双向障碍监测

姓名：胡天扬

学号：3190105708

专业：自动化（控制）

课程：人工智能与机器学习

指导教师：张建明

题目分析

本题共有2个特征表，分别是 **医学影像特征** 和 **肠道特征** 。其中 医学影像特征 有 *6670*维特征，肠道特征 有 *377*维特征，而对应的 样本数 仅有 *39*个，因此可以看出这是一个 **多特征少样本** 的学习过程。

由于这是一个 二分类任务，因此可以考虑的模型有 **决策树**、**神经网络**、**小样本学习** 等。

数据预处理

通过特征提取，我们能得到未经处理的特征，这时的特征可能有以下问题：

- 不属于同一量纲：即特征的规格不一样，不能够放在一起比较。无量纲化可以解决这一问题。
- 信息冗余：对于某些定量特征，其包含的有效信息为区间划分，例如学习成绩，假若只关心“及格”或不“及格”，那么需要将定量的考分，转换成“1”和“0”表示及格和未及格。二值化可以解决这一问题。
- 定性特征不能直接使用：某些机器学习算法和模型只能接受定量特征的输入，那么需要将定性特征转换为定量特征。最简单的方式是为每一种定性值指定一个定量值，但是这种方式过于灵活，增加了调参的工作。通常使用独热编码的方式将定性特征转换为定量特征。
- 存在缺失值：缺失值需要补充。

缺失值

```
data.isnull().sum()      # 返回为0说明没有缺失值
```

重复值

```
data.duplicated().sum()      # 返回为0说明没有重复值
```

异常值

利用箱线图进行观察，将超过限制的数据替换为四分位数。

```
from matplotlib import pyplot as plt
plt.boxplot(feature, data)
data = quartile(data)
```

无量纲化

可以采取的方法有 **标准化**、**归一化** 等，本题使用 归一化，即将所有数据缩放至$[0,1]$区间。

```
from sklearn.preprocessing import MinMaxScaler
MinMaxScaler().fit_transform(data)
```

独热编码

独热编码会增加数据维度，之后可以使用 PCA 进行降维。但本题中无需使用独热编码处理。

```
from sklearn.preprocessing import OneHotEncoder
OneHotEncoder().fit_transform(data)
```

特征选择

1. 当数据预处理完成后，我们需要选择有意义的特征输入机器学习的算法和模型进行训练。通常来说，从两个方面考虑来选择特征：

- **特征是否发散**：如果一个特征不发散，例如方差接近于0，也就是说样本在这个特征上基本上没有差异，这个特征对于样本的区分并没有什么用。
- **特征与目标的相关性**：这点比较显见，与目标相关性高的特征，应当优选选择。

2. 根据特征选择的形式又可以将特征选择方法分为3种：

- **Filter**：过滤法，按照发散性或者相关性对各个特征进行评分，设定阈值或者待选择阈值的个数，选择特征。
- **Wrapper**：包装法，根据目标函数，每次选择若干特征，或者排除若干特征。
- **Embedded**：集成法，先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。类似于Filter方法，但是是通过训练来确定特征的优劣。

相关系数选择

本题采用了过滤法中的 **相关系数法**，其中用到的指标是 **皮尔逊相关系数**。

```
from sklearn.feature_selection import SelectKBest
from scipy.stats import pearsonr
SelectKBest(lambda X, Y: array(map(lambda x:pearsonr(x, Y), X.T)).T,
k=2).fit_transform(data, features)
```

由于本题有两张特征表，因此对各自进行特征提取，然后将两者拼接在一起。

```
# 统计特征值和label的皮尔孙相关系数  对两类特征分别进行排序筛选特征
select_feature_number1 = 20
select_feature1 = SelectKBest(lambda X, Y:
tuple(map(tuple,np.array(list(map(lambda x:pearsonr(x, Y),   X.T))).T)),
k=select_feature_number1).fit(feature1,
np.array(label).flatten()).get_support(indices=True)
select_feature_number2 = 2
select_feature2 = SelectKBest(lambda X, Y:
tuple(map(tuple,np.array(list(map(lambda x:pearsonr(x, Y), X.T))).T)),
k=select_feature_number2 .fit(feature2,
np.array(label).flatten()).get_support(indices=True)
# 双模态特征选择并融合
new_features = pd.concat([feature1[feature1.columns.values[select_feature1]],
feature2[feature2.columns.values[select_feature2]]],axis=1)
```

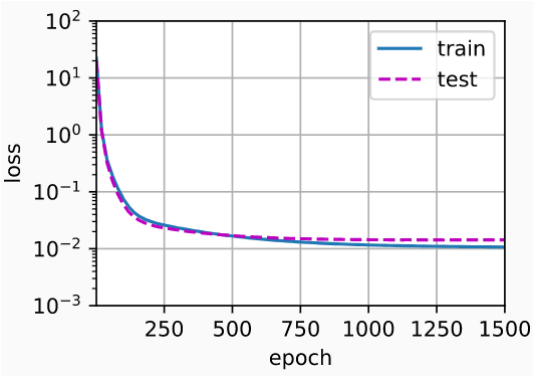
模型搭建

神经网络

用全连接层搭建一个输出为2维的 $Softmax$回归 网络，具体细节省略。

```
# 定义网络
output = 2
net = nn.Sequential(nn.Flatten(), nn.Linear(len(new_features), output))
# 初始化权重
net.apply(init_weights)
# 定义损失函数
loss = nn.CrossEntropyLoss()
# 定义优化器
updater = torch.optim.SGD(net.parameters(), lr=learning_rate)
```

但是经过训练发现，神经网络的超参数设置很多，而本题特征多样本少，因此尽管已经进行了特征提取，仍然十分容易过拟合。



准确率	召回率	F-score
0.8974	0.8235	0.875

随机森林

考虑到集成方法对分类的效果比较好，因此使用随机森林进行尝试。对 表一 提取20个特征，对 表二 提取10个特征，可以得到如下结果。

准确率	召回率	F-score
0.9231	0.8824	0.9091

可以看到效果比神经网络要好一些，但仍然不是很理想。猜想是集成方法仍然过于复杂，由此也可以排除 **小样本学习** 的选项，本题应使用简单的分类模型进行预测。于是采用 **决策树** 进行尝试。

决策树

直接运用题目给出的模型，效果比随机森林要好，于是确定这一模型，下面开始调参。

准确率	召回率	F-score
0.9487	0.8824	0.9375

由于表一的特征数远大于表二，因此从最极端的情况开始考虑，即只提取表一数据。随后根据效果逐步改变参数，直到准确率最高。下表中列出了部分参数的结果。

表一特征数	表二特征数	准确率	召回率	F-score
12	0	0.9231	0.9412	0.9143
12	2	0.9487	0.9412	0.9412
10	2	0.9487	0.9412	0.9412
8	2	0.9231	0.8824	0.9091
18	2	0.9744	1.0	0.9714
20	2	1.0	1.0	1.0

在选取特征数分别为 **20** 和 **2** 之后，可以是模型在我们自己划分的训练集、验证集上都做到三项评分全为 **1.0**，并且在最终的测试中也达到了这一效果。

```
决策树模型结果: {'acc_train': 1.0, 'acc_val': 1.0, 'recall_train': 1.0, 'recall_val': 1.0, 'f_train': 1.0, 'f_val': 1.0}
```

用例测试

测试点	状态	时长	结果
测试结果	✓	7s	测试成功，在10个测试样本中，准确率为 1.0, 召回率为 1.0, F-score为 1.0

心得体会

刚开始走了不少弯路，由于没有考虑到样本数太少的问题，导致使用的模型都过于复杂，无法通过简单地调节模型参数来提升效果。后来想到一句话，特征工程决定了上限，而模型的选取只是在逼近这个上限，于是转用最简单的决策树模型，并将参数调节的重点放在了特征数量上，最后终于使得三项指标达到最优。