暨南大學
JINAN UNIVERSITY

本科课程论文

（2022—2023 学年第一学期）

论文题目：搜索、优化策略在学习任务降维中的应用

课 程 名 称：人工智能原理

课 程 类 别：专业选修

学 生 姓 名：陈宇

学　　号：2020101642

学　　院：信息科学技术学院

学　　系：计算机系

专　　业：软件工程

任 课 教 师：张佳

教 师 单 位：信息科学技术学院

2022 年 12 月 9 日

搜索、优化策略在学习任务降维中的应用

[摘 要] 本文从一个假设的科学问题开始着手，研究在一个包含大量无关冗余，甚至噪声特征，同时含有 d 个特征的数据集，如何从这 d 个特征中找到一个对标记特定标记预测有效的特征子集，在问题解决中利用搜索策略和优化策略对该科学问题的数据进行降维。

[关键词] 机器学习；特征降维；搜索策略；优化策略；遗传算法

原理表述与问题描述

问题描述：假设存在一个训练数据集，其有n个样本（记为:$\{S1, S2 \dots, Sn\}$），d个特征（记为:$\{f1, f2 \dots, fd\}$）。每个样本可表示为一个d维向量。另外，每个样本隶属于一个可能标记L。若样本与该标记相关，其值为 1。反之，其值为 0。

搜索问题是具有初始状态和目标状态的问题，每个状态可以看作一个黑盒子，其状态空间能够表示为一棵树或者是一张图，这类问题的求解是通过搜索找到一个从初始状态到目标状态的最短路径。特别的求解过程是搜索。

优化问题与搜索问题的不同之处在于问题的解不考虑路径，往往无法事先得知问题的目标状态。最重要的一点是优化问题的解是从多元的解空间中提取出来的最优选项。

数据降维是将高维空间的数据映射到低维空间，并且保留原始数据特征的基本性质。对降维的工作原理有如下形式化描述：

R^N *表示一个 N 维实值向量的集合；*

X^K *输入空间是*R^N*(N>3)的一个子集；*

Y^L*输出空间是*R^N*(N<=3)的一个子集；*

给定一个高维的原始数据集 $S=\{x_1,x_2,\dots,x_m\}\subseteq X^{K\times M}$，以及一个特征映射的假设集$\Phi: X^K \to Y^L$，降维就是找到其中的一个特征映射假设$\phi \in \Phi$，使得到一个最优的低维目标数据集:

$$T=\{y_i \mid \forall i \in [1,M], y_i=\phi(x_i)\} \subseteq Y^{L\times M}$$

并且不改变原始数据特征的基本性质。其中，ϕ称为降维的假设函数。

降维的意义在于：在原始的高维空间中，包含冗余信息和噪声信息，会在实际应用中引入误差，影响准确率；而降维可以提取数据内部的本质结构，减少冗余信息和噪声信息造成的误差，提高应用中的精度。 降维的好处直观地好处是维度降低了， 便于计算和可视化 ，其更深层次的意义在于有效信息的提取综合及无用信息的摈弃 。

1. 将问题转化为搜索问题

1.1 问题空间状态集合

对于该科学问题转化为搜索问题，在没有采取任何操作的前提下，无法得知对于标签 L 有效的特征有哪些，因此每个特征都有可能是有效特征，可以采用类似于 01 背包问题的搜索策略，对每个特征进行取舍，问题的状态空间可以抽象为一棵解空间树（如下图 1）：

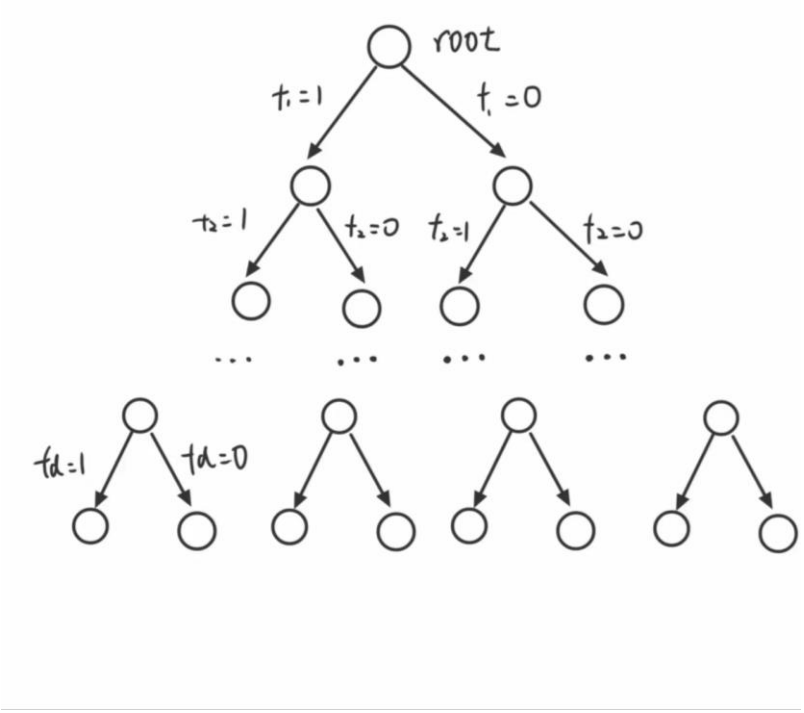

图 1：搜索子集树

对于含有 d 个特征的数据集来说，问题的解空间描述了 2^d 种可能的解。而该科学问题的**空间状态集合**就是 **d 个特征的所有子集**。

1.2 问题空间初始状态

在该问题的解空间树中，每个节点状态设立一个当前有效特征集合 E，存储当前选取的有效特征。对于该科学问题的初始状态，即智能主体的开始状态就是 **d 个特征全集本身**。初始状态有效特征集合 E 表述为 *E={f1,f2, f3, ... , fd}*。

1.3 问题空间动作集合

设 A 为该科学问题的动作集合；动作集合 A 中包含有**两个动作：保留 s 和舍去 d**；两个动作分别对应 1.1 节中的解空间树中的 fn = 1，fn = 0 两个操作。动作表示为对当前选择的特征 fn 采取保留还是舍去的执行动作。

$E_{i+1}=A_i(E_i)$表示为在当前有效特征集合 Ei 状态下执行动作 Ai，返回新的状态：有效特征集合是 Ei+1. 而由 n 个动作组成的动作序列则表示为$\{A_1(E_1),A_2(E_2),\ldots,A_n(E_n)\}$。智能主体的动作也可以称为迁移函数或者后继函数，就是说：因为动作导致状态迁移，或因为动作而生成后继状态。

1.4 问题空间目标检测

将科学问题的解空间树每个结点遍历或者减枝完全，最后保留的状态的当前有效特征集合为最终的目标状态。即检查该状态下的有效特征集合的特征数目是否减少，以及数据特征，模型的预测准确率没有被大幅度的改变。

1.5 问题空间路径代价

路径代价是指在搜索过程中添加和删除一个特征的代价。而该问题是在所有状态集合中选择最优解，没有涉及到任何路径代价的问题。

2.1 搜索策略求解

搜索问题的解是一个从初始状态到目标状态的路径，也就是智能主体寻找该路径的动作。在这里给出采用**深度优先搜索的回溯算法**来解决这个搜索科学问题。回溯搜索是一种通用的深度优先搜索算法，用于查找问题空间上的解，尤其是约束满足问题。它的工作方式是以深度优先方式递进地寻找解的候选，一旦确定该候选不是一个合法的解，就立即抛弃，原路返回，再一深度优先方式寻找下一个候选解，简单的回溯算法描述如下：

---

agent BACKTRACk-SEARCH

input problem, to be solved by CSP

output a solution, or failure

local assignment <- Null

return BACKTRACK(assignment, problem)

function BACKTRACK(assignment, csp)

```
if assignment is complete then return assignment
var <- SELECT-UNASSIGNED-VARIABLE(csp)
for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment then
        add {var = value} to assignment
        inferences <- INFERENCE(csp, var, value)
        if inferences != failure then
            add inferences to assignment
            result <- BACKTRACK(assignment, csp)
            if result = failure then return result
    remove {var = value} and inferences from assignment
end for
return failure
```

2. 将问题转化为优化问题

2.1 优化问题求解

对于一般的优化问题，通过一系列复杂的计算可以求得最优解；但是对 NP 完或者 NP 难的优化问题而言，人工智能是有条有效的途径。对于该科学问题来说，对原始的特征序列进行优化，无非就是在减少原有的特征数的同时，尽量避免整体数据特征的改变。求解方案我将采用**遗传算法**。

2.2 遗传算法简述

遗传算法是一种模仿自然选择过程的启发式搜索算法。该算法可看作是随机束搜索算法的变体，不同之处在于其后继节点是两个状态的组合而不是由单一状态所生成。它的处理过程相当于有性繁殖，而不是无性繁殖。

遗传算法属于进化算法这个大分类，它采用自然进化的方式来生成优化问题的解。而自然进化通常包含：遗传，突变， 选择， 杂交。

2. 3 遗传算法描述

---

agent GENETIC-ALGORITHM

inputs: population,

a set of individuals FITNESS-FN,

a function that measures the fitness of an individual

output: the best individual in population,

according to FITNESS-FN

repeat

new_population <- empty set

for i = 1 **to** SIZE(population) **do**

x <- RANDOM-SELECTION(population, FITNESS-FN)

y <- RANDOM-SELECTION(population, FITNESS-FN)

```
        child <- REPRODUCE(x, y)
        if (small random probability) then child <- MUTATE(child)\
        add child to new_population
    end for
    population <- new_population
until some individual is fit enough, or enough time has elapsed
return the best individual in population
```

值得注意的是，遗传算法开始时随机生成 k 个状态，称其为种群。其中每个状态为个体。

2. 真实数据集实验分析

3.1 实验设置

3.1.1 真实数据集

在本次实验中，将使用 UCI 数据集中的**心脏病二分类数据集（Heart Disease Data Set）**

网页链接：[[UCI Machine Learning Repository: Heart Disease Data Set]

数据集介绍：

数据集有 1025 行，14 列；每一行表示一个样例（病人）；前 13 列表示特征，最后一列表示分类标签：是否患有心脏病（部分数据截图如下图所示）

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

图 2：数据集部分数据

数据字段介绍：

字段名	含义	描述
age	年龄	无
sex	性别	1：男　　0：女
cp	心绞痛病史	1:典型心绞痛，2:非典型心绞痛，3:无心绞痛，4:无症状
trestbps	静息血压	单位为毫米汞柱(mm Hg)
chol	胆固醇含量	单位:mgldl
fbs	空腹时是否血糖高	如果空腹血糖大于 120 mg/dl，值为 1，否则值为
restecg	静息时的心电图特征	值 0：正常，值 1：有 ST-T 波异常（T 波倒置和/或 ST 升高或降低> 0.05 mV），值 2：

		根据 Estes 的标准显示可能或确定的左心室肥大
thalach	最大心率	无
exang	运动是否会导致心绞痛	1 表示会，0 表示不会
oldpeak	运动相比于静息状态，心电图中的 ST-T 波是否会被压平	1 表示会，0 表示不会
slope	心电图中 ST 波峰值的坡度	1:上升，2:平坦，3:下降
ca	心脏周边大血管的个数	0-3
thal	是否患有地中海贫血症	3=正常；6=固定缺陷；7=可逆缺陷
target	标签列：是否有心脏病	0 表示没有，1 表示有

3.1.2 数据集先处理

在进行搜索策略和优化策略进行实验之前，先对原数据集进行二分类学习和预测，得到模型的准确率，以便于在后续实验中进行对比和利用。在这里，我简单的使用逻辑回归模型和支持向量机模型（SVM）进行学习（结果如图 3 所示）。

```
the accuracy of the SVM is :0.8311688311688312
the accuracy of the logistic Regression is :0.8474025974025974
```

图 3：逻辑回归和 SVM 模型准确率

同时，利用 `VarianceThreshold` （移除低方差特征）的特征选择方法来先行一步简单对数据集降维：过滤掉方差在 0.6 以下的特征（结果如下图所示）。

```
原数据集中的特征数：
 13
原数据集中不同特征的方差：
 age           82.226151
sex             0.211737
cp              1.059126
trestbps      306.536058
chol         2659.190244
fbs             0.126987
restecg         0.278383
thalach       528.746971
exang           0.223296
oldpeak         1.379403
slope           0.381249
ca              1.061507
thal            0.384843
dtype: float64

方差阈值法选择的特征数：
 7
新数据集中不同特征的方差：
 [8.22261511e+01 1.05912576e+00 3.06536058e+02 2.65919024e+03
 5.28746971e+02 1.37940308e+00 1.06150720e+00]

选择特征：
 ['age', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca']
特征过滤后准确率：  0.8701298701298701
```

图 4：`VarianceThreshold` 降维结果

分析处理结果可以看出：

利用 `VarianceThreshold` 方法降维之后，选择特征数由原来的 13 减少为 7 的同时，模型对于该数据集的测试集准确率没有下降反而上升。说明数据降维，对该数据集的学习模型的效率有较大的提升，同时在不改变原有的数据性质外也能有效的去除数据冗余，加大机器学习的效率。（在后续实验中，考虑机器性能，将采用逻辑回归分析进行学习）

3.1.3 评估指标

使用了模型对数据集的测试集的准确率(Accuracy)来评估选择的特征子集的性能。

3.1.4 方法一：回溯搜索策略

在利用回溯法进行求解时，我将求解方法和求解变量封装为 **solution 类**，我利用原始的特征集合预测的准确率作为初始回溯的剪枝条件，具体表示为：

当该特征舍去后模型准确率小于该剪枝条件则保留该特征，即对舍去该特征的状态进行剪枝。

solution 类成员表：

成员名	属性	描述
df	变量	存储原数据集的 dataframe
X	变量	特征集
y	变量	标签集
attri	变量	特征字符串列表
isornot	变量	特征选择列表：1-选 0-不选
bestisornot	变量	最优特征选择列表
bestaccu	变量	最优的准确率
hierar	变量	特征数（树高）
regressionfunc	函数	计算准确率
constraint	函数	约束函数（返回 bool）
traceback	函数	回溯函数
runfunc	函数	封装其它成员函数
output	函数	输出结果成员函数

3.1.5 方法二：遗传算法优化策略

类似于搜索策略，我设计了一个 population 类，将遗传算法和其它变量函数封装起来。

算法具体细节如下：

1. 基因编码：一共有 13 个特征，每种特征的有无可以作为一个独立的基因片段。
2. 设计初始群体：为了计算方便设置初始群体为 4 个初始生物个体，随机产生
3. 适应度计算：用特征子集的模型准确率作为适应度
4. 生产下一代：被轮盘赌选中的基因需要进行基因重组产生下一代
5. 迭代计算：自然选择算法淘汰低适应度的状态，迭代退出条件：连续两代的最高适应度差值为 0 时退出

population 类成员表：

成员名	属性	描述
FINISHED_LIMIT	变量	终止限界
ACCURACY_LIMIT	变量	准确率限界
CHROMOSOME_SIZE	变量	染色体基因片段数
SELECT_NUMBER	变量	自然选择数
max_last	变量	上一代种群中最大准确率

diff_last	变量	上一代计算的改变量
is_finished	函数	算法退出模块
initchrom	函数	生成 4 条初始染色体
regressionfunc	函数	计算准确率
mapping	函数	生成 X
fitness	函数	计算适应度
filter	函数	自然选择模块
crossover	函数	交叉产生下一代：
runpopulation	函数	封装其它成员函数

3.2 实验结果

3.2.1 方法一：回溯搜索结果

回溯搜索实验结果：（见下图）

```
D:\python3.8.0\python.exe D:/作业/machinelearn/algorithm.py
选择的特征： ['age', 'sex', 'cp', 'chol', 'thalach', 'oldpeak', 'ca', 'thal']
准确率： 0.8701298701298701

进程已结束,退出代码0
```

图 5：回溯策略结果

3.2.2 方法二：遗传优化结果

遗传算法优化策略实验结果：（见下图）

```
fitnesses: [0.8409090909090909, 0.8409090909090909, 0.8409090909090909]
[[0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1], [0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1], [0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1]]
选择特征： ['sex', 'cp', 'fbs', 'thalach', 'exang', 'slope', 'ca', 'thal']
准确率： 0.8409090909090909

进程已结束,退出代码0
```

图 6：遗传优化结果

3.2 结果分析

由上述两个实验结果可知，使用回溯搜索降维后，由原来的 13 个特征减少为 8 个特征，最优特征子集为 8；使用遗传优化策略降维后，由原来的 13 个特征减少为 8 个特征，最优特征子集为 8；两个方法的最优模型预测准确率分别为 0.87 和 0.84，回溯搜索策略略微优于遗传优化结果。但由于遗传优化算法在自然选择后是在种群内随机遴选，所得到的结果有可能是局部最优；同时这是一个随机实验，因此我们无法得出哪种方法总体上更优。

课程结语：我对于降维在学习任务中的理解

总而言之，我对于降维的理解在于通过保留一些比较重要的特征，去除一些冗余的特征，减少数据特征的维度。而特征的重要性取决于该特征能够表达多少数据集的信息，也取决于使用什么方法进行降维。一般情况会先使用线性的降维方法再使用非线性的降维方法，通过结果去判断哪种方法比较合适。

使用降维的时机在于：

- 特征维度过大，可能会导致过拟合时
- 某些样本数据不足的情况（缺失值很多）
- 特征间的相关性比较大时

使用降维的优点：

- 节省存储空间；
- 加速计算速度，维度越少，计算量越少，并且能够使用那些不适合于高维度的算法；
- 去除一些冗余的特征（原数据中既有平方米和平方英里的特征--即相关性大的特征）
- 便于观察和挖掘信息（如将数据维度降到 2 维或者 3 维使之能可视化）
- 特征太多或者太复杂会使得模型过拟合。

注意到优点的同时，也要注意**缺点**：

在使用降维方法时，需要注意降维过程中可能会丢失一些信息，因此降维后的数据可能会导致模型性能下降。因此，在使用降维方法时，需要对比不同降维方法的性能，并结合具体应用场景进行选择。

所以，如何平衡优点和缺点，使得降维成为一个有挑战性的学习任务，因为它需要在保证数据特征信息尽可能完整的情况下，尽可能地将数据降低到低维空间。因此，在使用降维方法时，需要谨慎考虑数据的特征和结构，选择合适的降维方法以及参数设置。

参考文献

[1] 王文敏．人工智能原理[M] ．北京：高等教育出版社，2019：71-322．

[2] lilong117194．遗传算法求解背包问题 [EB/OL] ．(2017-12-26)[2022-12-23]
(188 条消息) 遗传算法求解背包问题_lilong117194 的博客-CSDN 博客_遗传算法背包问题

[3]函右右．UCI 心脏病数据集 Heart Disea Data Set
[EB/OL] ．(2017-12-26)[2022-12-23]
(188 条消息) UCI 心脏病数据集 Heart Disease Data Set_函右右的博客-CSDN 博客_heart 数据集

[4]第一次实验报告 实验 1-利用特征选择分析鸢尾花(iris)数据

附录

文件名	属性	描述
heart.csv	csv 文件	心脏病原始数据
dataex.py	python 文件	数据先处理
algorithm.py	python 文件	搜索策略代码文件
genetic.py	python 文件	遗传优化代码文件

dataex.py 文件：

```
import pandas as pd

import numpy as np

import warnings

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn import svm

from sklearn import metrics

from sklearn.feature_selection import VarianceThreshold



pd.set_option('display.max_columns', None)


# 忽视警告

warnings.filterwarnings("ignore")


df = pd.read_csv('data/heart.csv')
```

```
# 载入特征和标签集
X = df[['age',    #年龄
        'sex',    #性别
        'cp',     # 心绞痛病史，1:典型心绞痛，2:非典型心绞痛，3:无心绞痛，4:无症状
        'trestbps', #静息血压
        'chol',   #胆固醇含量
        'fbs',    #空腹时是否血糖高，如果空腹血糖大于 120 mg/dl，值为 1，否则值为 0
        'restecg', #静息时的心电图特征。0:正常。1:  ST-T 波有异常。2:根据 Estes 准则，有潜在的左
        'thalach', #最大心率
        'exang', # 运动是否会导致心绞痛,1 表示会，0 表示不会
        'oldpeak',# 运动相比于静息状态，心电图中的 ST-T 波是否会被压平。1 表示会，0 表示不会
        'slope', # 心电图中 ST 波峰值的坡度（1:上升，2:平坦，3:下降)
        'ca', #心脏周边大血管的个数(0-3)
        'thal']]# 是否患有地中海贫血症(3:无,6: fixed defect; 7: reversable defect)
```

```
y = df['target'] #  标签列。是否有心脏病，0 表示没有，1 表示有


# 对标签集进行编码
encoder = LabelEncoder()
y = encoder.fit_transform(y)
# print(y)


# 将数据集以 7：3 的比例，拆分为训练数据和测试数据：
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3, random_state=101)
# print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)


# 检验不同模型的准确性：
# svm
model = svm.SVC(kernel='linear', gamma=1)
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('the accuracy of the SVM is :{0}'.format(metrics.accuracy_score(prediction,test_y)))
```

```
# logistic regression
model = LogisticRegression()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('the accuracy of the logistic Regression
is :{0}'.format(metrics.accuracy_score(prediction,test_
y)))

# 特征选择
# 打印数据集中的特征数和每个特征的方差
print('原数据集中的特征数：\n', X.shape[1])
print('原数据集中不同特征的方差：\n', np.var(X, axis=0),
'\n')
# 使用 VarianceThreshold 来过滤掉方差在 0.6 以下的特征
selector = VarianceThreshold(threshold=0.6)
X_new = selector.fit_transform(X)

all_name = df.columns.values.tolist()  # 获得所有的特征名
称
```

```
select_name_index0 =
selector.get_support(indices=True)  # 留下特征的索引值，
list 格式
select_name0 = []
for i in select_name_index0:
    select_name0.append(all_name[i])


# 打印新数据集的特征数
print('方差阈值法选择的特征数：\n', X_new.shape[1])
print('新数据集中不同特征的方差：\n', np.var(X_new, axis=0),
'\n')
print('选择特征：\n', select_name0)


# 过滤后的回归分析
model = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7, random_state=0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
acc = metrics.accuracy_score(y_test, y_pred)
print('特征过滤后准确率：', acc)
```

algorithm.py 文件:

```
import pandas as pd
import numpy as np
import warnings
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics


pd.set_option('display.max_columns', None)


# 忽视警告
warnings.filterwarnings("ignore")


# 解决类
class solution:
        def __init__(self, df):
                self.df = df
                # 载入特征和标签集
                self.X = df[['age',   #年龄
                                'sex',   #性别
```

```
                                    'cp',    # 心绞痛病史，1:典型心绞痛，2:非典型心绞痛，3:无心绞痛，4:无症状
                                    'trestbps', #静息血压
                                    'chol',  #胆固醇含量
                                    'fbs',   #空腹时是否血糖高，如果空腹血糖大于 120 mg/dl，值为 1，否则值为 0
                                    'restecg', #静息时的心电图特征。0:正常。1:  ST-T 波有异常。2:根据 Estes 准则，有潜在的左
                                    'thalach', #最大心率
                                    'exang', # 运动是否会导致心绞痛,1 表示会，0 表示不会
                                    'oldpeak', # 运动相比于静息状态，心电图中的 ST-T 波是否会被压平。1 表示会，0 表示不会
                                    'slope', # 心电图中 ST 波峰值的坡度（1:上升，2:平坦，3:下降)
                                    'ca', #心脏周边大血管的个数(0-3)
                                    'thal']] # 是否患有地中海贫血症(3:无，6: fixed defect; 7: reversable defect)
                self.y = df['target'] #  标签列。是否有心脏病，0 表示没有，1 表示有
                # 对标签集进行编码
```

```
                encoder = LabelEncoder()
                self.y = encoder.fit_transform(self.y)
                self.attri = ['age','sex',
'cp',  'trestbps', 'chol',  'fbs',
'restecg','thalach','exang','oldpeak', 'slope', 'ca',
'thal']
                self.isornot =
[1,1,1,1,1,1,1,1,1,1,1,1,1]   #初始状态
                self.bestisornot = self.isornot.copy()
                self.bestaccu =
self.regressionfunc(self.X, self.y)
                self.hierar = 13


        def regressionfunc(self, X, y):
        # logistic regression
        # 将数据集以 7: 3 的比例，拆分为训练数据和测试数据:
            train_X, test_X, train_y, test_y =
train_test_split(X, y, test_size=0.3, random_state=101)
            model = LogisticRegression()
            model.fit(train_X, train_y)
            prediction = model.predict(test_X)
```

```
            return metrics.accuracy_score(prediction,
test_y)


        def constraint(self, isornot):
            # 根据 isornot list 来选择特征计算准确值
            temp_attr = []
            for (i, j) in zip(isornot, self.attri):
                if i == 1:
                    temp_attr.append(j)
            X = self.df[temp_attr]
            accur_tmp = self.regressionfunc(X, self.y)
            if accur_tmp >= self.bestaccu:
                return [True, accur_tmp]
            else:
                return [False]


        def traceback(self, dimension):
            if dimension > self.hierar:
                return
            else:
                self.isornot[dimension-1] = 0
                tmp_list = self.constraint(self.isornot)
```

```
                if tmp_list[0]:
                    self.bestaccu = tmp_list[1]
                    self.bestisornot =
self.isornot.copy()
                    self.traceback(dimension+1)
                self.isornot[dimension-1] = 1
                self.traceback(dimension + 1)


        def runfunc(self):
            self.traceback(1)
            self.output()


        def output(self):
            bestattri=[]
            for (i,j) in zip(self.bestisornot,
self.attri):
                if i==1:
                    bestattri.append(j)
            print('选择的特征：', bestattri)
            print('准确率：', self.bestaccu)



mysolution = solution(pd.read_csv('data/heart.csv'))
```

```
mysolution.runfunc()
```

genetic.py 文件：

```
import pandas as pd
import numpy as np
import warnings
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
import random


pd.set_option('display.max_columns', None)


# 忽视警告
warnings.filterwarnings("ignore")


class population:   #种群类
    #回归分析功能块
    def regressionfunc(self, X, y):
        # logistic regression
```

```
        # 将数据集以 7：3 的比例，拆分为训练数据和测试数据：
        train_X, test_X, train_y, test_y =
train_test_split(X, y, test_size=0.3, random_state=10)
        model = LogisticRegression()
        model.fit(train_X, train_y)
        prediction = model.predict(test_X)
        return metrics.accuracy_score(prediction, test_y)


    def __init__(self, X, y, limit):
        self.FINISHED_LIMIT = 0      #终止界限
        self.ACCURACY_LIMIT
=limit    #self.regressionfunc(X, y)   #准确率限界(以初始
状态的特征集做限界)
        self.CHROMOSOME_SIZE = 13
        self.SELECT_NUMBER = 4
        self.max_last = 0     #上一代种群中最大准确率
        self.diff_last = 10000  #上一次计算的改变量


    #算法退出模块
    def is_finished(self, fitnesses):
        max_current = 0
        for v in fitnesses:
```

```
            if v > max_current:
                max_current = v    #寻找该代种群最优染色体
        print('max_current:', max_current)
        diff = max_current - self.max_last    #适应度改变
大小
        #这里判断连续两代的改变量如果都等于0，则停止迭代
        if diff == self.FINISHED_LIMIT and self.diff_last
< self.FINISHED_LIMIT:
            return True
        else:
            self.diff_last = diff
            self.max_last = max_current
            return False

    def initchrom(self):    #生成4条初始染色体
        chromosome_states = []
        count = 0
        for i in range(1, 100):
            chromosome_state = []
            flag = True
            for j in range(1, self.CHROMOSOME_SIZE+1):
```

```
                chromosome_state.append(random.randint(0,
1))
            # print(chromosome_state)
            for k in chromosome_states:
                if k==chromosome_state:
                    flag = False
            if flag:
                chromosome_states.append(chromosome_stat
e)
                count += 1
                if count == 4:
                    #
print('chromosome_states:',chromosome_states)
                    return chromosome_states
            else:
                continue

    def mapping(self, chromosome_state):  #生成X
        temp_attr = []
        for (i, j) in zip(chromosome_state, attri):
            if i == 1:
                temp_attr.append(j)
```

```
        return df[temp_attr]


    def fitness(self, chromosome_states):   #计算适应度
        fitnesses = []
        for chromosome_state in chromosome_states:  #遍历所有染色体
            X = self.mapping(chromosome_state)
            fitnesses.append(self.regressionfunc(X, y))
        return fitnesses


    def filter(self, chromosome_states, fitnesses):    #自然选择模块
        # 准确率小于 accuracy_limit 被淘汰
        index = len(fitnesses) - 1
        while index >= 0:
            index -= 1
            if fitnesses[index] < self.ACCURACY_LIMIT:
                chromosome_states.pop(index)   #弹出不符合的染色体
                fitnesses.pop(index)
        # 遴选
```

```
        selected_index = [0] * len(chromosome_states)
        for i in range(self.SELECT_NUMBER-1):
            # 随机选择染色体，然后得到相应的索引
            j =
chromosome_states.index(random.choice(chromosome_states
))
            selected_index[j] += 1
        return selected_index


    #交叉产生下一代:
    def crossover(self, chromosome_states,
selected_index):
        chromosome_states_new = []
        index = len(chromosome_states)-1
        while index >= 0:  # 遍历完所有的染色体组的染色体（其
中下标-1 代表最后一个染色体的索引）
            print('index:', index)
            index -= 1
            chromosome_state =
chromosome_states.pop(index)
            print('弹出后的染色体组 chromosome_states_3: ',
chromosome_states)
```

```
            print('弹出的染色体: ', chromosome_state)
            for i in range(selected_index[index]):
                chromosome_state_x =
random.choice(chromosome_states)  # 随机选择一个染色体
                print('chromosome_state_x:',
chromosome_state_x)
                pos = random.choice(range(1,
self.CHROMOSOME_SIZE - 1))  # 随机[1, 2, 3, 4]其中的一个数
                print('pos( 随机[1, 2, 3, 4]其中的一个数):',
pos)
                chromosome_states_new.append(chromosome_
state[:pos] + chromosome_state_x[pos:])
                print('chromosome_states_new:',
chromosome_states_new)
            chromosome_states.insert(index,
chromosome_state)  # 恢复原染色体组
            print('chromosome_states_4:',
chromosome_states)
        return chromosome_states_new  # 返回得到的新的染色
体组


    def runpopulation(self):
```

```
        chromosome_states = self.initchrom()
        print('初始染色体组：', chromosome_states)
        n = 100  #迭代次数
        while n > 0:
            n -= 1
            #适应度计算
            fitnesses = self.fitness(chromosome_states)
            if self.is_finished(fitnesses):
                break
            print('1:', fitnesses)
            #遴选
            selected_index =
self.filter(chromosome_states, fitnesses)
            print('2:', selected_index)
            print('chromosome_states_2:',
chromosome_states)
            #产生下一代
            chromosome_states =
self.crossover(chromosome_states, selected_index)
            print('3:', chromosome_states)
            print(str(n)+'.......................')  #
迭代次数
```

```
        fitnesses = self.fitness(chromosome_states)
        print('fitnesses:', fitnesses)
        print(chromosome_states)
        maxfitness = 0
        maxindex = 0
        for i in range(0, len(fitnesses)):
            if fitnesses[i] > maxfitness:
                maxfitness = fitnesses[i]
                maxindex = i
        maxchromosome = chromosome_states[maxindex]
        bestattri = []
        for (i, j) in zip(maxchromosome, attri):
            if i == 1:
                bestattri.append(j)
        print('选择特征：',bestattri)
        print('准确率：', maxfitness)



if __name__ == '__main__':
    df = pd.read_csv('data\heart.csv')
    X = df[['age',  # 年龄
            'sex',  # 性别
```

```
            'cp',  # 心绞痛病史，1:典型心绞痛，2:非典型心绞痛，3:无心绞痛，4:无症状
            'trestbps',  # 静息血压
            'chol',  # 胆固醇含量
            'fbs',  # 空腹时是否血糖高，如果空腹血糖大于 120 mg/dl，值为 1，否则值为 0
            'restecg',  # 静息时的心电图特征。0:正常。1:  ST-T 波有异常。2:根据 Estes 准则，有潜在的左
            'thalach',  # 最大心率
            'exang',  # 运动是否会导致心绞痛,1 表示会，0 表示不会
            'oldpeak',  # 运动相比于静息状态,心电图中的 ST-T 波是否会被压平。1 表示会，0 表示不会
            'slope',  # 心电图中 ST 波峰值的坡度（1:上升，2:平坦，3:下降)
            'ca',  # 心脏周边大血管的个数(0-3)
            'thal']]  # 是否患有地中海贫血症(3:无，6: fixed defect; 7: reversable defect)
    y = df['target']  # 标签列。是否有心脏病，0 表示没有，1 表示有
    attri = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca',
```

```
          'thal']


def funcmodule(X, y):
    try:
        mypopulation = population(X, y, 0.78)
        print(mypopulation.ACCURACY_LIMIT)
        mypopulation.runpopulation()
    except IndexError:
        print('重新执行! ')
        raise IndexError


# 对标签集进行编码
encoder = LabelEncoder()
y = encoder.fit_transform(y)
while True:
    try:
        funcmodule(X, y)
    except IndexError:
        continue
    else:
        break
```

本科生课程论文评定表（封底）

论文评语：

评定分数（百分制）：_____　　签章：

年　月　日