类别不均衡问题处理

csdn

1、介绍

许多数据集存在 **类别不均衡问题 (Class-imbalance problem)** 或者说 **长尾问题 (Long-tail problem)**，即数据集中不同类别样本数比例差异过大的情况。然而，大部分机器学习算法在这类不均衡数据集上表现不佳。为此，可通过 设置类别权重、仔细选择指标、尝试不同算法、异常点检测、重采样数据集 等各种方法应对。以下则梳理了使用 imblance-learn 库 (简称 imblearn) 处理数据不均衡问题的若干方法。

2、过采样（Over-sampling）

2.1 随机过采样

随机过采样的原理

随机过采样通过随机复制少数类别的样本来增加其在数据集中的比例，从而达到与多数类别相似的数量级。这种方法直接增加了少数类别的样本数量，改善了类别之间的不平衡问题。通过这种方式，模型在训练过程中能够更好地学习到少数类别的特征，提高对少数类别的识别准确率。

随机过采样的步骤

1. **确定少数类别和多数类别**：首先分析数据集，确定哪些类别是少数类别，哪些是多数类别。
2. **复制少数类别的样本**：随机选择少数类别中的样本并复制，直到少数类别的样本数量达到预定的数量，这个数量通常与多数类别的样本数量相近或相等。
3. **合并数据集**：将复制的少数类别样本与原数据集中的其他样本合并，形成新的、更平衡的数据集。
4. **重新训练模型**：使用新的、平衡的数据集重新训练机器学习模型。

优点：

- 简单易实现：随机过采样技术操作简单，易于实现。
- 改善不平衡：能够有效改善数据集的不平衡问题，提高模型对少数类别的识别能力。

缺点：

- 过拟合风险：由于是通过复制少数类别样本来增加样本数量，可能会增加模型对这些重复样本的过拟合风险。
- 数据真实性：增加的样本是复制现有样本得到的，没有引入新的信息，可能影响模型学习到的少数类别的多样性。

```
from imblearn.over_sampling import RandomOverSampler
from sklearn.model_selection import train_test_split

# 假设 X 和 y 是你的数据
# X = ...  # 你的特征
# y = ...  # 你的标签

# 可选：首先分割数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 初始化过采样器
```

```
ros = RandomOverSampler(random_state=42)

# 应用过采样到训练数据
X_resampled, y_resampled = ros.fit_resample(X_train, y_train)

# 现在 X_resampled 和 y_resampled 是过采样后的训练数据，可以用于模型训练
# 测试集不应该采样
```

2.2 合成少数类过采样技术 (Synthetic Minority Over-sampling Technique, SMOTE)

合成少数类过采样技术（Synthetic Minority Over-sampling Technique，简称SMOTE）是一种解决不平衡数据集问题的流行方法。与简单的随机过采样不同，SMOTE通过在少数类样本之间插值生成新的合成样本，而不是简单地复制现有样本。这样可以增加少数类的样本数量，同时引入一定的多样性，减少过拟合的风险。

SMOTE的工作原理

SMOTE的基本步骤如下：

1. **对于每一个少数类样本**：从它的最近邻中随机选择一些样本（通常选择k个最近邻）。
2. **插值**：对于每一个选中的最近邻，生成一个新的样本。新样本的特征是原始样本特征和这个邻居样本特征之间的差值的随机加权和，加上原始样本的特征。
3. **重复**：重复上述过程，直到达到预期的少数类样本数量。

```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

# 假设 X 和 y 是你的数据
# X = ...  # 你的特征
# y = ...  # 你的标签

# 可选：分割数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 初始化SMOTE实例
smote = SMOTE(random_state=42)

# 应用SMOTE到训练数据
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

# 现在 X_resampled 和 y_resampled 是过采样后的训练数据，可以用于模型训练
```

2.3 自适应合成采样（Adaptive Synthetic Sampling，简称ADASYN）

自适应合成采样（Adaptive Synthetic Sampling，简称ADASYN）是一种用于处理不平衡数据集的过采样技术，与SMOTE（合成少数类过采样技术）类似。ADASYN的核心思想是根据数据分布自动适应地生成少数类样本，以此来改善学习器的性能。与SMOTE相比，A**DASYN在生成合成样本时更加关注那些难以正确分类的少数类样本。**

ADASYN工作原理

ADASYN通过以下步骤增加少数类的样本数，从而减少模型学习过程中的类别不平衡问题：

1. **计算每个少数类样本的类不平衡比率**：对于每个少数类样本，计算其与多数类样本之间的距离，以此来确定每个少数类样本需要生成的合成样本数。样本周围多数类样本越多，需要生成的合成样本

就越多，这样可以确保模型在这些“困难”区域得到更好的训练。
2. **生成合成样本**：对每个少数类样本，根据其邻近的少数类样本随机生成新的合成样本。生成的新样本数量与第一步计算的类不平衡比率成正比。
3. **更新数据集**：将生成的合成样本添加到原始数据集中，形成一个新的、更平衡的数据集。

ADASYN的优点

- **更好地关注困难样本**：通过为难以正确分类的少数类样本生成更多的合成样本，ADASYN帮助模型在这些区域获得更好的性能。
- **提高模型的泛化能力**：通过引入更多多样性的合成样本，减少了模型在少数类样本上过拟合的风险。

ADASYN的缺点

- **增加计算成本**：与SMOTE相比，ADASYN在确定每个少数类样本的合成样本数时需要更多的计算，因此可能会增加总体的计算成本。
- **可能引入噪声**：如果少数类样本本身就很分散，ADASYN生成的合成样本可能会引入噪声，影响模型的准确性。

3、欠采样（Under-sampling）

3.1 随机欠采样 (RandomUnderSampler)

对多数类样本进行随机抽样，保留多数类样本的一个子集，以减少原数据集中多数类样本的比例，实现类别均衡。

```
# API：https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html
from imblearn.under_sampling import RandomUnderSampler

# 实例化 RandomUnderSampler 类对象
rus = RandomUnderSampler(random_state=0)
# 生成新数据集
x_train_rus, y_train_rus = rus.fit_resample(x_train, y_train)
```

3.2 ClusterCentroids

ClusterCentroids 通过 用 KMeans 算法的簇质心替换一个由多数类样本组成的簇 来对多数类样本进行欠采样。注意，作为一种 原型生成 (Prototype Generation) 技术，其将减少原数据集目标类别的样本数，但剩余的样本是从原数据集中 合成得到的，而非诸如随机欠采样等 原型选择 (Prototype Selection) 技术选择留下的。

```
# API：https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.ClusterCentroids.html
from imblearn.under_sampling import ClusterCentroids

# 实例化 ClusterCentroids 类对象
cc = ClusterCentroids(random_state=0)
# 生成新数据集
x_train_cc, y_train_cc = cc.fit_resample(x_train, y_train)
```

3.3 NearMiss

NearMiss是一种欠采样技术，用于处理不平衡数据集中的类不平衡问题。它的核心思想是从多数类中选择那些最接近少数类的样本，从而减少多数类样本的数量，以达到类平衡的目的。这种方法通过保留多数类中与少数类最相似（或"最近"）的样本，希望能够保留对建模最有价值的信息。

NearMiss有几个变体，主要区别在于选择多数类样本的具体规则：

NearMiss-1

选择距离少数类样本平均距离最近的多数类样本。

NearMiss-2

选择距离少数类样本最远的平均距离最近的多数类样本。

NearMiss-3

对每个少数类样本，选择离它最近的k个多数类样本。

优点

- **提高分类性能**：通过专注于那些与少数类相似的多数类样本，NearMiss旨在提高模型对少数类的分类性能。
- **减少计算成本**：减少数据集的大小，可以降低模型训练的计算成本和时间。

缺点

- **信息丢失**：欠采样可能会导致丢失重要信息，特别是当随机丢弃多数类样本时。
- **增加过拟合风险**：通过过度关注与少数类相似的样本，可能会增加模型对这些特定样本的过拟合风险。

```
from imblearn.under_sampling import NearMiss
from sklearn.model_selection import train_test_split

# 假设 X 和 y 是你的数据
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 初始化NearMiss实例
nm = NearMiss(version=1, n_neighbors=3)

# 应用NearMiss欠采样
X_resampled, y_resampled = nm.fit_resample(X_train, y_train)

# 现在 X_resampled 和 y_resampled 包含了重采样后的数据
```

3.3 Tomek Links

Tomek Links是一种用于处理不平衡数据集的欠采样方法。它的核心思想是识别并移除那些多数类和少数类之间的边界上的样本对，这些样本对被称为Tomek Links。通过移除这些样本，可以使类别之间的边界更加清晰，从而帮助改善分类器的性能。

Tomek Links的定义

一对样本（x, y）被认为是一个Tomek Link，如果它们属于不同类别，并且在它们之间没有其他样本更接近任何一个样本。换句话说，两个样本彼此是最近邻，并且属于不同的类别。

Tomek Links的工作原理

1. **识别Tomek Links**：在数据集中寻找所有的Tomek Links对。
2. **移除样本**：从数据集中移除这些Tomek Links中多数类的样本。这样做的目的是去除那些可能导致分类决策边界模糊的样本。

使用Tomek Links的优点

- **减少模型的复杂性**：通过清除边界上可能导致误分类的多数类样本，Tomek Links有助于简化模型的决策边界。
- **保留有用信息**：与随机欠采样不同，Tomek Links方法只移除对分类边界有负面影响的多数类样本，尽量保留有用信息。
- **适用性广**：Tomek Links可以作为其他采样技术的补充，提高处理不平衡数据集时的灵活性。

使用Tomek Links的缺点

- **可能不足以解决严重不平衡问题**：如果数据集非常不平衡，仅仅移除Tomek Links可能不足以显著改善类别不平衡问题。
- **计算成本**：寻找Tomek Links可能需要较多的计算资源，尤其是在大数据集上

```
from imblearn.under_sampling import TomekLinks
from sklearn.model_selection import train_test_split

# 假设 X 和 y 是你的数据
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 初始化TomekLinks实例
tl = TomekLinks()

# 应用TomekLinks欠采样
X_resampled, y_resampled = tl.fit_resample(X_train, y_train)

# 现在 X_resampled 和 y_resampled 包含了重采样后的数据
```

3.4 Edited Nearest Neighbors

Edited Nearest Neighbors (ENN) 是一种欠采样技术，用于处理不平衡数据集。ENN的目的是通过编辑多数类的样本来减少类之间的重叠，从而改善分类器的性能。这种方法通过移除那些与最近邻属于不同类的多数类样本来实现。

Edited Nearest Neighbors的工作原理

1. **计算最近邻**：对于数据集中的每个样本，计算其k个最近邻（k通常是一个小整数，例如3）。
2. **编辑规则**：如果一个样本的大多数（例如，超过一半）最近邻属于不同的类别，则该样本被认为是噪声或边界点，并从数据集中移除。通常，这种编辑只应用于多数类的样本。
3. **重采样**：通过重复上述过程，数据集中多数类的样本被编辑，减少了类别之间的重叠，使得分类边界更加清晰。

使用Edited Nearest Neighbors的优点

- **提高分类精度**：通过减少类别之间的重叠，ENN有助于提高分类器对少数类的识别能力。
- **减少噪声**：ENN可以移除那些可能被错误标记或在决策边界附近的多数类样本，从而减少数据集的噪声。
- **保持数据分布**：与随机欠采样相比，ENN更加关注于移除重叠区域的样本，尽量保持了原始数据的分布特征。

使用Edited Nearest Neighbors的缺点

- **计算成本**：寻找最近邻并判断是否移除样本需要一定的计算资源，尤其是在大规模数据集上。
- **可能的信息丢失**：虽然ENN旨在移除噪声和边界样本，但有时也可能会导致有用信息的丢失。

```
from imblearn.under_sampling import EditedNearestNeighbours
from sklearn.model_selection import train_test_split

# 假设 X 和 y 是你的数据
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 初始化ENN实例
enn = EditedNearestNeighbours()

# 应用ENN欠采样
X_resampled, y_resampled = enn.fit_resample(X_train, y_train)

# 现在 X_resampled 和 y_resampled 包含了重采样后的数据
```

4.过采样和欠采样组合方法

SMOTE 可通过在边缘离群点和内联点之间插入新的样本点，但这样的样本有时会被视为噪声样本影响分类。为此，可通过诸如 Tomeklinks 和 Edited Nearest Neighbours 等数据清洗技术清除噪点。于是，通过结合两类技术，先进行过采样，再执行数据清洗，即可形成一个组合了过采样和欠采样方法的非平衡学习 Pipeline。

常用的现有工具有：SMOTETomek (SMOTE + Tomeklinks) 和 SMOTEENN (SMOTE + Edited Nearest Neighbours)。其中，SMOTEENN 比 SMOTETomek 更容易清除噪声样本。

```
from imblearn.combine import SMOTEENN
from sklearn.model_selection import train_test_split

# 假设 X 和 y 是你的数据
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 初始化SMOTEENN实例
smote_enn = SMOTEENN(random_state=42)

# 应用SMOTEENN
X_resampled, y_resampled = smote_enn.fit_resample(X_train, y_train)

# 现在 X_resampled 和 y_resampled 包含了经过处理后的数据
```