人工智能第二次实验实验报告

第一部分：监督学习算法

李瀚民
PB16001680

[任务描述]:
根据所给的数据，选取一定的特侦，对最终结果 G3 进行预测。其中数据分为两组，一组是数学成绩中的 G3 属性，另一种是葡萄牙语成绩中的 G3 属性。并且，为了进行分析测试，对于每一组数据，将会对预测属性包括 G1，G2 以及预测属性不包括 G1，G2 进行分类。

[算法思想]:
首先是 KNN，KNN 很简单，我们不需要对训练的数据进行什么特殊的'学习'过程，只需要将其输入并记录，在之后我们进行预测时，我们首先将预测的数据点读入，然后根据选定的参数 k，进行筛选，选出 k 个与该点最为临近的测试点，之后我们根据这 k 个点的类别进行预测，k 个点中种类数多的，即为要预测点的种类。
其次是 SVM，SVM 的本质是最大 Margin 预测，这个边界是软的，因此要增加参数 C，我们可以根据数据集，进行已知的问题建立，求解 max *MARGIN* with *CONSTRAINTS* 这一问题，但是直接这样解无法适用于非线性预测，因此采用其对偶问题，并将其改成最小值问题如下：Min $(\frac{1}{2}\sum_{i,j=1}^{n} \alpha i\alpha jyiyjxixj - \sum_{i=1}^{n} \alpha i)$. 这个问题可以直接用二次规划问题来求解，解出α以后，可以直接用其来恢复 b，并且将预测函数写成关于α和 b 的函数，这样一来，就可以直接进行预测了。
最后我选的额外的方法是朴素贝叶斯，这建立在条件独立的假设之上，我特意选取了一些数据，使得他们大概在认知上是独立的，之后就是遍历训练集统计频率，然后根据频率预测概率，最后根据测试集每一元组的具体表现，来确定到底哪一种类的可能性较大，由此来判断其为哪一类的。值得说明的是，在朴素贝叶斯模型上，我也采用了 f_score 来衡量整个模型的性能，至于我的目标，其也是用含 G1，G2 的特征和其他一些特征的数据来预测最终的分数 G3

[实验结果说明与分析]:
（以下分析基于同一组特征）
利用 KNN 预测的结果：(这里设置 k = 9)

```
C:\AILab2\venv\Scripts\python.exe C:/AILab2/main.py
Mat Without G1, G2: Accuracy: 0.6181818181818182  f_score: 0.7355163727959698
Mat With G1, G2: Accuracy: 0.8872727272727273  f_score: 0.9164420485175202
Por Without G1, G2: Accuracy: 0.8458149779735683  f_score: 0.9160671462829736
Por With G1, G2: Accuracy: 0.9295154185022027  f_score: 0.9596977329974811
```

可以看到，对于 KNN，如果我们选用了 G1, G2，这两个特征，其准确率会比不选用 G1，G2，这两个特征要高出很多，并且分数也会高很多，但是，无论基于不基于 G1，G2，除了第一组以外，其他的组的准确率都很高。由于并没有选全部的特征，所以可能会漏掉一些因素，因此如果把所有特征都拿来训练，准确率应该会进一步提高。从实验结果来看，KNN 用来预测分类是一个不错的模型。
利用朴素贝叶斯预测的结果：

```
C:\AILab2\venv\Scripts\python.exe C:/AILab2/main.py
NABE Mat With G1, G2: Accuracy: 0.6809090909090909  f_score: 0.8063982349696637
NABE Por With G1, G2: Accuracy: 0.8537995594713657  f_score: 0.9207107660146333

Process finished with exit code 0
```

我们可以看到，葡萄牙语的预测比数学的预测高了很多，这说明条件独立性假设可能不是完全成立，并且两个成绩的特征影响权重分布应该有较大的差别。如果将所有的特征都纳入计算，应该在准确性方面会有所提高。从实验结果来看，朴素贝叶斯模型没有 KNN 表现好，是一个较为一般的预测模型。

利用 SVM 进行预测：这里设定软边距参数 C = 1.0，Gaussian 核中σ = 1.0
使用线性核：

```
Optimal solution found.
Test success
Kernel function type:Linear
SVM Mat Without G1, G2: Accuracy: 0.32727272727272727  f_score: 0.041025641025641026
SVM Mat With G1, G2: Accuracy: 0.8909090909090909  f_score: 0.9211822660098523
SVM Por Without G1, G2: Accuracy: 0.8502202643171806  f_score: 0.9170616113744076
SVM Por With G1, G2: Accuracy: 0.9361233480176211  f_score: 0.9598965071151359
```

使用二次核：

```
Test success
Kernel function type:Quadratic
SVM Mat Without G1, G2: Accuracy: 0.5054545454545455  f_score: 0.5917159763313609
SVM Mat With G1, G2: Accuracy: 0.32  f_score: 0.010471204188481674
SVM Por Without G1, G2: Accuracy: 0.14977973568281938  f_score: 0.005128205128205128
SVM Por With G1, G2: Accuracy: 0.9118942731277533  f_score: 0.9458762886597938
```

使用 Gaussian 核：

```
Optimal solution found.
Test success
Kernel function type:Gaussian
SVM Mat Without G1, G2: Accuracy: 0.44727272727272727  f_score: 0.4076923076923077
SVM Mat With G1, G2: Accuracy: 0.8436363636363636  f_score: 0.8780487804878048
SVM Por Without G1, G2: Accuracy: 0.8480176211453745  f_score: 0.9157769869513641
SVM Por With G1, G2: Accuracy: 0.8502202643171806  f_score: 0.9170616113744076
```

由上述实验结果可以看出，使用线性核核高斯核效果较好，使用二次核效果较差，并且使用 G1，G2 作为分类参数的时候，会比没有使用 G1，G2 作为分类参数的时候效果要好很多。所以我们最终选取线性核或者高斯核，至于在某一个特定的 Testcase 上出现准确率较低的原因，一方面可能是因为训练集里的元组较少，并且我选的特征，为了便于测试，没有特别多，并可能没有非常地贴近真实的影响因素，更换特征的选择，以及特征选择的数量，可以使得准确率提高。

[特别说明]:
本实验中所用的包有三个：Pandas，Numpy，Cvxopt。
Pandas 和 Numpy 用来解决一般的数值计算和数据读入等，Cvxopt 则用来解决 SVM 中的二次规划问题，如果直接实现效率很低。

实验环境是 Windows10 + Pycharm pro + miniconda3（64 bit）
在别的环境运行时，请适当地替换 main.py 中 141，以及 142 行的数据文件的地址，这里使用的是 Windows 下的地址，并且实验进行时因为使用了 conda 的缘故，采用了绝对地址。

在安装 CVXOPT 包时，请在 conda 中使用如下命令：
Conda install pandas
Conda install numpy (如果没有在安装 Pandas 时安装 numpy 的话)
conda install -c conda-forge/label/gcc7 cvxopt （Win10）
conda install -c conda-forge cvxopt （linux）