模式识别与机器学习 Project - AI Cures

杜逸闲 17300240036

宁晨然 17307130178

本项目代码 已上传至Github, 其中核心代码为main, utils和configure.

依赖库：

1. torch & torchvision
2. rdkit
3. dgl
4. dgllife
5. matplotlib
6. sklearn

问题描述

给出若干种化合物的SMILES表示，要求判断其是否对Covid-19具有抗性，输出为0或者1，是一个2分类问题。

化合物的SMILES表示

SMILES (implified molecular input line entry specification), 是一种用ASCII字符串描述分子结构的规范，语法如下：

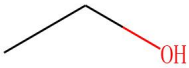
1. 原子用方括号内的化学元素表示，可以省略，如`[Au]`表示金原子。在有机物中只有C、N、O、P、S、Br、Cl、I可以省略方括号。
2. 氢原子忽略不写，当化合价不足时用氢原子补足。如水的SMILES是`O`。
3. 原子间双键用'='表示，三键用'#'表示。
4. 如果结构中有环，将环断开并在断开的两个原子旁用同一个数字标记，表示原子间有键相连。
5. 芳香环中的C、O、S、N原子分别用小写字母c, o, s, n表示。
6. 分支结构用圆括号表示。

SMILES的局限性在于没法很好的表示分子的空间排列。

下面是一些实例：

1. 水：O

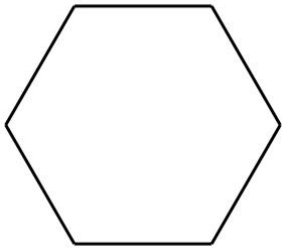
2. 乙醇：CCO



3. 二氧化碳：O=C=O

4. 氰化氢：C#N

5. 环已烷：C1CCCCC1

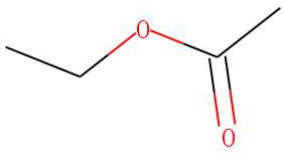


6. 乙烷：CC

7. 乙烯：C=C

8. 乙炔：C#C

9. 乙酸乙酯：CC(=O)OCC



数据集获取

在官方论坛填写自己的姓名和邮箱可以获取到数据本次Benchmark的数据，也可以在数据页面下载其他数据集。

流程图

本次实验的整个处理流程如下。以最终采用的MPNN模型为例。

评价标准

了解数据评价标准可以帮助我们更好的评价自己的模型。该任务的评价标准有四个，见Task页面下的排名.

10-fold CV ROC-AUC	10-fold CV PRC-AUC	Test ROC-AUC	Test PRC-AUC

交叉验证

由于该数据集的规模较小，为了防止过拟合，数据集将被平分为10份。每次训练我们选取其中的9份作为训练集，剩下的1份作为验证集。上图的10-fold CV (cross validation)前缀就代表交叉验证。

ROC-AUC && PRC-AUC

对于交叉验证和总测试集，我们都有两个指标，ROC-AUC和PRC-AUC, 他们均能一定程度上反应分类器的性能，但在本项目中PRC-AUC会用作分类器排序的第一参数。我们将分别介绍ROC-AUC和PRC-AUC并分析为何PRC-AUC在此项目中更为重要。

在一个二分类任务中，对于一个分类器的预测结果，我们有如下的定义：

1. TP, True Positive为真正例（测试集标记为正，分类器预测为正）的数量。

2. FP, False Positive为假正例（测试集标记为负，而分类器预测为正）的数量。
3. FN, False Negative为假负例（测试集标记为正，分类器预测为负）的数量。
4. TN, True Negative为真负例（测试集标记为负，分类器预测为负）的数量。

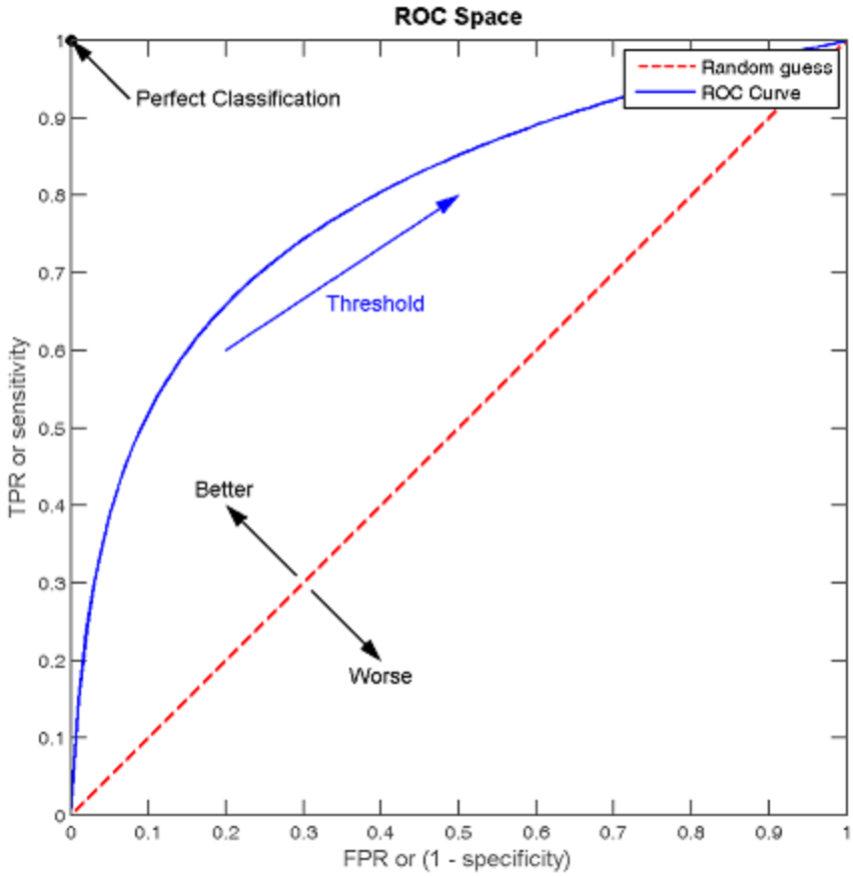
ROC曲线

在了解ROC (Receiver Operating Characteristic) 曲线之前，我们定义两个指标：

1. 真正例率 (True Positive Rate, TPR), 表示所有正例中，预测为正例的比例：
   $$TPR = \frac{TP}{TP + FN}$$
2. 假正例率 (False, Positive Rate, FPR), 表示所有负例中，预测为正例的比例：
   $$FPR = \frac{FP}{TN + FP}$$

在一个二分任务里，很多分类器输出一个实数值或者一个概率，接着设定一个阈值，当高于这个阈值时标记为正例，反之标记为负例。

因此，当改变这个阈值的时候，TPR和FPR也会相应的改变。我们以FPR为横坐标，TPR为纵坐标，当阈值改变时就可以得到一系列的点（很显然这些点关于阈值的函数是“连续”的），我们将这些点连接起来平滑处理就可以得到ROC曲线了。下图中蓝色实现即为某分类器的ROC曲线，而红色虚线为随机情况下的ROC曲线。



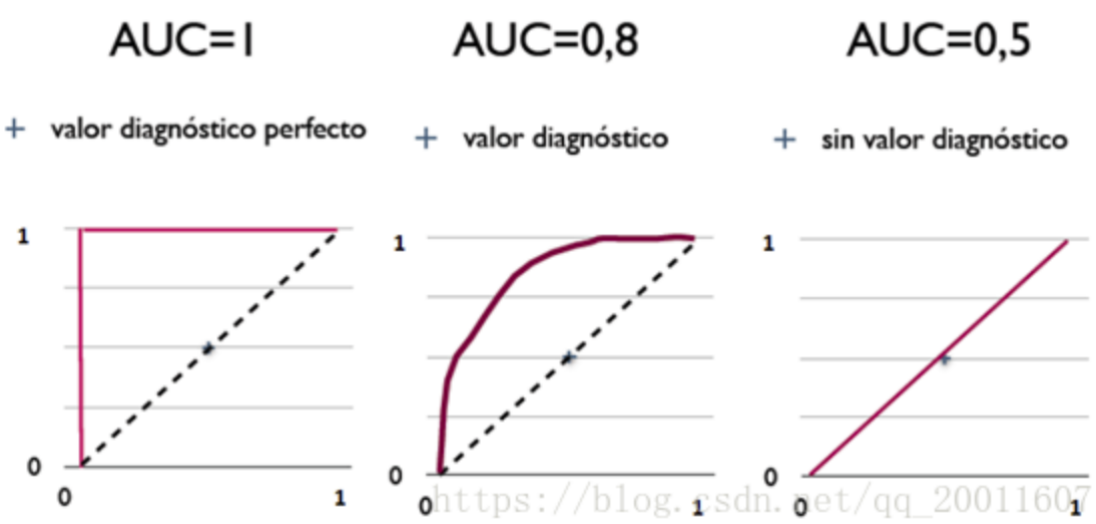
ROC-AUC

AUC (Area Under Curve) 就是曲线下的面积。该值的实际上代表了一个概率：随机选择一个正例和一个负例，该分类器对于正例的输出比负例高的概率（之前说过分类器输出一个实数值）。那么ROC-AUC越大，代表分类器越有对于正例输出较大的值，即能够更好的分类。

用以下标准可以大概通过AUC预估分类器的性能:

1. $AUC = 1$, 该分类器是完美的，因为其ROC曲线是一条经过(0, 1)的折线，一定存在一个分类器输出的阈值使得，$TPR = 1, FPR = 0$, 即正确率为100%.
2. $0.5 < AUC < 1$, 该分类器有一定价值，选取合适阈值的话可以得到优于随即猜测的结果。

3. $AUC = 0.5$, 该分类器效果基本等于随机，随机分类的ROC曲线为一条直线。
4. $AUC < 0.5$, 该分类器劣于随机，需要优化或者找出错误时，但将输出结果取反可以得到由于随机的结果。

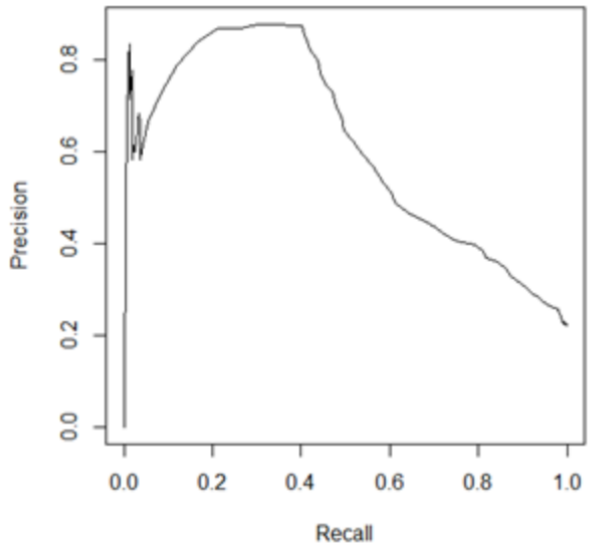


PR曲线

在了解PRC (Precision-Recall Curve)之前，我们同样定义两个指标：

1. 查准率 (Precision), 表示所有分类器预测的正例中，真正的正例的比例：
   $$Precision = \frac{TP}{TP + FP}$$
2. 查全率 (Recall), 表示所有正例中，预测为正例的比例：
   $$Recall = \frac{TP}{TP + FN}$$

与ROC曲线类似，以$Recall$作为横坐标, $Precision$作为纵坐标，改变阈值后得到的点连线后得到一条曲线（折线）。下图就是一个PR曲线的例子：



同样PRC-AUC也是指PRC曲线下方的面积，取值范围在[0, 1]之间，越大越好。

PRC-AUC的优势

PR和ROC在面对一个正负样本不均衡的数据集时的表现是不同的。在数据不平衡时，PR曲线是很敏感的，随着正负样本比例的变化，PR曲线会发生强烈的变化；而ROC曲线是不敏感的，其曲线基本不变。

因此ROC面对不平衡数据的一直表现表明其可以衡量一个模型本身的预测能力，而这个预测能力与样本正负比例无关。而PR曲线因为对样本比较敏感，因此在样本比例发生变化时，可以看出分类器对应的效果。

在实际数据中，样本比例经常是不平衡的，即正例远多于负例或者反之。因此PR曲线更有助于了解分类器实际的效果和作用，以此来改进模型。

而在本数据集下，正负样本数量相差悬殊，负例远远多于正例。在总训练集`train.csv`中，正例只有48个，而我们总共有2097个例子！在这种情况下，哪怕是一个非常差劲的模型，ROC的效果看上去依然不错，但是PR上效果则一般。这说明当正负样本比例悬殊较大时，PR曲线比ROC曲线更能反映分类器的性能。

因此在本项目中采用PRC-AUC作为第一评测指标。

工具

RDKit

RDKit是一个强大的开源化学信息python工具包，其和机器学习的联系紧密，可以方便的将分子模型转化成图神经网络所需的格式，同时可以生成用于机器学习的分子描述符。由于其核心数据结构和算法均由C++实现，效率比较令人满意。

RDKit可以很方便的将分子的SMILES串转化成分子模型并以多种格式输出，如下代码可以读入SMILES串并保存为图片格式：

```
SMILES = "C1CCCCC1"
Mol = Chem.MolFromSmiles(SMILES)
Draw.MolToImageFile(Mol, "环已烷.jpg")
```

DGL (Deep Graph Library)

DGL是一款开源的专门面向图神经网络的框架，是基于Rdkit和Pytorch或Mxnet框架上的Python框架。因此习惯于Pytorch的用户可以快速上手DGL, 也可以在DGL的框架内用Pytorch自定义很多功能，如提取特征等。

DGL-LifeSci

DGL-LifeSci是DGL的一个子包，专门用于将图神经网络应用于化学和生物领域的多种任务中，其基于Pytorch和DGL, 提供如下功能：

1. 各种用于数据处理，模型训练和模型评估的工具，比如内置的各种对于点和边的特征提取器(Featurizer), 将分子SMILES串直接转化成图的工具、自定义损失函数、自动计算多种机器学习评判标准如mae, roc_auc等。
2. 可以灵活自定义的模型，如GCN, MGCN, GAT, MPNN, SchNet等。
3. 已经预训练好的可以直接使用的模型。

本项目中代码依赖RDKit, Pytorch, DGL, DGL-LifeSci包实现。

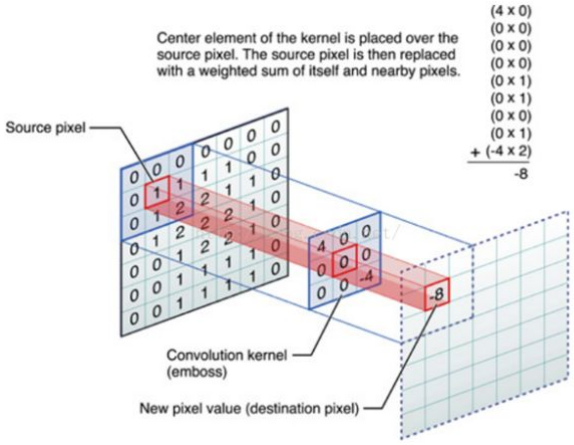
模型

GCN (Graph Convolutional Network)

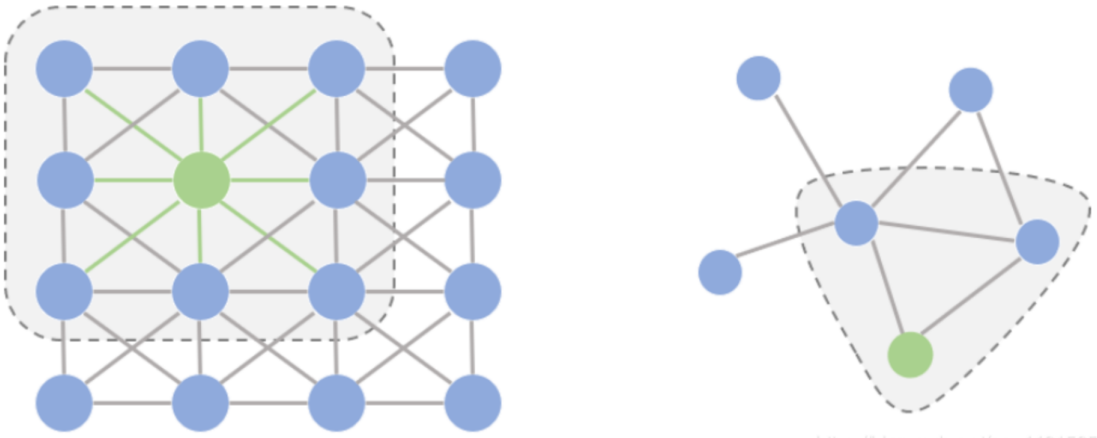
在本项目中，分子的数据结构是图。不同于在CV等领域的图等输入数据，分子结构图属于拓扑结构。因此，图神经网络用来处理传统神经网络如RNN和CNN无法处理的具有复杂结构的数据类型。

图神经网络的应用范围和前景广阔，如对已知分子结构的化合物判断其性质，对物理模型进行建模，分析社交网络甚至处理在CV和NLP等领域也存在的拓扑关系。

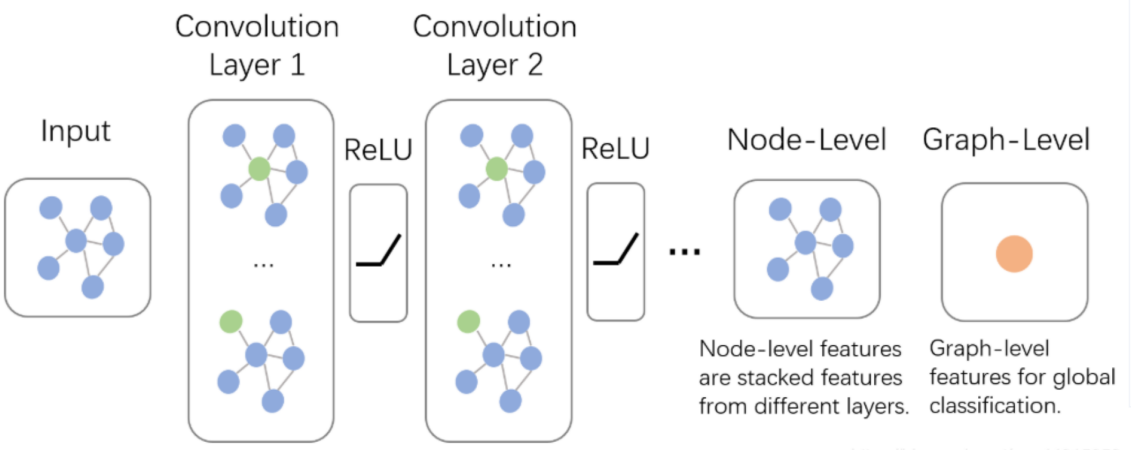
而卷积操作作为效果显著的局部特征提取手段，在传统神经网络中得到了大量的应用，得到了卷积神经网络。



注意在传统的卷积神经网络，如上图中，一个点总是从他和他相邻的一共九个点中学习信息至下一层，这要求每个节点的相邻节点个数是固定的（边界可填充空节点）。但是在拓扑结构中，这样的前提并不能总是被满足，某些点可能会有很多邻居而有些点可能只有很少邻居。



那么构建图卷积网络的思路就是要找到适用于图数据结构特征的可学习的卷积核。而构建图卷积网络的框架并没有变，同样是将网络分层，层于层之间用卷积核来传递信息。



在第一层网络中，对每个节点的另据都进行一次卷积操作，并用结果经过激活函数（下图中为ReLU）更新下一层的该节点，反复次数就是该CGN的层数。和GNN类似，GCN最后一层后也有一个输出函数，用于将整张图的状态转化成标签，比如本项目中的活跃度二分类。

而卷积实现的方法多种多样，一个最简单的聚合邻居节点信息的卷积方式是：将所有相邻邻居节点的隐藏状态直接求和，来更新下一层当前节点的隐藏状态。

$$h_v^{l + 1} = \sum_{u \in ne[v]} h_u^l$$

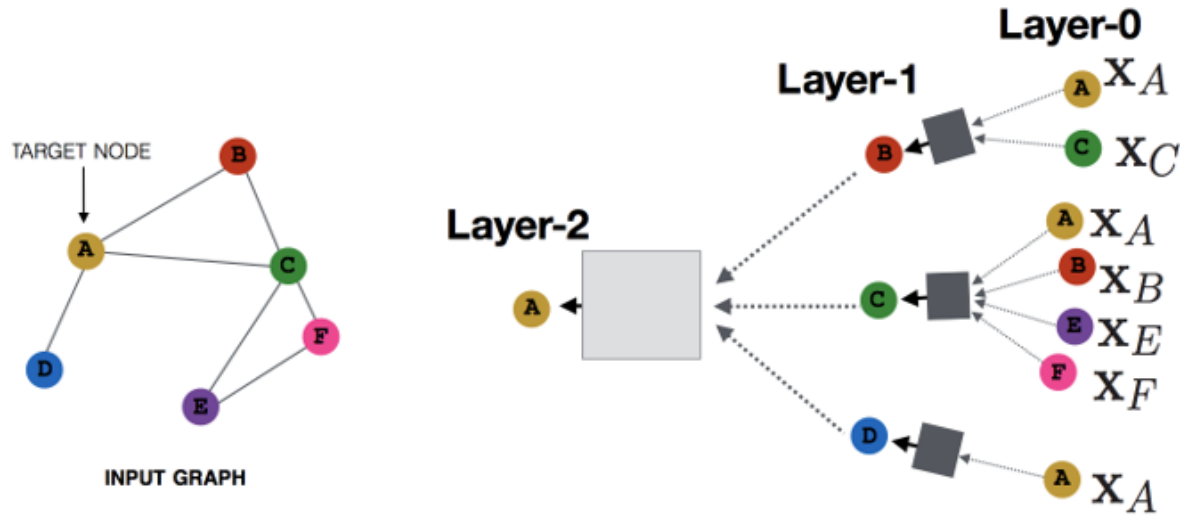
其中$h_v^{l + 1}$代表当前节点v在下一层$l + 1$的隐藏状态，$ne[v]$是v的所有相邻节点组成的集合，h_u^l是v的邻居u在当前层l的隐藏状态。

MPNN (Message Passing Neural Network)

MPNN是由Google提出的一种GCN框架，他将上面提到的卷积步骤分为两个过程：**消息传递**和**状态更新**，定义为两个函数M_l和U_l. 将节点v的特征直接作为初始状态h_v^0，隐藏状态的更新公式如下：

$$h_v^{l + 1} = U{*l + 1*}(h_v, \sum{u \in ne[v]} M{*l + 1*}(h_v^l, h_u^l, x{uv}))$$

其中U和M是我们要学习的函数，x_{uv}为节点u和v之间边的特征，l表示卷积网络的第l层。



数据处理

工具

使用rdkit+dgl的方法，使用rdkit的分子表示，并且用dgl中的工具提取特征。

方法

由于一开始数据表示使用的是smiles的字符串，如`CCN(CC)C(=S)SSC(=S)N(CC)CC`

可以通过如下步骤将其变为模型输入：

特征提取器

`CanonicalAtomFeaturizer`和`CanonicalBondFeaturizer`分别作为提取分子节点和分子边信息的工具。这在dgl的工具中`featurizer`中定义。

- 原子特征提取为：atom_type_one_hot, atom_degree_one_hot, atom_implicit_valence_one_hot, atom_formal_charge, atom_num_radical_electrons，atom_hybridization_one_hot, atom_is_aromatic, atom_total_num_H_one_hot

- 键特征提取为：bond_type_one_hot, bond_is_conjugated, bond_is_in_ring, bond_stereo_one_hot

特征提取方法

使用dgl中定义的特征提取api，可以直接从smiles字符串转为dglgraph的图

`smiles_to_bigraph(m, add_self_loop=False, node_featurizer=atom_featurizer,edge_featurizer=bond_featurizer)`

然后可以将其封装为测试集和训练集，再使用torch自带的dataloader组装成batchsize为64（视情况定）的训练集和测试集。

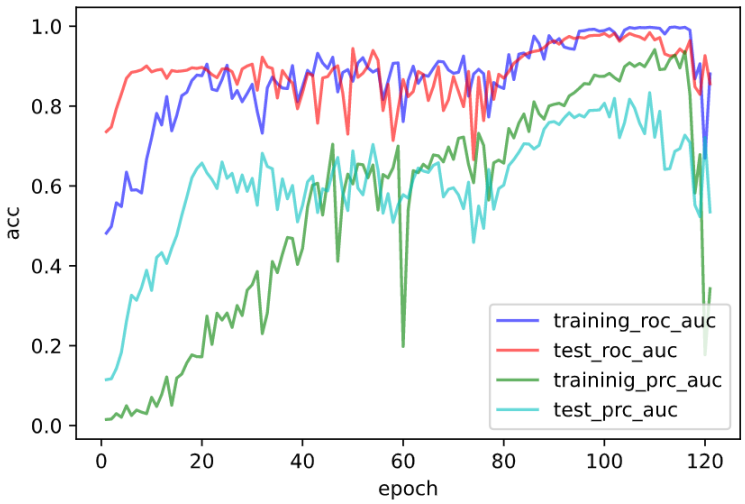
结果

MPNN

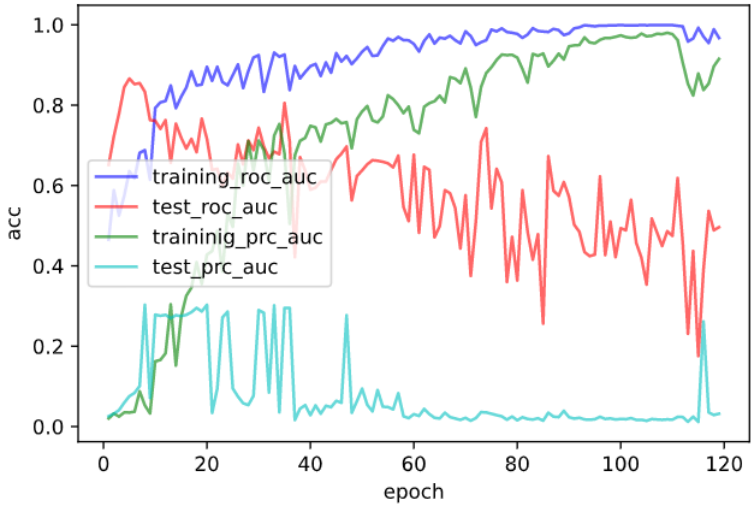
使用MPNN模型进行训练，提取点特征74个，边特征12个，当batch大小设置为64, epoch大小设置为20。虽然epoch较小，对于10个分数据集，训练集ROC-AUC都在接近20个epoch时达到0.85左右，但对测试集进行10折交叉验证后ROC-AUC，较好的数据集在0.6到0.8范围内波动，而剩下的数据集中波动程度很大。

以下一些数据集的图示结果：

fold0:

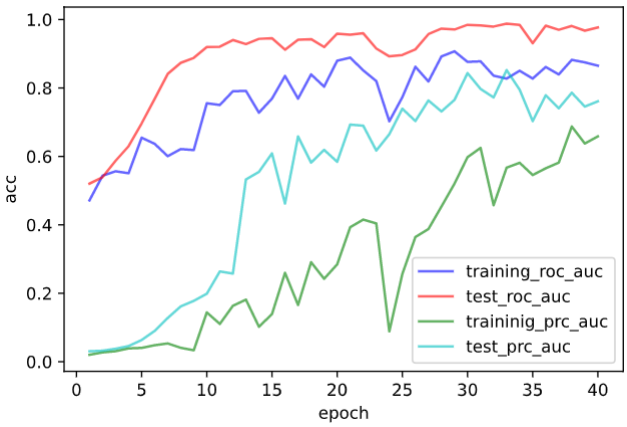
1. 

fold2:

2. 

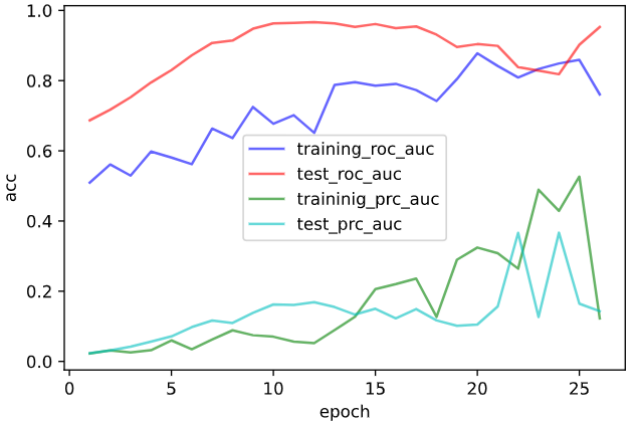
fold3:

3.



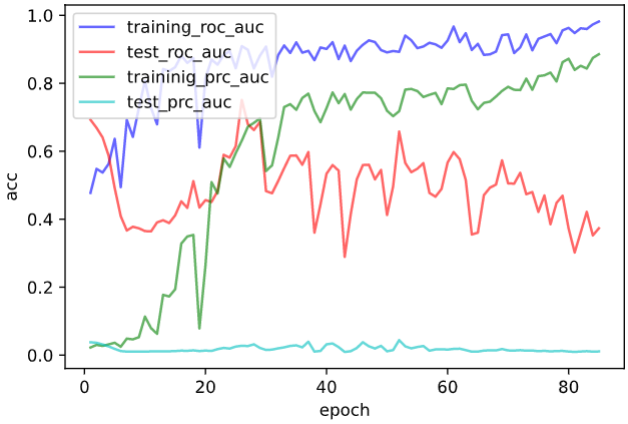
fold4:

4.



fold5:

5.



为了继续分析该模型的性能，将其中一个小数据集拿出来以后，将epoch设置到150。由于我们电脑性能的不足，训练了将近两小时才得出结果。epoch增大时，训练集ROC-AUC基本稳步增大，当120之后基本收敛到0.97左右，这说明该分类器在训练集上的表现已经相当好。可是对测试集进行交叉验证以后发现ROC-AUC一直在0.65到0.85范围内波动，且方差较大。推测原因可能有如下几点：

1. 模型在训练过程中过拟合。
2. 由于测试集太小，再对其进行十折交叉验证得到的实际测试集就过于小，因此波动较大。
3. 测试集和训练集内负例都远远多于正例，导致ROC-AUC值不能很好的反映模型的分类能力。

最终根据其他测试集的结果可以看出当epoch很大时可以确定存在过拟合现象。

下面是训练过程中表现比较好的数据集中最优情况的记录。

```
epoch 109/200, training roc_auc_score 0.9981 prc_auc 0.9195
epoch 109/200, validation roc_auc_score 0.9818, prc_auc_score 0.8335, best va
lidation roc_auc_score 0.9818
 epoch 52/200, training roc_auc_score 0.9086 prc_auc 0.7178
 epoch 52/200, validation roc_auc_score 0.6624, prc_auc_score 0.0441, best val
 idation roc_auc_score 0.6624
 epoch 37/200, training roc_auc_score 0.8586 prc_auc 0.3502
 epoch 37/200, validation roc_auc_score 0.7440, prc_auc_score 0.0088, best val
 idation roc_auc_score 0.7440
```

```
epoch 21/200, training roc_auc_score 0.7819 prc_auc 0.2745
epoch 21/200, validation roc_auc_score 0.9911, prc_auc_score 0.5193, best val
idation roc_auc_score 0.9911
```

```
 epoch 44/200, training roc_auc_score 0.8732 prc_auc 0.3767
 epoch 44/200, validation roc_auc_score 0.7657, prc_auc_score 0.0111, best val
 idation roc_auc_score 0.7657
```

```
epoch 31/200, training roc_auc_score 0.8379 prc_auc 0.5715
epoch 31/200, validation roc_auc_score 0.9958, prc_auc_score 0.8500, best val
idation roc auc score 0.9958
```

```
epoch 52/200, training roc_auc_score 0.9500 prc_auc 0.8301
epoch 52/200, validation roc_auc_score 0.8628, prc_auc_score 0.4442, best val
idation roc_auc_score 0.8628
```

下面是最佳效果时模型的表现。跑通了十个fold，mpnn模型比较优秀的能够应对这个问题。

下表展示了不同数据集最佳情况下模型的表现。

fold	train roc auc	train prc auc	dev roc auc	dev prc auc
0	0.9981	0.9195	0.9818	0.8335
1	0.9350	0.7171	0.8712	0.3110
2	0.9985	0.9675	0.8535	0.0210
3	0.8711	0.4535	0.9718	0.7314
4	0.7999	0.0891	0.9613	0.1549
5	0.9721	0.8808	0.6893	0.0383
6	0.8732	0.3767	0.7657	0.0111
7	0.7819	0.2745	0.9911	0.5193
8	0.8379	0.5715	0.9958	0.8500
9	0.9500	0.8301	0.8628	0.4442

实验中也试验了SchNet, MGCN模型，但由于效果太差，我们最终采用MPNN模型，根据以上实验结果，其可以较好的进行Property Prediction工作。

人员分工

宁晨然 17307130178

主要负责完成问题分解、搭建模型、资料查询等工作，具体做的有：

- 问题初次尝试：查阅DGL框架和Rdkit使用方法，搭建简单GCN模型，完成了csv到dglgraph的完整数据读入和处理流程，确定模型流程。
- 问题再尝试：修改了数据处理bug，使用dgl的api进行特征提取，包含每个分子的节点和边信息
- 问题过程：跑通几个模型，确定mpnn模型后进行十个fold的测试，得出实验结果
- 问题总结：合作完成报告

杜逸闲 17300240036

主要负责阅读文献、搜寻代码示例、撰写报告工作：

- 前期了解了问题描述，数据构成以及SMILES语法。
- 阅读了关于GNN, GCN, MPNN模型的文献和文章。
- 搜寻了本次项目使用的DGL框架样例代码。
- 撰写报告的主要内容。