重要性节点的图对训练的模型有什么不同
5. loss有尖刺，（存在离群点干扰？），是否跟设备有关？
6. 怎么调用已有的模型（√），需要forward吗，好像不用
7. 测试集对比（数据量太大，不易实现，先不管了）：

1. 某扰动图模型的泛化能力，在其他扰动图上的损失效果： **代码变量调用可能有问题**
2. **大图（效果不好，未知问题，放置）**，改变节点后模型的能力： 模型中间的层需要更改??
或者只改变节点个数，不改变特征的维数
(节点个数会导致数据量变大，但似乎不影响层数。但是改变层数后模型行不收敛了)

如果损失函数开始时几乎不变（或者说变化很小），然后逐渐开始收敛，这通常意味着以下几种情况之一：

1. 初始阶段数据稀疏或权重初始化不佳：

- 在训练初期，如果数据集很大，或者权重初始化不当（比如所有权重都初始化为零），那么模型可能需要一些时间来“热身”，即开始有效地调整其参数以适应数据。

2. 学习率设置：

- 如果使用的是一个固定的学习率（learning rate），且这个值设置得过小，那么在训练初期模型的更新可能非常缓慢，导致损失函数的变化不大。随着训练的进行，模型参数逐渐调整，损失函数开始下降并收敛。

3. 梯度消失或爆炸：

- 如果模型结构较深或者激活函数选择不当（如使用了sigmoid或tanh作为深层网络中的激活函数），可能会遇到梯度消失或梯度爆炸的问题。这会导致在训练初期权重更新非常缓慢甚至不动。解决这个问题的方法包括使用ReLU这样的激活函数，或者采用梯度裁剪等技术。

4. 局部最小值：

- 损失函数可能有多个局部最小值，在某些情况下，模型可能会陷入一个局部最小值，在这里损失函数的变化很小。随着进一步的训练，它可能会跳出这个局部最小值，并继续向全局最小值靠近。

5. 正则化效应：

- 如果使用了正则化技术（如L1或L2正则化），这些技术可以帮助避免过拟合，并可能导致损失函数在某些阶段变化较小。但是，随着训练的继续，模型会学习到更合适的参数，损失函数会继续下降。

6. 数据预处理:

- 数据预处理（如归一化、标准化）可以影响损失函数的变化速率。如果数据未正确预处理，可能会导致训练初期损失函数变化不大。

在任何情况下，重要的是要监控训练过程中的损失函数和其他相关指标，以确保模型能够有效地学习并且不会过拟合或欠拟合。如果有疑问，可以尝试调整学习率、使用不同的优化算法、改变模型架构或者增加/减少正则化强度等方法来改善训练过程。

计划

先用效果好的小图导出模型，计算数值往后做

241024

1. GAT 需要edge index ,但是edge index 不规则，不能npz
2. 构建graph pair 类
3.

```
feature_1 = (feature_1 - mean_1) / std_1
```

对每个特征进行标准化处理（也称为标准化或 z-score 标准化）是一种常用的数据预处理技术，目的是使每个特征具有零均值和单位方差。标准化处理可以改善数据的分布，从而提高机器学习模型的性能。以下是标准化处理的具体解释：

标准化处理公式

对于给定的一组数据 ($\{x_1, x_2, \dots, x_n\}$)，其标准化处理公式为：

$$[z_i = \frac{x_i - \mu}{\sigma}]$$

其中：

- (z_i) 是标准化后的数据；
- (x_i) 是原始数据；
- (μ) 是原始数据的平均值（均值）；
- (σ) 是原始数据的标准差。

标准化的目的

1. **消除量纲影响**：不同特征可能有不同的量纲或尺度，标准化后可以消除这种差异，使特征在同一尺度上进行比较。
2. **加速收敛**：在梯度下降等优化算法中，标准化后的数据可以使损失函数更快收敛。
3. **提高模型性能**：对于一些机器学习算法（如支持向量机、线性回归等），标准化后的数据可以提高模型的性能。

总结

通过上述标准化处理，每个特征的平均值变为 0，标准差变为 1。这样可以使得特征在相同的尺度上进行比较，有助于提高模型的性能和收敛速度。如果你有进一步的问题或需要更具体的解释，请随时告诉我。

4.

```
label_exp = torch.exp(-1 * label) # 无用
```

您提到的将 `label` 转换为概率值的过程可以通过指数函数 (e^{-x}) 来实现。这里我们来详细解释一下为什么这样做可以将正数转换为接近 0 的值，负数转换为接近 1 的值。

指数函数的性质

指数函数 (e^{-x}) 有一些重要的性质：

1. 当 ($x > 0$):

- (e^{-x}) 的值会小于 1，并且随着 (x) 的增大，(e^{-x}) 的值迅速趋近于 0。

2. 当 ($x < 0$):

- (e^{-x}) 的值会大于 1，并且随着 (x) 的减小，(e^{-x}) 的值迅速趋近于无穷大。
- 但是，如果我们关注的是 ($e^{-|x|}$)，那么当 ($x < 0$) 时，(e^{-x}) 会变成 (e^x)，此时 (e^x) 的值会大于 1，并且随着 (x) 的减小，(e^x) 的值会迅速趋近于 0。

3. 当 ($x = 0$):

- ($e^{-0} = e^0 = 1$)。

举例说明

假设 `label` 是一个表示得分或距离的标量或向量，我们可以通过 ($e^{-\{x\}}$) 来将其转换为概率值：

◦ 当 `label` 为正数:

- ($e^{-\text{positive number}}$) 会接近 0，表示该得分或距离较高时，转换后的值接近 0。

◦ 当 `label` 为负数:

- ($e^{-\text{negative number}}$) 会接近 1，表示该得分或距离较低时，转换后的值接近 1。

修正为概率

如果我们希望将 `label` 转换为介于 0 和 1 之间的概率值，可以使用 Sigmoid 函数 ($\sigma(x) = \frac{1}{1+e^{-x}}$)。Sigmoid 函数确保所有的值都在 0 和 1 之间：

总结

通过 (e^{-x}) 或 Sigmoid 函数 ($\frac{1}{1+e^{-x}}$)，可以将 `label` 转换为介于 0 和 1 之间的概率值。这种方法在很多场景中都非常有用，特别是在需要将得分或距离转换为概率值的情况下。如果你有具体的上下文或应用场景，请提供更多信息，以便我能给出更具体的解释和建议。

5. 粗心大意，输错矩阵，不是大小对不上GAT，是维度

6. 如果用一个图对训练出模型，需要都训练一遍吗？(√) 还是后面的其实不用

1. 不需要训练的数据怎么快速通过模型获得结果
2. 他的模型没有考虑每个扰动图之间的关系？？
3. 还是说图对能够把所有扰动图考虑进去

7. 测试集对比一下吧 (先不管了)

8. 某个图对的loss有尖刺，什么原理，是否跟设备有关？？

241023※

1. 每个图对都分别计算相似度分数，计算N个？

与AI交流<https://lxblog.com/qianwen/share?shareId=d1f6fc34-953b-4241-941d-a260f037fbba>

1. N个扰动图怎么凑图对：尽管存在N个扰动图，但每个扰动图都是独立地与基准图形成图对并通过相同的流程进行处理。
2. GAT中未扰动的节点也要学习表示？：GAT（图注意力网络）来学习节点表示时，即使某些节点本身没有改变，它们也会参与到其他节点特征的学习过程中，因为GAT是基于节点之间的相互作用来进行特征提取的。

此外，扰动图的设计目的是为了评估每个节点在网络中的贡献。通过对比扰动前后网络的变化情况，可以更好地理解每个节点的重要性。即使有些节点没有被直接扰动，它们在扰动图中的表现也会受到周围被扰动节点的影响，从而间接反映出它们在整个网络中的地位和功能。

综上所述，即使是那些在扰动图中没有直接发生改变的节点，它们的处理也不是多余的。

3. "图间"是？：相似度系数 β_{mj} 和 β_{in} 是用来衡量基准图 G_c 中节点 v_m 与扰动图 G_o 中节点 v_m^c 之间的相似度，以及基准图 G_c 中节点 v_i 与扰动图 G_o 中节点 v_i^c 之间的相似度。相似度系数是通过计算节点间表示向量的余弦相似度来获得的。
4. "图间表示"的加权：图间表示是通过加权获得的原因在于它旨在捕捉两个图之间节点特征的关联性，并以此来反映节点在不同图中的相似性和重要性。这里所谓的“加权”，实际上是对节点在基准图和扰动图中表示的融合过程。
5. 加权的N：在相似度系数计算过程中，分母中的 NN 指的是图对中的另一个图中的节点数量。
6. 学习一个节点的图间表示，为什么 h_m^* 和 h_i^* 要计算2边：这种做法的目的在于评估节点在不同图结构下的表现一致性或差异性。

7. 全局读出：获得图的表示

基准图和扰动图的节点级图内（没有考虑彼此的相似度）和图间表示（考虑彼此的相似度）

8. 交叉匹配：图对相似，则图对的图内表示和图间表示之间的差异很小

图间表示：假如图对相似，他们的图内表示相似，求的余弦相似度是这个点和其它节点的，**如果笼统的看作是节点在图中的表现的话肯定是相似，但具体而言呢？**相似度系数肯定笼统而言，是相近的，加权后的表示也是。

9. 越重要，相似度越低

2. pycharm更新导致代码不能运行？？？

退回版本后可以了

```
from . import _csparssetools
ImportError: DLL load failed while importing _csparssetools: 动态链接库(DLL)初始化例程失败。
```

3. 写成函数似乎运行速度会变慢？？（通过计时，发现不会）

4. MGC的数据读取部分，以及模型函数的调用部分需要重构

241022

最近一直在忙汇报和实验课

关于扰动图，因为要考虑每个节点的重要性，所以每个节点都有一个扰动图



扰动确实是只扰动一个节点

GAT

那我GAT只需要对这一个节点做？（对每个节点都做，虽然某些节点没有改变，但是会参与到其他节点的学习中去，扰动图设计的目的是为了评估每个节点在网络中的贡献，虽然某些节点没有直接被扰动，它们在扰动图中的表现也会受到周围被扰动节点的影响，从而间接反映出他们在整个网络中的地位和功能）

本章均采用均方误差作为损失函数

采用重构损失？常见的重构损失包括均方误差（MSE）或交叉熵损失

加权求和（相似度系数）

1. 用相似度系数加权，是在和谁加权，N是谁？？总节点？？邻居？

GAT中已经考虑了邻居，图间表示是哪一步抽象？

不同图之间加权，对N个扰动图进行加权？

~~每个节点的扰动特性来源于N个扰动图中分别计算出的每个节点的表示，~~

~~可是，这样的话，大部分节点的表示都是相同的，每个扰动图只有一个节点只差~~

原图和扰动图分别图间表示

2. 相似度系数，是谁和谁相似，该节点在每个扰动图之间的表示

(图间表示没有搞懂，从图级表示入手倒推理解一下)

241017

文件调用

内存不够了，内存映射，但是好像没有效果

只扰动了一个点，某个点都一个扰动图

图对是每个节点都有一对吗

两个图分别GAT学习图表示？ G_c 有很多个图，怎么学习图表示

用传感器测量值作为节点特征这合理吗？？一段时间的温度数据是这个点的气候变化特征，对这个特征进行抽象，有什么意义吗

数据的特征，是该数据的抽象，表征，类似label，但不是准确的label

241016

重新下载数据集

整理报告的论文

241015

写实验报告

241014

看了看课程汇报的素材。但是没有定下题目。

241013

完成了无线投屏，确实略有延迟，但是可用

241012

配置了SSH的基本连接

pycharm需要专业版才能ssh连接

无线显示器方案