评阅成绩

西北工業大學

复杂网络分析课程报告

专业：	软件工程
姓名：	张苏宇
班级：	14012106
学号：	2021302853

一、实验目的

通过本课程作业加深对复杂网络基础理论的认识和了解，锻炼定量、定性分析复杂拓扑结构特征，并解决问题的能力。通过对复杂网络进行建模、分析、验证，基本掌握复杂网络数据挖掘的基本过程，为后续学习大数据挖掘、人工智能等奠定基础。

二、实验原理

数据预处理和特征工程

通过节点度和节点中心性等网络特征，增强基因分类模型的特征表示能力。标准化特征以提高模型训练的效果。

- **节点度（Node Degree）**：表示每个节点（基因）在网络中连接的其他节点的数量。节点度高的基因可能在网络中具有重要的地位。
- **节点中心性（Node Centrality）**：通过相邻节点的连接度来衡量一个节点的重要性。中心性高的节点可能在网络中具有更重要的角色。
- **特征标准化**：使用 StandardScaler 对特征进行标准化，使每个特征的均值为0，标准差为1。这有助于模型更快收敛，并防止某些特征对模型产生过大影响。

特征选择

在机器学习中，特征选择是通过某种算法从数据中选择最重要特征的过程，目的是提高模型的性能和效率，减少训练时间，降低模型的复杂性以及避免过拟合。

随机森林进行特征选择

1. **随机森林**：它是一种集成学习方法，通过构建多个决策树并结合其预测结果来提高模型的准确性和稳定性。每棵树在训练时使用了随机选择的样本和特征。
2. **特征重要性**：随机森林可以计算每个特征的重要性。特征重要性是通过计算一个特征在所有树中的分裂点对模型性能的贡献来确定的。通常，这种贡献是通过衡量每个特征的Gini不纯度或信息增益来计算的。
3. **SelectFromModel**：这个类允许你基于特定的阈值选择重要特征。在这个例子中，基于 `RandomForestClassifier` 训练出的特征重要性，选择重要性高于某个阈值的特征。

模型训练

模型训练的目标是找到能够最好地拟合训练数据的模型参数。本文中使用了多种模型进行训练和评估，下面以 `XGBoost` 为例，详细说明模型训练的过程。

```
models = {
    'XGBoost': xgb.XGBClassifier(tree_method='gpu_hist', gpu_id=0, n_estimators=100, random_state=42)
}

param_grids = {
    'XGBoost': {'n_estimators': [100, 200], 'learning_rate': [0.01, 0.1]}
}
```

XGBoost 模型训练

1. **初始化模型**：通过设置 `tree_method='gpu_hist'` 和 `gpu_id=0` 来使用 GPU 进行加速训练。
2. **超参数搜索**：通过 `GridSearchCV` 进行超参数搜索，找到最优的超参数组合。超参数包括 `n_estimators` 和 `learning_rate`。
3. **交叉验证**：使用 `StratifiedKFold` 进行交叉验证，将数据分成多个折叠，确保每个折叠中类别分布相同。交叉验证可以有效评估模型的性能，减少过拟合风险。
4. **训练模型**：在交叉验证的每个折叠上训练模型，并在验证集上进行评估，计算 AUPRC 和 ROCAUC。

主要步骤

1. **交叉验证**：使用 `StratifiedKFold` 将数据分成 5 份，循环进行训练和验证。每次都使用 4 份数据训练，1 份数据验证，确保模型的稳定性和泛化能力。
2. **网格搜索**：使用 `GridSearchCV` 进行超参数搜索，通过交叉验证选择最优的超参数组合。
3. **训练模型**：使用最佳超参数组合在训练数据上训练模型。
4. **预测与评估**：在验证集上进行预测，并计算平均精度（AUPRC）和受试者工作特性曲线下面积（ROCAUC）。

模型评估

使用 AUPRC 和 ROCAUC 指标评估模型在测试集上的性能，确保模型能够有效区分基因类别。模型评估是确定模型性能的关键步骤。主要通过一些性能指标来衡量模型在测试集上的表现。在基因分类任务中，常用的评估指标包括：

- **平均精度-召回曲线下的面积（AUPRC, Area Under Precision-Recall Curve）**：

AUPRC 衡量模型在所有阈值下的精度和召回率之间的权衡。对于不平衡数据集（如基因分类问题），AUPRC 是比 ROC 曲线更好的评估指标。

- **ROC 曲线下的面积（ROCAUC, Area Under Receiver Operating Characteristic Curve）**：

ROCAUC 衡量模型在所有阈值下的真阳性率（TPR）和假阳性率（FPR）之间的权衡。ROCAUC 的值在0.5到1之间，越接近1表示模型性能越好。

三、实验方法

这段代码主要实现了一个基因分类的机器学习实验，包括数据加载、预处理、特征工程、特征选择、模型训练与评估、结果保存等步骤。下面是对每个部分的详细解释：

导入库

首先导入所需的库：

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.metrics import roc_auc_score, average_precision_score
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectFromModel
import xgboost as xgb
import joblib
```

这些库涵盖了数据处理、模型选择、特征工程、模型训练和评估等功能。

数据加载

定义 `load_data` 函数，加载特征矩阵、标签和邻接矩阵：

```
def load_data():
    features = pd.read_csv('C:/Users/admin/Desktop/test/data/features.csv')
    labels = pd.read_csv('C:/Users/admin/Desktop/test/data/labels.csv')
    adj = pd.read_csv('C:/Users/admin/Desktop/test/data/adj.csv', header=None)
    return features, labels, adj
```

数据预处理

定义 `preprocess_data` 函数，处理标签中的缺失值，拆分训练集和测试集：

```python
def preprocess_data(features, labels):
    labels = labels.replace(-1, np.nan)
    known_labels = labels.dropna()
    X_train, X_test, y_train, y_test = train_test_split(
        features.loc[known_labels.index],
        known_labels.values.ravel(),
        test_size=0.2,
        random_state=42,
        stratify=known_labels.values.ravel()
    )
    return X_train, X_test, y_train, y_test
```

特征工程

定义 `feature_engineering` 函数，计算节点度和节点中心性，并进行标准化处理：

```python
def feature_engineering(features, adj):
    node_degrees = adj.sum(axis=1)
    features['node_degree'] = node_degrees

    # Adding more network features
    node_centrality = adj.dot(adj).sum(axis=1)
    features['node_centrality'] = node_centrality

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(features)
    return X_scaled, scaler
```

定义 `add_node_degree` 函数，计算节点度和节点中心性并添加到特征中：

```python
def add_node_degree(features, adj):
    node_degrees = adj.sum(axis=1)
    features['node_degree'] = node_degrees

    # Adding more network features
    node_centrality = adj.dot(adj).sum(axis=1)
    features['node_centrality'] = node_centrality

    return features
```

特征选择

定义 `select_important_features` 函数，使用随机森林选择重要特征：

```python
def select_important_features(X_train, y_train):
    selector = SelectFromModel(RandomForestClassifier(n_estimators=100, random_state=42))
    selector.fit(X_train, y_train)
    X_train_selected = selector.transform(X_train)
    return X_train_selected, selector
```

模型训练与评估

定义 `train_and_evaluate` 函数，使用交叉验证进行模型训练和评估，计算 AUPRC 和 ROCAUC：

```python
def train_and_evaluate(X_train, y_train, model, param_grid=None):
    kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

    auprc_scores = []
    rocauc_scores = []

    for train_index, val_index in kf.split(X_train, y_train):
        X_train_fold, X_val_fold = X_train[train_index], X_train[val_index]
        y_train_fold, y_val_fold = y_train[train_index], y_train[val_index]

        if param_grid:
            grid_search = GridSearchCV(model, param_grid, cv=3, scoring='roc_auc')
            grid_search.fit(X_train_fold, y_train_fold)
            best_model = grid_search.best_estimator_
        else:
            best_model = model
            best_model.fit(X_train_fold, y_train_fold)

        y_val_pred = best_model.predict_proba(X_val_fold)[:, 1]

        auprc = average_precision_score(y_val_fold, y_val_pred)
        rocauc = roc_auc_score(y_val_fold, y_val_pred)

        auprc_scores.append(auprc)
        rocauc_scores.append(rocauc)

    return np.mean(auprc_scores), np.std(auprc_scores), np.mean(rocauc_scores), np.std(rocauc_scores)
```

定义 `evaluate_on_test` 函数，在测试集上评估模型：

```python
def evaluate_on_test(model, X_test, y_test):
    y_test_pred = model.predict_proba(X_test)[:, 1]
    auprc_test = average_precision_score(y_test, y_test_pred)
    rocauc_test = roc_auc_score(y_test, y_test_pred)
    return auprc_test, rocauc_test
```

结果保存

定义 `save_results` 函数，保存评估结果到 CSV 文件和文本报告中：

```python
def save_results(results):
    results_df = pd.DataFrame(results)
    results_df.to_csv('C:/Users/admin/Desktop/test/results.csv', index=False)

    with open('C:/Users/admin/Desktop/test/report.txt', 'w') as report:
        report.write("基因分类项目实验报告\n")
        report.write("==================================\n\n")
        for result in results:
            report.write(f"Model: {result['Model']}\n")
            report.write(f"AUPRC: {result['AUPRC']:.4f} ± {result['AUPRC_STD']:.4f}\n")
            report.write(f"ROCAUC: {result['ROCAUC']:.4f} ± {result['ROCAUC_STD']:.4f}\n")
            report.write(f"Test AUPRC: {result['Test_AUPRC']:.4f}\n")
            report.write(f"Test ROCAUC: {result['Test_ROCAUC']:.4f}\n")
            report.write("==================================\n")
```

主函数

定义 `main` 函数，加载数据，进行预处理，特征工程和特征选择，训练和评估模型，并保存结果：

```python
def main():
    features, labels, adj = load_data()
    X_train, X_test, y_train, y_test = preprocess_data(features, labels)

    # Feature engineering on training data
    X_train = add_node_degree(X_train, adj)
    X_train_scaled, scaler = feature_engineering(X_train, adj)

    # Feature selection
    X_train_selected, selector = select_important_features(X_train_scaled, y_train)
```

```
11
12      # Add node degree to test data and scale it
13      X_test = add_node_degree(X_test, adj)
14      X_test_scaled = scaler.transform(X_test)
15      X_test_selected = selector.transform(X_test_scaled)
16
17      models = {
18          'XGBoost': xgb.XGBClassifier(tree_method='gpu_hist',
    gpu_id=0, n_estimators=100, random_state=42)
19      }
20
21      param_grids = {
22          'XGBoost': {'n_estimators': [100, 200], 'learning_rate':
    [0.01, 0.1]}
23      }
24
25      results = []
26
27      for name, model in models.items():
28          print(f"Training {name}...")
29          param_grid = param_grids.get(name, None)
30          auprc_mean, auprc_std, rocauc_mean, rocauc_std =
    train_and_evaluate(X_train_selected, y_train, model, param_grid)
31          model.fit(X_train_selected, y_train)
32          auprc_test, rocauc_test = evaluate_on_test(model,
    X_test_selected, y_test)
33
34          results.append({
35              'Model': name,
36              'AUPRC': auprc_mean,
37              'AUPRC_STD': auprc_std,
38              'ROCAUC': rocauc_mean,
39              'ROCAUC_STD': rocauc_std,
40              'Test_AUPRC': auprc_test,
41              'Test_ROCAUC': rocauc_test
42          })
43
44      save_results(results)
45
46  if __name__ == '__main__':
47      main()
```

四、实验结果

模型性能比较

在本实验中，四个模型分别为RandomForest、GradientBoosting、LogisticRegression和SVC。我们通过计算和比较每个模型在验证集上的平均AUPRC和ROCAUC，以及它们在测试集上的表现来评估性能。

1. 随机森林（RandomForest）

随机森林是一种基于多棵决策树的集成学习方法，通过引入随机性来提高模型的泛化能力。

2. 梯度提升（GradientBoosting）

梯度提升是一种集成学习方法，通过逐步训练多个弱学习器（通常是决策树），每一步都试图纠正前一步的错误。

3. 逻辑回归（LogisticRegression）

逻辑回归是一种广泛使用的线性分类模型，适用于处理二分类问题，特别是当数据具有线性可分性时效果较好。

4. 支持向量机（SVC）

支持向量机是一种监督学习模型，通过找到一个超平面来最大化不同类别之间的边界。使用非线性核函数可以处理复杂的数据分布。

模型比较代码

```
def main():
    features, labels, adj = load_data()
    X_train, X_test, y_train, y_test = preprocess_data(features, labels)

    # Feature engineering on training data
    X_train = add_node_degree(X_train, adj)
    X_train_scaled, scaler = feature_engineering(X_train, adj)

    # Feature selection
    X_train_selected, selector = select_important_features(X_train_scaled, y_train)

    # Add node degree to test data and scale it
    X_test = add_node_degree(X_test, adj)
    X_test_scaled = scaler.transform(X_test)
    X_test_selected = selector.transform(X_test_scaled)

    models = {
```

```
        'RandomForest': RandomForestClassifier(n_estimators=100, random_state=42),
        'GradientBoosting': GradientBoostingClassifier(n_estimators=100, random_state=42),
        'LogisticRegression': LogisticRegression(max_iter=1000, random_state=42),
        'SVC': SVC(probability=True, random_state=42)
    }

    param_grids = {
        'RandomForest': {'n_estimators': [100, 200]},
        'GradientBoosting': {'n_estimators': [100, 200], 'learning_rate': [0.01, 0.1]},
        'LogisticRegression': {'C': [0.01, 0.1, 1, 10]},
        'SVC': {'C': [0.01, 0.1, 1, 10]}
    }

    results = []

    for name, model in models.items():
        print(f"Training {name}...")
        param_grid = param_grids.get(name, None)
        auprc_mean, auprc_std, rocauc_mean, rocauc_std = train_and_evaluate(X_train_selected, y_train, model, param_grid)
        model.fit(X_train_selected, y_train)
        auprc_test, rocauc_test = evaluate_on_test(model, X_test_selected, y_test)

        results.append({
            'Model': name,
            'AUPRC': auprc_mean,
            'AUPRC_STD': auprc_std,
            'ROCAUC': rocauc_mean,
            'ROCAUC_STD': rocauc_std,
            'Test_AUPRC': auprc_test,
            'Test_ROCAUC': rocauc_test
        })

    save_results(results)

if __name__ == '__main__':
    main()
```

性能比较结果

运行完代码后得到以下结果：

1. 随机森林（RandomForest）

- **AUPRC:** 0.7691 ± 0.0224
- **ROCAUC:** 0.8464 ± 0.0201
- **Test AUPRC:** 0.7459
- **Test ROCAUC:** 0.8257

2. 梯度提升（GradientBoosting）

- **AUPRC:** 0.7548 ± 0.0218
- **ROCAUC:** 0.8451 ± 0.0072
- **Test AUPRC:** 0.7355
- **Test ROCAUC:** 0.8158

3. 逻辑回归（LogisticRegression）

- **AUPRC:** 0.7298 ± 0.0118
- **ROCAUC:** 0.8411 ± 0.0190
- **Test AUPRC:** 0.7151
- **Test ROCAUC:** 0.7953

4. 支持向量机（SVC）

- **AUPRC:** 0.7324 ± 0.0099
- **ROCAUC:** 0.8423 ± 0.0120
- **Test AUPRC:** 0.7258
- **Test ROCAUC:** 0.8051

五、评价分析

这个代码实现了一个完整的基因分类流程，包括数据加载、预处理、特征工程、特征选择、模型训练和评估，以及结果保存。使用了包括 `RandomForest`、`GradientBoosting`、`LogisticRegression`、`SVC` 等在内的多个机器学习工具和方法，并且特意加入了 `XGBoost` 进行 GPU 加速训练。

结果分析

1. **RandomForest:**
    - AUPRC和ROCAUC都较高，说明在处理不平衡数据上具有较好的表现。
    - 在测试集上的表现也较为稳定，证明其泛化能力较强。
2. **GradientBoosting:**
    - AUPRC和ROCAUC稍低于RandomForest，但依然有较好的表现。
    - 测试集上的表现也较为稳定，且与训练集的结果较为一致。

3. **LogisticRegression:**
    - AUPRC和ROCAUC相对较低，可能因为数据集的非线性关系较复杂，线性模型的表现不如非线性模型。
    - 但逻辑回归具有解释性强、计算效率高的优点。
4. **SVC:**
    - AUPRC和ROCAUC优于LogisticRegression，但略低于RandomForest和GradientBoosting。
    - 在处理复杂数据时，支持向量机的非线性核函数提供了一定的优势。

总结

- **RandomForest**表现最佳，具有较高的AUPRC和ROCAUC，且在测试集上也表现出色，适合处理该数据集。
- **GradientBoosting**次之，虽然表现略逊于RandomForest，但依然表现良好，适合需要较高准确率和召回率的场景。
- **SVC**在复杂数据集上的表现优于LogisticRegression，但计算复杂度较高。
- **LogisticRegression**虽然表现不如前几者，但在需要快速计算和结果可解释性时仍然是一个不错的选择。

根据具体应用场景和资源限制，可以选择合适的模型进行部署。