廣東工業大學

QG 中期考核详细报告书

题　　目　算法实现

学　　院　信息工程

专　　业　电子信息类

年级班别　19 级四班

学　　号　3119002303

学生姓名　郑坤泽

2020 年 04 月

目录

1.K-means 算法

1.1 算法思路和实现步骤

k-means，基于距离的聚类技术，使用质心定义原型，质心为一组点的均值，作用 n 维连续空间中的对象。

步骤（伪代码）：

选择 k 个点作为初始点
While (True)
    将每个点划分到离它最近的簇中心所在的簇
    重新计算簇中心
    If 所有簇中心位置不在变化 or 达到最大迭代次数:
        停止迭代

1.2 数据集的处理

1.2.1 对数据的大致情况进行了解

使用 pandas 中 DataFrame 数据类型自带的 info 或者 describe 方法查看数据大致情况。

查看的结果：

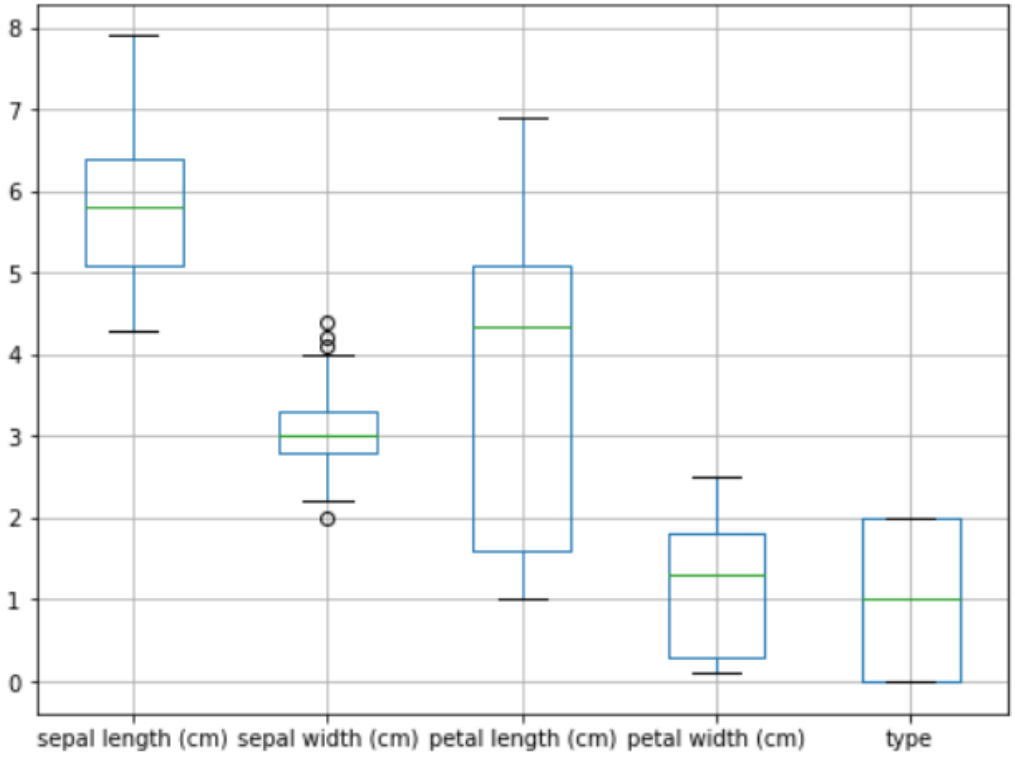
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   type               150 non-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.3 KB
```

1.2.2 数据清洗，缺失值处理等操作

这步主要是除去一些异常的数据，像一些错误的数据，如时间 2100 年之类的，还有一些异常值的发现和处理。

Iris 数据集本身比较完好不存在缺失值，错误值这种情况。至于异常值，使用箱型图观察各个特征是否存在异常值。

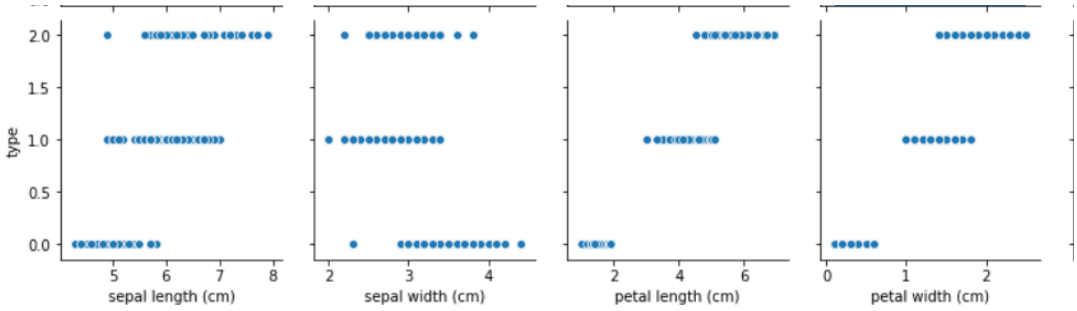
箱型图绘制结果：



可以看到 sepal width 上存在一些离群点。

1.2.3 特征选择

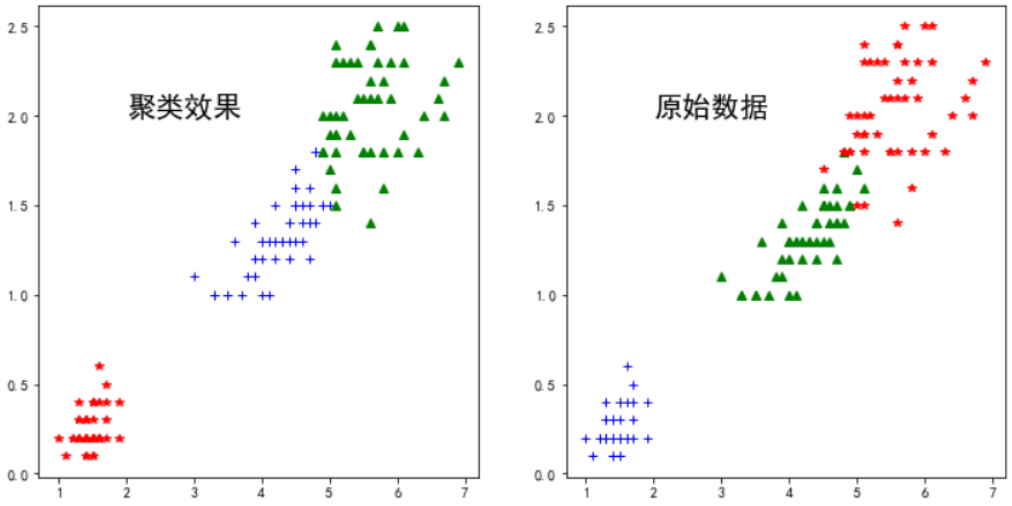
考虑到绘制图像方便，同时保证预测的准确性，这里对数据进行特征选择，绘制多变量图，观察各个特征与预测结果之间的关系。



从图中可以看到花瓣长度和花瓣宽度与最后预测的结果相关性较大。所以这里使用这两个数据进行聚类。

1.3 结果评估

结果可视化以及对比



由于聚类出来的结果标签并不是和原来数据标签一致，这里需要先处理一下预测标签。

分别计算预测结果的兰德指数和轮廓系数 **Silhouette Coefficient**

兰德指数 (Rand index)	轮廓系数 (Silhouette Coefficient)
0.8856970310281228	0.6604800085022658

1.4 不足以及可优化处

1. 度量方式

   这里只是单一的使用欧式距离，可改进增添度量方式，使得算法可以根据应用场景不同选择合适的度量方式。

2. 初始点

   距离尽可能远的点，可以先随机选择一个点，再选择离这个点最远的点，在选择离这两个点最远的点，这样类推，直到选择出 k 个点。

   或者是使用二分 k-means。

2. K 近邻算法

2.1 算法思路和实现步骤

2.1.1 思路

给定一个训练数据集，对于新的输入实例，在训练数集中找到与该实例最近的 k 个实例，这 k 个实例的多数属于某个类别，那么就把该输入实例归为这个类别。

2.1.2 实现步骤

fit 拟合训练集
    存储训练集数据以及其标签

For 单个数据样本 in 数据集：
    计算该样本点与训练集所有样本点的距离
    得出距离最近的前 K 个样本
    统计这些样本中出现最多的类别
    将这个样本归为这个类别

2.2 数据集处理

1. 大致了解数据集，使用 describe 方法查询的结果

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	type
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

2. 由于 K 近邻使用的是欧式距离，所以要对数据进行标准化，消除纲量的影响，同时使得特征之间发挥的作用相同。
3. 接着就是使用 sklearn 中自带的留出法 train_test_split 来划分数据

2.3 结果评估

	Sklearn 中的 knn	My_knn
准确率	0.9666666666666667	0.9666666666666667
混淆矩阵	[[8 0 0] [0 10 0] [0 1 11]]	[[8 0 0] [0 10 0] [0 1 11]]

2.4 不足和优化

Knn 属于非参数学习模型，每预测一个样本都需要计算一遍与所有训练集样本之间的距离，这样会导致在数据量很大时，会非常的耗时。因此可以将 knn 优化为一个类似二叉树的模型，对特征空间进行划分，这样可以提高搜索最近邻的效率。

3.线性回归

3.1 算法思路和实现步骤

1. 思路
   通过最小化预测结果和真实值之间的均方误差，来求出模型的最优参数。
2. 实现步骤
   对于最小化的总误差（即代价函数），我们可以这样表示为：

$$D = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

这里的 f(x)也就是我们的模型，而最终目标就是要求出模型中的最优参数。对于样本数据集，为了方便统一计算，我们需要实现向量化。于是，

$$X = [x_0, x_1, x_2, ...x_n], \theta = [a_0(\text{也就是原来的}b), a_1, a_2, ...a_n]^T$$

那么原来模型的式子就可以用这两个向量来表示：

$$a_1 x_1 + a_2 x_2 + ...a_n x_n + b = X\theta$$

那么代价函数可以表示为矩阵表达式（这里在前面添加一个 1/2，方便后序计算）：

$$D = \frac{1}{2}(X\theta - y)^T (X\theta - y)$$

将其展开得到：

$$D(\theta) = \frac{1}{2}(\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta - y^T y)$$

这里要利用矩阵求导的三个公式：

$$\frac{dAB}{dB} = A^T$$

$$\frac{dX^T AX}{dX} = 2AX$$

$$\frac{dX^T B}{dX} = B$$

这里要求出的是使得代价函数最小的最优参数，于是我们这里对参数进行求导（结合上面的三个式子）：

$$\frac{\partial D(\theta)}{\partial \theta} = X^T X\theta - X^T y$$

由于原来的式子很明显是一个凸函数，所以这里直接让求导结果等于 0，求出最小值处的最优参数向量：

$$\theta = (X^T X)^{-1} X^T y$$

用 Python 代码表示：

```
”””””
X 样本数据集(array 矩阵)
y 样本目标值(array 数组，一维)
np.linalg.inv 矩阵求逆
.T  矩阵转置
.dot  矩阵乘法
”””””
self.fit_theta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

于是就可以得到最后的最佳参数。

3.2 数据集处理

1. 这里使用的是波士顿房价数据集，第一步依旧是观察大概的情况
   除了上面讲过的 describe 和 info，这里还可以使用 DataFrame.head()或者是 DataFrame.tail()来查看数据头五个样本和最后的五个样本。当然不一定要拘束五行，也可以自己切片取

出自己要观察的样本。

从得出的情况来看

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

	AGE	DIS	RAD	TAX	PTRATIO	B	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	
std	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	
25%	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	
50%	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	
75%	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	

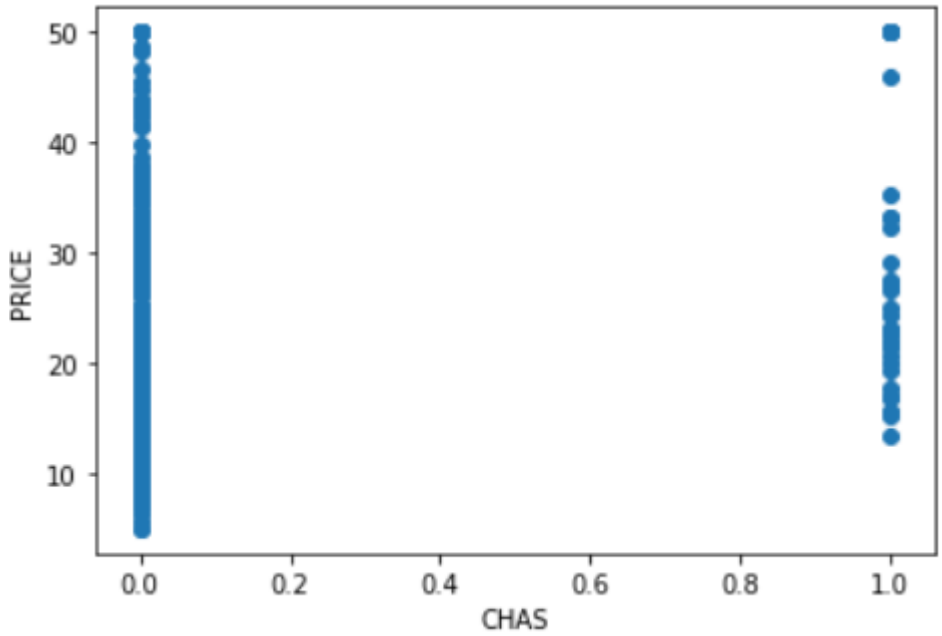
	LSTAT	PRICE
count	506.000000	506.000000
mean	12.653063	22.532806
std	7.141062	9.197104
min	1.730000	5.000000
25%	6.950000	17.025000
50%	11.360000	21.200000
75%	16.955000	25.000000
max	37.970000	50.000000

，

这些值都是数值类型，而且没有缺失值，但是有个 CHAS 特征很奇怪，最大 1，最小 0，估计是离散类型的特征， 这里留着后面绘制图像一起判断。

2. 特征选取

在这些特征里面我们要选取一些特征，上面提到 CHAS 可能是离散类型，这里先单独把它绘制出来。



从这个图像中可以看到 CHAS 这个特征确实是离散类型的，同时也可以看到这个值与最后的预测值之间的相关性不大。

这边我们可以绘制出其他变量的多变量图，同时我们也可以直接计算这些特征与预测值之间的相关系数来反映他们之间的相关性。下面是计算出来的结果：

```
CRIM:          -0.38830460858681154
ZN:          0.360445342450543
INDUS:          -0.48372516002837285
CHAS:          0.1752601771902986
NOX:          -0.42732077237328264
RM:          0.6953599470715391
AGE:          -0.3769545650045962
DIS:          0.24992873408590394
RAD:          -0.3816262306397776
TAX:          -0.46853593356776685
PTRATIO:          -0.5077866855375619
B:          0.3334608196570664
LSTAT:          -0.7376627261740147
PRICE:          1.0
```

果然一算 CHAS 的相关系数很低，当然也有很多较低的，这里会使用强相关和中等强度相关的特征（里面没有极强相关的），也就是算相关系数绝对值在 0.4 以下的特征不会被采用。

3. 划分数据

3.3 预测评估

	自制模型	Sklearn 模型
平均平方误差	28.395195174389666	28.395195174389666
平均绝对误差	3.486245132197312	3.486245132197312

3.4 不足和优化

使用正规方程在数据量较小的时候很有效，不过数据量大时正规方程计算速度慢。可以考虑使用梯度下降法，同时为了防止过拟合还可以在梯度下降法里面加上正则项。

4. 朴素贝叶斯算法

4.1 算法思想和实现步骤

1. 算法思想

贝叶斯定理：

$$P(Y=c_k|X=x)=\frac{P(X=x|Y=c_k)P(Y=c_k)}{\sum_k P(X=x|Y=c_k)P(Y=c_k)}$$

贝叶斯定理可以计算出在样本的情况下样本属于 c_k 这个类别的概率。实际中通过计算这个样本属于总类别中每个类别的概率，比较这些概率的大小，进而得出该样本属于哪一个类别。

先验概率 $P(Y=c_k)$ 是训练集中属于 c_k 这个类别占的比例。而条件概率 $P(X=x|Y=c_k)$ 表示属于 c_k 这个类别中，样本为 x 所占的比例。而分母全概率 $\sum_k P(X=x|Y=c_k)$ 在各个类别中 x 所占比例的累加。朴素贝叶斯对原来的贝叶斯公式进行了修改，对原式中的 $P(X=x|Y=c_k)$ 进行了独立性假设，即：

$$P(X=x|Y=c_k)=P(X^{(1)}=x^{(1)},....,X^{(n)}=x^{(n)}|Y=c_k)=\prod_{j=1}^{n}P(X^{(j)}=x^{(j)}|Y=c_k)$$

于是原式可以更改为：

$$P(Y=c_k|X=x)=\frac{P(Y=c_k)\prod P(X^{(j)}=x^{(j)}|Y=c_k)}{\sum_k P(Y=c_k)\prod P(X^{(j)}=x^{(j)}|Y=c_k)}$$

2. 实现步骤

由于在给定数据集的情况下分母部分的概率是相同的，所以在实现的时候就只需要计算分子部分就可以了。

(伪代码以邮件分类为例)
fit 训练数据：
For 单个词袋模型向量 in 训练集的词袋模型：
If 单个词袋模型向量 属于 正类：
正类总词袋模型 += 单个词袋模型向量
正类数量 += 1
else 负类的：
……

每一个词在正类样本中占的比例 = 正类总词袋模型 / 正类数量
每一个词在负类样本中占的比例 = 负类总词袋模型 / 负类数量
计算正类在总样本中的比例

Predict (对于单个词袋模型向量)：

计算属于正类概率 = 词袋模型向量 × 每一个词在正类样本中占的比例 ×正类的比例

计算属于负类概率 = 词袋模型向量 × 每一个词在负类样本中占的比例 × 负类比例

比较两个概率的大小，判断该样本属于那一类别。

4.2 处理数据

1. 对于给定的邮件文本，首先要将数据一个一个从文本中提取出来，分词，加上停用词处理，最后构成总的数据集。
2. 对于总数据集作集合词处理，利用集合中元素的唯一性提取不重复的词集。
3. 构建词袋模型

4.3 结果评估

结果和对应标签完全一致。

造成原因：

可能实验数据量过小。

4.4 不足以及优化

1. 上面讲到的式子中的分子部分在实际计算中还是有可能出现值为 0 的情况，所以一般会在计算条件概率的时候加上一个常数，一般称为拉普拉斯平滑

$$P(X^{(j)}=x^{(j)}|Y=c_k)=\frac{\sum_{i=1}^{N}I(x_i^{(j)}=a_{jl},y_i=c_k)+\lambda}{\sum_{i=1}^{N}I(y_i=c_k)+S_j\lambda}$$

2.计算过程多概率遇到很小的值相乘，容易出现下溢出和浮点舍入问题。

可以使用自然对数处理，虽然取值改变，不过不会影响最后结果。