基于慕测数据分析学生编程能力

侯锐[1]　吴耀恩[2]　徐宇轩[3]

([1] 南京大学软件学院 软件工程 181250xxx)
([2] 南京大学软件学院 软件工程 181250xxx)
([2] 南京大学软件学院 软件工程 181250xxx)

摘要：利用慕测平台一次编程作业的数据，运用一些数据处理方法，实现对于学生编程能力的评价

关键词：PCA,TOPSIS, 题目难度，编程能力

1　研究问题

1.1　背景

随着人类社会技术的进步，编程逐渐成为信息科学一个基本知识和技能，成为一个跨专业、学科所必备的一个基本素养。很多高校现在更是把编程能力的培养作为理工科学生的必修课，随之流行起来的便是在线测评系统 (OJ)。在线测评系统可以保存大量的测评数据，从中我们分析学习者的学习行为。本次研究就是基于慕测平台一次在线作业的数据，对参-+ 与该次作业的学生的编程能力做一次评价。

1.2　详细介绍

在本次研究中，一方面，我们通过主成分分析法 (PCA) 对数据中的题目做了分析，得到每个题目的难度系数，将难度系数与参与者对该题目的最终得分相结合，得到学生得分情况的测评；另一方面，我们通过 xxxx(侯锐补充) 的方法，对参与者提交的代码内容进行分析，得到学生代码书写情况的测评。考虑到不同的学生可能有不同的长处和喜好，分门别类的去评价一个学生的编程能力更加客观，我们针对每个学生，从字符串、树、图、排序等八个角度搜集学生的得分情况和代码情况，通过优劣解距离法 (TOPSIS) 实现对学生最终编程能力的打分。

1.3　应用场景

通过一次线上作业，老师对每类题型学生的掌握情况做一个大致的了解，方便改进和完善教学方法和促进与学生的沟通。作为学生，在完成一次作业后可以了解到自己与同期学生的差距在哪里，知道自己的优势和短处，有利于后期的学习和提高。

2 研究方法

2.1 数据集

基于慕测平台的提交记录数据，格式是 json 文件，内容包括每个参与者对每道题目的提交代码、最终得分、每次提交时间、每次提交得分等多个维度数据。通过学生 id 检索到具体学生，使用 pythn 提取数据，使用 numpy，pandas，sklearn 等工具分析。

2.2 数据分析方法

2.2.1 主成分分析法分析题目难度

(1) 主成分分析法

主成分分析法也称主分量分析，是把多指标转化为少数几个综合指标，其中每个主成分都能够反映原始变量的大部分信息，主要用于数据降维。其步骤大致分为以下几步：

- 整理原始矩阵 $X_{m\times n}$
- 求原始矩阵 $X_{m\times n}$ 的协方差矩阵 $S_{m\times n}$
- 求解协方差阵的特征值和特征向量
- 选取最大的 K(人为给定) 个特征值所对应的特征向量组成构成矩阵 $W_{n\times k}$
- 最后计算 $Z_{m\times k} = X_{m\times n}W_{n\times k}$

(2) 维度的选取

根据已有的数据集，先要提取特征集。在提取特征之前，我们首先对数据集中的各个特征之间的关系进行了分析，选择出六个对编程题目难度研究最为有效的特征 (X)。

1. X_1：1A 率

   在寻找合适维度的过程中，我们首先考虑到的就是题目的 AC 数量。我们随机抽取了几道题目，发现不同的题目的 AC 数量有着较为明显的差异 (见图 x)。但是我们随即结合自身考虑到这样一种情况，：当一个同学在做练习时，最终运行出正确结果，但他对自己的解法并不满意，即通过了该题但是为了优化解题过程而去多次修改代码并提交通过，这样就提高了 AC 量，使得这个特征不能那么客观地反映题目的难度类型。考虑到这个问题，我们最终选取 1A 率作为第一个维度，即一次通过的比率，用首次提交就通过的人数比总人数。

2. X_2:AC 率

   用一道题目的通过人数比上总答题人数

3. X_3: 提交次数

   一道题目的答题次数能反映这道题的难度

4. X_4: 最终平均分

   每个同学提交成绩的最高分是这个题目的最终分，那么一道题目的最终分数的平均值可以反映这道题目在学生中的大体情况

5. X_5: 提交平均分

   同学的每次提交都会得到一个及时反馈，将所有提交分数的均值作为另一个维度能一定程度上反映题目的难度

6. X_6:Debug 成效

   部分同学最后一次提交突然变为 0 然后放弃 debug，所以取最终得分而不是最后一次得分，另外也要假定同学做一道题途中不因该题过难而跳过去做另一道题然后过很久再倒回来的情况相对少见，而因题目过难而先跳过去的情况下时间会很长，因而也能反映题目难度。记一位同学最终得分 $Score_{final}$，初次提交得分 $Score_{first}$，最终提交时间 $Time_{final}$，初次提交时间 $Time_{first}$，那么

$$X_6 = \frac{Score_{final} - Score_{first}}{Time_{final} - Time_{first}}$$

(3) 数据处理与可视化

在决定最后保留几个维度的时候,我们首先通过不对 n_components 赋值,此时默认返回 min(X.shape) 个特征,这样虽然没有减少特征个数,但是可以画出累计可解释方差贡献率曲线,以此选择最好的 n_components。累积可解释方差贡献率曲线是一条以降维后保留的特征个数为横坐标，降维后新特征矩阵捕捉到的可解释方差贡献率为纵坐标的曲线，能够帮助我们选择合适的维度。从图 2-1 可以看出来，选取二维或者三维的变化是不显著的。

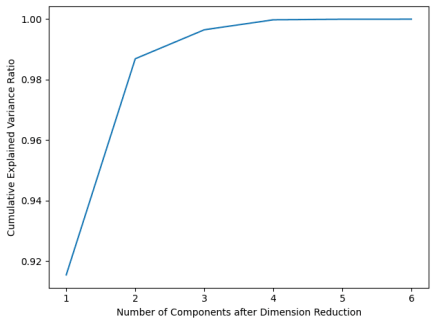

图 2-1 累积可解释方差贡献率曲线

最终 PCA 降维的结果如图 2-2 所示。

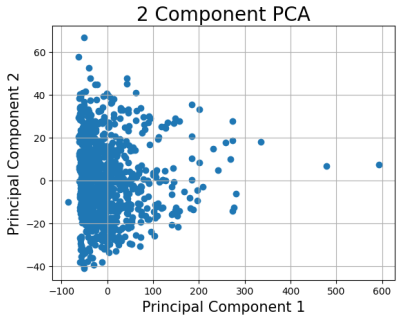

图 2-2 降维后

最后我们将两个维度加权求和，再将其指标正向化，并做标准化处理，得到一个 (0,1) 区间的系数，我们称为难度系数。我们随机抽取几道题目，如图所示，可见其难度系数是有显著差异的。

2.2.2 代码内容分析

(1) 维度的选取

根据我们当初写 python 题的经验提取了 3 个能在一定程度上反映出代码水平的特征（X）:

1. X_1：代码规范性

   代码的规范性能从一定程度上反映出一个人的代码的水平，渣的代码各有各的渣法，优雅的代码通常有共通之处。毕竟各路大神都比较强调这个。

   具体做法：借助 pylint 库来检测每个同学每道题的代码并得到一个得分作为该道题代码规范性的分数，最终取平均值作为该同学的平均代码规范得分。

2. X_2：if-elif-else 与字面常量出现的次数

   根据我们之前做这些练习题的经验，面向用例编程大概有两种不同的方向。简单点的是直接 if-elif-else(这样 if-elif-else 出现的次数会比较多), 复杂一点的就是把所有结果都放在列表里遍历列表然后用一个 if-else 即可判断出结果 (这样 if-else 出现的次数少了但字面常量出现的次数会急剧上升)，另一方面练习题并不是单靠简单的 if-else 的判断的堆积就能解出来的，字面常量的过多使用本身也不符合代码的编写规范，因此我们把这些的出现次数作为一个维度来评判代码的水平

   具体做法: 将一个学生做的所有题的代码都计算出次数并最终取平均作为该学生的最终次数

3. X_3：非 python 代码的出现次数

   由于我们是 python 练习题测试，那些直接复制粘贴网上的 C++，Java 代码 (而不是弄懂代码逻辑后自己用 python 写一遍) 且频率较高的学生的代码水平极大概率比较低，所以我们统计提交的非 python 代码的出现次数并将之作为一个维度

   具体做法:X_1 中当 pylint 库检测的代码非 python 代码时检测结果会报错，通过检索这种错误就能得知该代码是不是 python 代码。(我们发现 properties 文件记录的代码类型有少部分是错误的，所以没有直接用那个数据来统计)

(2) 数据的处理

由于选取的维度并不是直接从代码的算法上分析其水平而是从一些侧重点来反映代码水平，因此直接保留数据的绝大部分信息，降维表示代码的水平可能有失偏颇，误差较大。于是我们选取了损失信息来提高准确度的方式。仅根据这 3 个维度来将代码水平分成 3 类以期望获得较准确的结果。另一方面，物以类聚。相同代码水平的学生这些维度上表现相近的可能也较大。在分类的时候，我们最初的想法是直接分成高中低 3 类。但这样的话就要先选取部分学生的代码人工判定它是高水平还是低水平，然后再将剩余学生的代码的数据分析后往类里面塞。考虑到我们自己先来判定代码好坏的话主观因素过大，我们最终采取了另一种方式。利用数据间本身的相似特性来聚类，具有相似特征的数据如果能被聚成一类，然后我们再看这些类里的学生的代码普遍是高水平还是低水平，客观性就强多了。

最终我们使用 KMeans 算法来进行聚类分析，利用了 sklearn 里面的 KMeans 实现，将学生的代码水平分成高低 3 个档次，得到了一个学生 id 和其代码水平的字典。

2.2.3 优劣解距离法分析编程能力

(1) 优劣解距离法

C.L.Hwang 和 K.Yoon 于 1981 年首次提出 TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution)。TOPSIS 法是一种常用的组内综合评价方法，能充分利用原始数据的信息，其结果能精确地反映各评价方案之间的差距。基本过程为基于归一化后的原始数据矩阵，采用余弦法找出有限方案中的最优方案和最劣方案，然后分别计算各评价对象与最优方案和最劣方案间的距离，获得各评价对象与最优方案的相对接近程度，以此作为评价优劣的依据。该方法对数据分布及样本含量没有严格限制，数据计算简单易行。其步骤大致分为以下几步：

1. 指标属性同向化

   TOPSIS 法使用距离尺度来度量样本差距，使用距离尺度就需要对指标属性进行同向化处理（若一个维度的数据越大越好，另一个维度的数据越小越好，会造成尺度混乱）。一般情况下会选择将极小型指标、中间型指标、区间型指标转化为极大型指标 (数值越大评价越好)，在本次研究中，数值均为该种情况，无需转化。

2. 构造归一化矩阵

   假设有 n 个评价对象，每个对象都有 m 个指标，则原始数据矩阵构造为：

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_{11}} & \boldsymbol{x_{12}} & \cdots & \boldsymbol{x_{1m}} \\ \boldsymbol{x_{21}} & \boldsymbol{x_{x_{22}}} & \cdots & \boldsymbol{x_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x_{n1}} & \boldsymbol{x_{n2}} & \cdots & \boldsymbol{x_{nm}} \end{bmatrix}, \tag{2-1}$$

   构造加权规范矩阵，属性进行向量规范化，即每一列都除以当前列向量的范数:

$$Z_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2}} \tag{2-2}$$

   由此得到归一化后的标准矩阵 Z：

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z_{11}} & \boldsymbol{z_{12}} & \cdots & \boldsymbol{z_{1m}} \\ \boldsymbol{z_{21}} & \boldsymbol{x_{z_{22}}} & \cdots & \boldsymbol{z_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{z_{n1}} & \boldsymbol{z_{n2}} & \cdots & \boldsymbol{z_{nm}} \end{bmatrix}, \tag{2-3}$$

3. 确定最优方案和最劣方案

   最优方案 $\boldsymbol{Z^+}$ 由 $\boldsymbol{Z}$ 中每列的最大值组成:

$$\boldsymbol{Z^+} = (\boldsymbol{max}\{z_{11}, z_{21}, \cdots, z_{n1}\}, \boldsymbol{max}\{z_{12}, z_{22}, \cdots, z_{n2}\}, \cdots, \boldsymbol{max}\{z_{1m}, z_{2m}, \cdots, z_{nm}\}) = (Z_1^+, Z_2^+, \cdots, Z_m^+) \tag{2-4}$$

最劣方案 $\boldsymbol{Z^-}$ 由 $\boldsymbol{Z}$ 中每列的最小值组成:

$$\boldsymbol{Z^-} = (\boldsymbol{min}\{z_{11}, z_{21}, \cdots, z_{n1}\}, \boldsymbol{min}\{z_{12}, z_{22}, \cdots, z_{n2}\}, \cdots, \boldsymbol{min}\{z_{1m}, z_{2m}, \cdots, z_{nm}\}) = (Z_1^-, Z_2^-, \cdots, Z_m^-) \tag{2-5}$$

4. 计算各评价对象与最优方案和最劣方案的接近程度

$$\boldsymbol{D_i^+} = \sqrt{\sum_{j=1}^{m} w_j(\boldsymbol{Z_j^+} - \boldsymbol{z_{ij}})^2} \tag{2-6}$$

$$\boldsymbol{D_i^-} = \sqrt{\sum_{j=1}^{m} w_j(\boldsymbol{Z_j^-} - \boldsymbol{z_{ij}})^2} \tag{2-7}$$

其中 w_j 是第 j 个属性的权重，本次研究采用人为确定的方法。

5. 计算最终各个评价对象和最优方案的接近程度

$$\boldsymbol{C_i} = \frac{\boldsymbol{D_i^-}}{\boldsymbol{D_i^-} + \boldsymbol{D_i^+}} \tag{2-8}$$

$0 \leq C_i \leq 1$,$C_i \to 1$ 表示评价对象越优秀

数据可视化

3 代码介绍

3.1 开源地址

开源地址

3.2 实现逻辑

4 案例分析

5 意见和建议

6 参考文献

A 附录 1

Code Listing 1 主函数

```
import json
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from download import Download
import getQuestions

fp =  open("test_data.json", "r", encoding="UTF-8")
data = json.load(fp)

questions = getQuestions.get_questions(data)

keys = questions.keys()
X = pd.DataFrame([x for x in
                  [questions.get(key) for key in keys]])

pca = PCA()
line = pca.fit(X)
plt.plot([1, 2, 3, 4, 5, 6], np.cumsum(pca.explained_variance_ratio_))
plt.xticks([1, 2, 3, 4, 5, 6])
plt.xlabel("Number of Components after Dimension Reduction")
plt.ylabel("Cumulative Explained Variance Ratio")
plt.show()

pca = PCA(n_components=0.95)
X_reduction = pca.fit_transform(X)
print(pca.explained_variance_ratio_)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.set_xlabel('Principal Component 1', fontsize=15)
ax.set_ylabel('Principal Component 2', fontsize=15)
ax.set_title('2 Component PCA', fontsize=20)
ax.scatter([x[0] for x in X_reduction], [x[1] for x in X_reduction])
ax.grid()
plt.show()



temp =  list()
for x in X_reduction:
    temp.append(x[0] * pca.explained_variance_ratio_[0] + x[1] * pca.explained_variance_ratio_[1])
temp = np.array(temp)
print(np.column_stack((X_reduction, temp)))
```

B 附录 2

Code Listing 2 **TOPSIS**

```python
import pandas as pd
import numpy as np

def dataDirection_3(datas, x_min, x_max, x_minimum, x_maximum):
    def normalization(data):
        if data >= x_min and data <= x_max:
            return 1
        elif data <= x_minimum or data >= x_maximum:
            return 0
        elif data > x_max and data < x_maximum:
            return 1 - (data - x_max) / (x_maximum - x_max)
        elif data < x_min and data > x_minimum:
            return 1 - (x_min - data) / (x_min - x_minimum)

    return  list( map(normalization, datas))

def entropyWeight(data):
        data = np.array(data)
        # 归一化
        P = data / data. sum(axis=0)

        # 计算熵值
        E = np.nansum(-P * np.log(P) / np.log( len(data)), axis=0)

        # 计算权系数
        return (1 - E) / (1 - E). sum()

def topsis(data, weight=None):
        # 归一化

        data = data / np.sqrt((data ** 2). sum())

        # 最优最劣方案
        Z = pd.DataFrame([data. min(), data. max()], index=['负理想解', '正理想解'])

        # 距离
        weight = entropyWeight(data) if weight is None else np.array(weight)
        Result = data.copy()
        Result['正理想解'] = np.sqrt(((data - Z.loc['正理想解']) ** 2 * weight). sum(axis=1))
        Result['负理想解'] = np.sqrt(((data - Z.loc['负理想解']) ** 2 * weight). sum(axis=1))

        # 综合得分指数
        Result['综合得分指数'] = Result['负理想解'] / (Result['负理想解'] + Result['正理想解'])
        Result['排序'] = Result.rank(ascending=False)['综合得分指数']
```

```
45
46          return Result
47
48  # data = pd.DataFrame(
49  # #     {'人均专著': [0.1, 0.2, 0.4, 0.9, 1.2], '生师比': [5, 6, 7, 10, 2], '科研经费': [5000, 6000,
        7000, 10000, 400],
50  # #      '逾期毕业率': [4.7, 5.6, 6.7, 2.3, 1.8]}, index=['院校' + i for i in list('ABCDE')])
51  # #
52  # # data['生师比'] = dataDirection_3(data['生师比'], 5, 6, 2, 12)   # 师生比数据为区间型指标
53  # # data['逾期毕业率'] = 1 / data['逾期毕业率']   # 逾期毕业率为极小型指标
54  # # out = topsis(data, weight=[0.2, 0.3, 0.4, 0.1])    # 设置权系数
55  # # print(out)
```

C 附录 3

Code Listing 3 获取数据

```
1   import json
2   import numpy as np
3   import pandas as pd
4   import matplotlib.pyplot as plt
5
6   from sklearn.decomposition import PCA
7
8   from download import Download
9   import getQuestions
10
11  fp =  open("test_data.json", "r", encoding="UTF-8")
12  data = json.load(fp)
13
14  questions = getQuestions.get_questions(data)
15
16  keys = questions.keys()
17  X = pd.DataFrame([x for x in
18                    [questions.get(key) for key in keys]])
19
20  pca = PCA()
21  line = pca.fit(X)
22  plt.plot([1, 2, 3, 4, 5, 6], np.cumsum(pca.explained_variance_ratio_))
23  plt.xticks([1, 2, 3, 4, 5, 6])
24  plt.xlabel("Number of Components after Dimension Reduction")
25  plt.ylabel("Cumulative Explained Variance Ratio")
26  plt.show()
27
28  pca = PCA(n_components=0.95)
29  X_reduction = pca.fit_transform(X)
```

```
30 print(pca.explained_variance_ratio_)
31
32 fig = plt.figure()
33 ax = fig.add_subplot(1, 1, 1)
34 ax.set_xlabel('Principal Component 1', fontsize=15)
35 ax.set_ylabel('Principal Component 2', fontsize=15)
36 ax.set_title('2 Component PCA', fontsize=20)
37 ax.scatter([x[0] for x in X_reduction], [x[1] for x in X_reduction])
38 ax.grid()
39 plt.show()
40
41
42
43 temp =  list()
44 for x in X_reduction:
45     temp.append(x[0] * pca.explained_variance_ratio_[0] + x[1] * pca.explained_variance_ratio_[1])
46 temp = np.array(temp)
47 print(np.column_stack((X_reduction, temp)))
```