****

****

**QG中期考核详细报告书**

**题    目     机器学习库的实现**

**学   院       计算机学院**

**专 业 计算机科学与技术**

**年级班别 19级（1）班**

**学 号 3119004757**

**学生姓名 许继元**

**2020年04月24日**

# 1、K-means算法

此处使用K-means算法处理Iris数据集。

* **数据集的处理**

1. 首先引入数据集
2. 接着去除数据集中表示分类结果的列
3. 然后把数据集转换为数组形式
4. data\_X = pd.read\_csv(r"iris.csv")
5. data\_X = data\_X.drop(data\_X.columns[4], axis=1)
6. data\_X = np.array(data\_X)

* **算法步骤和思想**

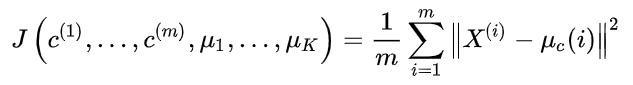
K-means算法大致思想：对于给定的样本集，按照样本之间的距离大小，将样本集划分为K个簇。

**实现步骤：**

1. 随机选取K个初始点为质心（类别）
2. 遍历每一条数据，计算其与K个质心的距离
3. 选择与之距离最近的质心作为该数据所属的类别
4. 每个簇的质心更新为该簇所有点的平均值
5. 重复2、3、4步骤，直到代价函数收敛到最小值

* **算法实现结果评估**

K-means算法的代价函数为：

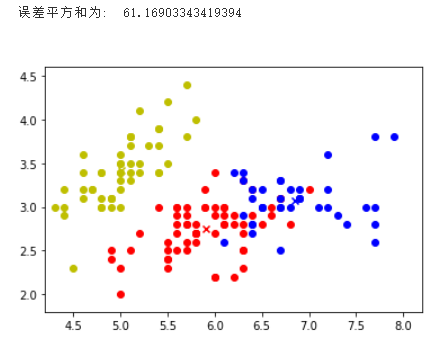


该代价函数为SSE（误差平方和）

其中是样本点，是簇，是质心（簇的均值向量）。

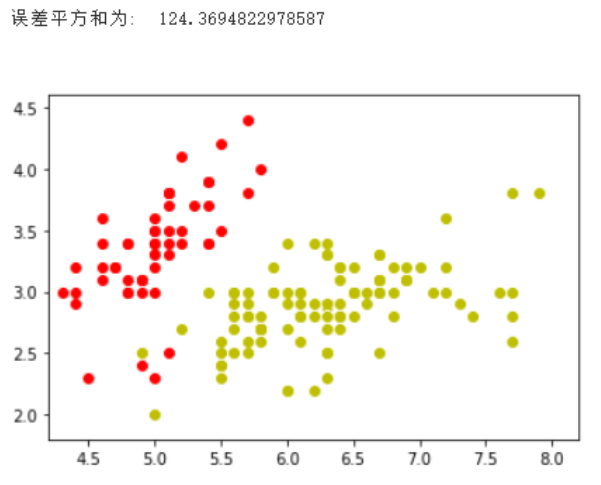
我们的优化目标为最小化该代价函数。

**在Jupyter Notebook运行并可视化如下：**



根据结果可知误差平方和为61.169左右，还是蛮大的。

当K选取为2时：



由此可见K-means算法还是存在一些不足之处，总结如下：

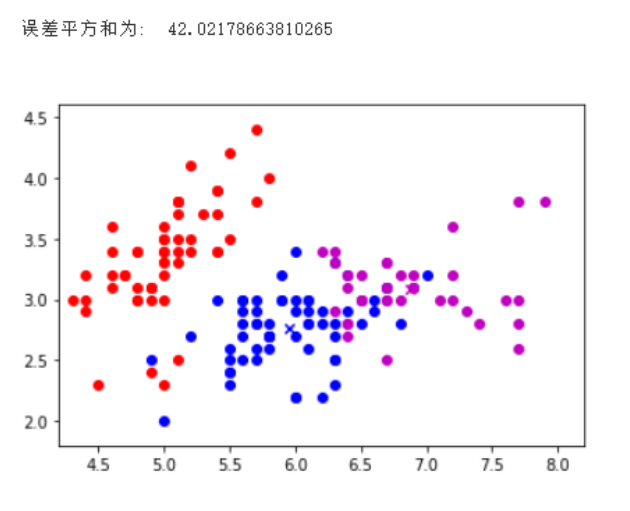
* 1. 对初值敏感，对于不同的初始值，可能会导致不同结果。
  2. 通常会收敛于局部最小值。
* **优化之处：**

为了优化K-means算法，此处尝试使用Bisecting K-means算法，也就是二分K-means算法。

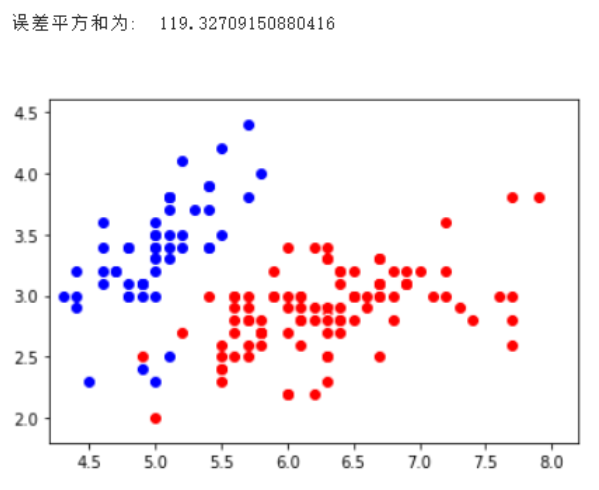
**Bisecting K-means算法 与 K-means算法的不同之处：**

Bisecting K-means算法并不是一开始就随机选择 K 个中心点，而是先把所有点归为一个簇，然后将该簇一分为二。计算各个所得簇的代价函数（SSE），选择SSE最大的簇再进行划分以尽可能地减小误差，重复上述基于SSE划分过程，直到得到用户指定的簇数目为止。

**再次运行，可视化结果如下：**



当K选取为2时：



误差平方和相对于K-means算法有所下降，可见Bisecting K-means算法在K-means算法的基础上提高了聚类的性能。

# 2、LinearRegression算法

此处使用LinearRegression算法处理housing数据集。

* **数据集的处理**

1. 首先引入数据集，同时使用正则表达式对空格符进行分割

2. 然后把数据集分为样本特征项和输出标记项

3. 接下来将数据转换为数组形式

1. data = pd.read\_csv("housing.txt", sep="\s+", header=None)
2. X = np.array(data.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,11,12]])
3. y = np.array(data.iloc[:,[13]])
4. y = y.reshape([506,])

* **算法步骤和思想**

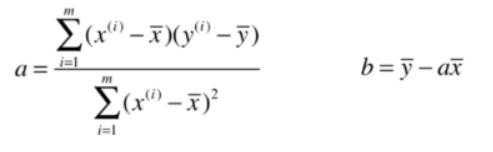
LinearRegression算法的思路大致是，寻找一条直线，最大程度地拟合样本特征和样本输出标记之间的关系。

**实现步骤：**

1.通过最小二乘法训练LinearRegression模型

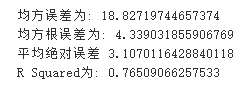
2.得出拟合数据的直线的参数

**最小二乘法公式：**



* **算法实现结果评估**

对拟合结果进行误差评估如下：

****

* **不足之处：**

可使用梯度下降算法训练LinearRegression模型。

# 3、LogisticRegression算法

此处使用LogisticRegression算法处理adult数据集(未完成)和Titanic数据集。

* **数据集的处理**

1. 首先引入数据集，给数据集添加列标签

2. 然后进行独热编码，缺失值填充等

3. 未完成

1. data = pd.read\_csv("adult.csv", sep="\s+", header=None, names=["age","workclass", "fnlwgt", "education", "education\_num", "maritial\_status","occupation", "relationship", "race", "sex", "capital\_gain", "capital\_loss", "hours\_per\_week", "native\_country", "income"])
2. data = data.dropna(how='all')
3. data.loc[data['income'] == '>50K', 'income'] = 1
4. data.loc[data['income'] == '>50K.', 'income'] = 1
5. data.loc[data['income'] == '<=50K', 'income'] = 0
6. data.loc[data['income'] == '<=50K.', 'income'] = 0

发现特征工程难度不是很小，加上时间不太充足，故更换Titanic数据集。

* 1. 首先引入Titanic数据集，用.head方法预览数据
  2. 接下来进行缺失值填充和独热编码
  3. 初步运行代码后准确率不高，尝试特征工程
  4. 构造和删去了一些特征
* **算法步骤和思想**

**LogisticRegression算法的大致思想：**

逻辑回归算法是通过概率来进行预测的。假设y是想要的结果概率，而有n个因变量会影响该概率，将n个因变量表示为x1，x2，...，xm，逻辑回归算法做的就是把这些因素对应的得分进行求和，和的结果越大表示可能性越大，反之越小。每个因变量都有对应的权值，表示为θ1，θ2，...，θm，将因变量乘以相应的权值得到的加权和就是最后的得分，最后根据得分进行归类。

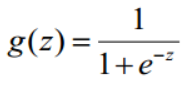
**实现步骤：**

1. 选择Sigmoid函数作为预测函数

2. 通过梯度下降法训练LogisticRegression模型

3. 对比预测结果和实际结果，计算准确率

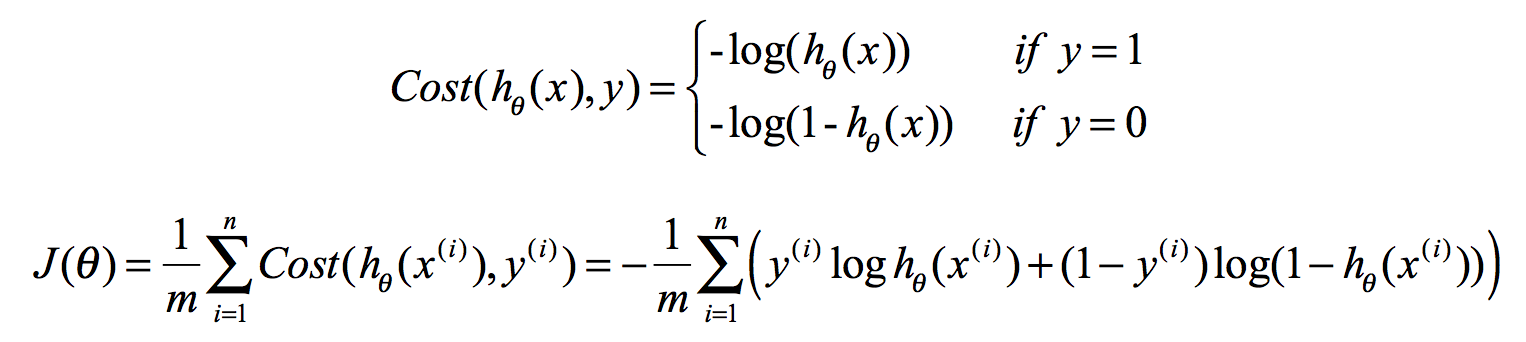
**Sigmoid函数：**

****

**函数图像：**



**逻辑回归算法的损失函数：**



它是基于极大似然估计推导得到的。

* **算法实现结果评估**

对预测结果进行准确率计算如下：

****

* **不足之处：**

1. 缺失数据可视化
2. 误差分析不够

# 4、决策树算法

此处使用决策树算法处理lenses数据集。

* **数据集的处理**

1. 首先引入数据集

2. 然后去除数据的空格符

* **算法步骤和思想**

**决策树算法思想：**

根据决策树模型，从根节点开始，根据分裂的特征和阈值进行分裂（即决策），然后由判断结果决定进入哪个分支节点，直至到达叶节点处，得到结果。

**实现步骤：**

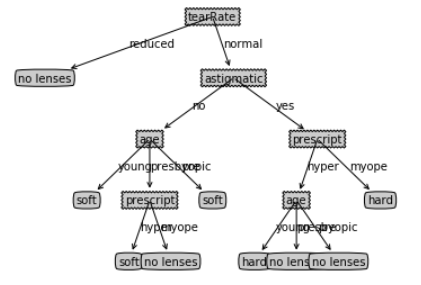
1. 根据信息增益的准则，筛选出跟分类结果相关性较高的特征，也就是分类能力较强的特征。
2. 从根节点开始，对每个节点计算所有特征的信息增益，选择信息增益最大的特征作为节点特征，根据该特征的不同取值建立子节点；然后对每个子节点使用相同的方式生成新的子节点，直到信息增益很小或者没有特征可以选择为止。
3. 主动去掉部分分支，防止过拟合。（未实现）

**此处基于ID3算法进行最优决策：**

ID3算法的思路：信息增益最大化  
信息增益 = 熵 - 条件熵

* **算法实现结果评估**

参考于《机器学习实战》，运行结果如下：



{'tearRate': {'reduced': 'no lenses', 'normal': {'astigmatic': {'no': {'age': {'young': 'soft', 'presbyopic': {'prescript': {'hyper': 'soft', 'myope': 'no lenses'}}, 'pre': 'soft'}}, 'yes': {'prescript': {'hyper': {'age': {'young': 'hard', 'presbyopic': 'no lenses', 'pre': 'no lenses'}}, 'myope': 'hard'}}}}}}

* **不足之处：**

1. 可以使用CART算法和C4.5算法
2. 尝试随机森林算法

# 5、BP神经网络算法

此处使用BP神经网络算法处理MNIST数据集。

* **数据集的处理**

1. 从<http://yann.lecun.com/exdb/mnist/index.html>下载数据集

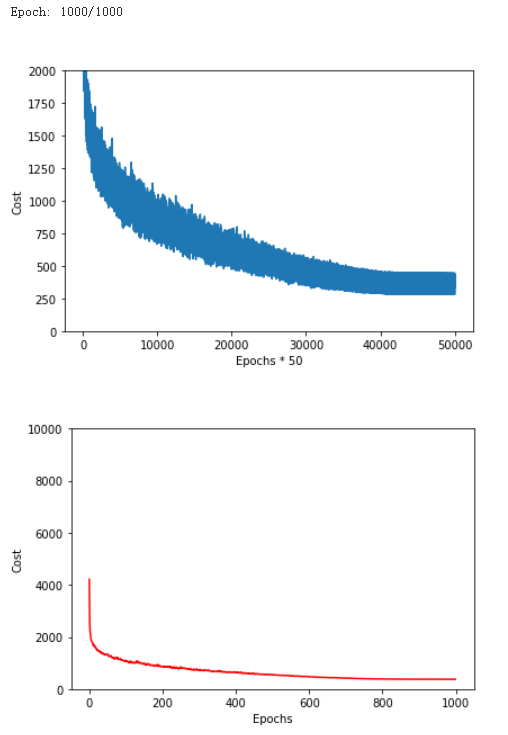
* **算法步骤和思想**

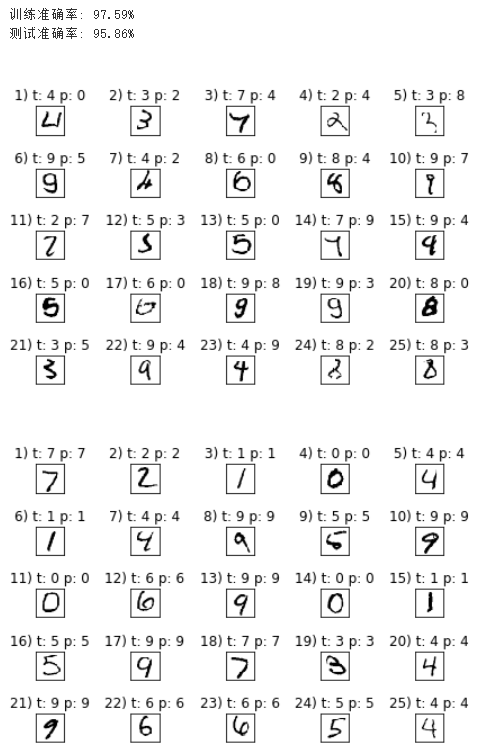
**算法步骤：**

1. 构建一个神经网络，然后随机初始化权重。（通常把权重初始化为很小的值，接近于零）
2. 然后执行前向传播算法，（也就是对于该神经网络的任意一个输入x^ (i)，计算出对应的h(x^ (i))值，也就是一个输出值y的向量）
3. 接下来计算出代价函数J(θ)
4. 然后执行反向传播算法来算出这些偏导数项，也就是J(θ)关于参数θ的偏导数
5. 使用梯度下降法进行最优化，计算出偏置和权重

* **算法实现结果评估**

参考于《Neural Networks and Deep Leaming》，运行结果如下：





* **不足之处：**

1.基于数学原理的推导上不够熟悉

2.相关代码还需要多阅读