基于线性回归与逻辑回归分类葡萄酒类型与预测葡萄酒品质

1 问题描述

葡萄酒是一种广受人们欢迎的含酒精饮品，按照其制作原材料可以被分为红葡萄酒与白葡萄酒两类。由于制作工艺与产生的原材料的不同，导致两类葡萄酒的风味也具有显著的差别，适用于在不同的场合下品尝。同时，葡萄酒的特有的风味来源于其中含有的各类化学物质与其物理的属性，也就是说其理化性质决定了其评测质量的参考与依据。从而，通过分析葡萄酒中含有的各类风味物质的含量与其物理特性，我们能够区分出葡萄酒的种类，并且预测其质量指数。

因此，在本文中，基于两个葡萄牙北部的葡萄酒质量数据集，我们通过线性回归与逻辑回归两种模型，根据不同种类葡萄酒之间的理化特点，对葡萄酒类型进行分类。同时，利用上述两种方法，选取合适的理化参数，对其风味与质量进行回归预测，以检测葡萄酒的优劣。

2 算法策略

2.1 特征归一化

由于数据中各类特征的数值各不相同，甚至部分特征数据之间存在有数量级之间的区别，这一现象会导致个参数之间方差与均值之间的不平衡。从而，为避免以上问题，我们对特征进行缩放，使其位于同一范围内。特征归一化方法如下：

$$Z = (X - \mu)/\sigma$$

其中，μ为样本均值,σ为样本的标准差。

2.2 基于梯度下降法的线性回归

线性回归算法线性函数拟合大型数据集。从而，在线性回归模型的建构过程中，我们首先定义线性函数为：

$$g(x) = w^T x + b$$

其中，样本为$x = (x_1, x_2 \dots x_n)$，权重为$w^T = (w_1, w_2 \dots w_n)^T$，偏移量为$b$

在本项目中，考虑梯度下降法拟合线性函数。从而，使用代价函数评估当前模型的质量。代价函数的方法如下所示：

$$J(w) = \frac{1}{2n} \sum_{i=1}^{n} (g_w(x)^{(i)} - y^{(i)})^2$$

求解代价函数的偏导数：

$$\frac{\partial J(w)}{\partial w_j} = \frac{1}{2n} \sum_i 2\left(y^i - z^i\right) \frac{\partial \left(y^i - z^i\right)}{w_j} = -\frac{1}{n} \sum_i \left(y^i - z^i\right) x_j^i$$

我们将代价函数的导数乘以学习率η，然后用参数w的当前值减去它，获得新的更新参数w完成梯度下降过程：

$$w_j = w_j - \eta \frac{\partial J(w)}{\partial w_j}, j = 0, 1, 2 \dots n$$

2.3 基于梯度下降法的逻辑回归

考虑到线性回归模型无法解决线性不可分的分类任务。从而，为解决上述问题，在回归的线性函数基础上，引入$Sigmoid$函数（逻辑函数），将线性函数映射到$[0, 1]$区间之内。$Sigmoid$函数表示为：

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)}$$

对于二元逻辑回归而言，$P(y = 1) = \sigma(z), P(y = 0) = 1 - \sigma(z)$。从而，我们可以写出逻辑回归的似然函数。

$$L(w) = \prod_{i=1}^{m} P\left(y^{(i)} \mid x^{(i)}; w\right) = \prod_{i=1}^{m} \left(\sigma\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - \sigma\left(x^{(i)}\right)\right)^{1 - y^{(i)}}$$

对似然函数取对数得到：

$$l(w) = \log L(w) = \sum_{i=1}^{m} \left(y^{(i)} \log\left(\sigma\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(x^{(i)}\right)\right)\right)$$

从而， 构建交叉熵损失函数：

$$J(w) = -\frac{1}{m} l(w) = -\frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \log\left(1 + e^{-w^T x^{(i)}}\right) - \left(1 - y^{(i)}\right) \log\left(1 + e^{w^T x^{(i)}}\right)\right)$$

对损失函数求偏导，计算下降梯度：

$$\frac{\partial}{\partial w_j} J(w) = \frac{1}{m} \sum_{i=1}^{m} \left(\sigma\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

我们将代价函数的导数乘以学习率η，然后用参数w的当前值减去它，获得新的更新参数w完成梯度下降过程：

$$w_j = w_j - \frac{\partial}{\partial w_j} J(w) = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\sigma \left(x^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

2.4 分类判别条件

在分类问题中，将红、白两种类别转变为（0，1）标签，放入到模型中运算分析，以下设定标签判别函数。

（1）线性回归

在线性回归模型中，以0作为判别边界，故设定标签判别函数为：

$$\phi(g(x)) = \begin{cases} 1, g(x) \geq 0 \\ 0, otherwise \end{cases}$$

（2）逻辑回归

在逻辑回归模型中，以0.5作为判别边界，故设定标签判别函数为：

$$\phi(g(x)) = \begin{cases} 1, g(x) > 0.5 \\ 0, otherwise \end{cases}$$

3 代码实现

3.1 数据集导入与训练集、测试集划分

利用$pandas$库中的文件操作函数，读入数据集。之后，以80%训练集、20%测试集的形式随机从读取的数据集中随机划分，得到实验中使用的训练集与测试集。

（1）分类问题数据集操作

```
import numpy as np
import pandas as pd

# 导入数据集并拼接为一个data_frame，颜色标签：red=1，white=0，并随即生成训练集与测试集
def load_data(df1, df2):
    df_red = pd.read_csv(df1, sep=";")
    df_red.columns = ["fixed acidity", "volatile acidity", "citric acid",
"residual sugar", "chlorides",
                      "free sulfur dioxide", "total sulfur dioxide", "density",
"pH", "sulphates", "alcohol", "quality"]
    df_red["type"] = 1

    df_white = pd.read_csv(df2, sep=";")
    df_white.columns = ["fixed acidity", "volatile acidity", "citric acid",
"residual sugar", "chlorides",
                        "free sulfur dioxide", "total sulfur dioxide",
"density", "pH", "sulphates", "alcohol",
                        "quality"]
    df_white["type"] = 0

    df_red = np.array(df_red)
    df_white = np.array(df_white)

    df_red_train = df_red[:int(df_red.shape[0] * 0.8), :]
    df_red_test = df_red[int(df_red.shape[0] * 0.8) + 1:-1, :]
    df_white_train = df_white[:int(df_white.shape[0] * 0.8), :]
    df_white_test = df_white[int(df_white.shape[0] * 0.8) + 1:-1, :]
```

```
    train_set = np.concatenate((df_red_train, df_white_train))
    test_set = np.concatenate((df_red_test, df_white_test))

    return train_set, test_set

# 导入数据,进行分类问题处理
red_wine = r".\winequality-red.csv"
white_wine = r".\winequality-white.csv"

train_data, test_data = load_data(red_wine, white_wine)

tr_data = train_data[:, :train_data.shape[1]-2]
tr_label = train_data[:, train_data.shape[1]-1:]
te_data = test_data[:, :test_data.shape[1]-2]
te_label = test_data[:, test_data.shape[1]-1:]

tr_label = tr_label.T[0, :]
te_label = te_label.T[0, :]
```

(2)回归预测问题数据集操作

```
import pandas as pd
import numpy as np


# 导入数据集并拼接为一个data_frame，颜色标签：red=1，white=0，并随即生成训练集与测试集
def load_data(df1):
    df = pd.read_csv(df1, sep=";")
    df.columns = ["fixed acidity", "volatile acidity", "citric acid", "residual
sugar", "chlorides",
                  "free sulfur dioxide", "total sulfur dioxide", "density",
"pH", "sulphates", "alcohol", "quality"]
    df = np.array(df)

    train_set = df[:int(df.shape[0] * 0.8), :]
    test_set = df[int(df.shape[0] * 0.8) + 1:-1, :]

    return train_set, test_set

# 导入数据,进行回归问题处理
red_wine = r".\winequality-red.csv"
train_data, test_data = load_data(red_wine)

tr_data = train_data[:, :train_data.shape[1] - 1]
tr_label = train_data[:, train_data.shape[1] - 1:train_data.shape[1]]
te_data = test_data[:, :test_data.shape[1] - 1]
te_label = test_data[:, test_data.shape[1] - 1:test_data.shape[1]]
```

3.2 特征归一化

对导入数据中的特征参数归一化，代码实现如下所示：

```
# 特征归一化
mu = []
std = []
```

```
def normalize(data):
    for i in range(0, data.shape[1] - 2):
        data[:, i] = ((data[:, i] - np.mean(data[:, i])) / np.std(data[:, i]))
        mu.append(np.mean(data[:, i]))
        std.append(np.std(data[:, i]))

#归一化特征参数
normalize(tr_data)
normalize(te_data)
```

3.3 线性回归模型

基于2.2中讨论的线性回归模型的推导过程，完成基于梯度下降法的线性回归模型的代码实现。

```
# 回归模型
class liner_regression:
    def __init__(self, data, lr, epoch, label):
        self.lr = lr
        self.epoch = epoch
        self.data = np.hstack((np.ones((data.shape[0], 1)), data))
        self.weight = np.zeros((self.data.shape[1], 1))
        self.loss = []
        self.label = np.reshape(label, (label.shape[0], 1))

    def c_loss(self):
        return np.dot((np.dot(self.data, self.weight)-self.label).T,
(np.dot(self.data, self.weight)-self.label))/(2*self.label.shape[0])

    def train(self):
        for i in range(self.epoch):
            temp = np.dot(self.data, self.weight)
            cost = (1/self.data.shape[0])*np.dot(self.data.T, (temp-self.label))
            self.weight = self.weight-self.lr*cost
            self.loss.append(self.c_loss())
        return self

#分类问题
    def predict_classification(self, test):
        return np.where(np.dot(test, self.weight[1:]) + self.weight[0] >= 0.0,
1, 0)

#回归问题
    def prediction(self, test):
        result = []
        temp = np.dot(test, self.weight[1:]) + self.weight[0]
        result.append(temp)
        return result

#运行线性回归模型
demo = liner_regression(lr=0.001, epoch=1000, data=tr_data,
label=tr_label).train()
result = demo.prediction(te_data)
result2 = demo.predict_classification(de_data)
acc = demo.f_acc(te_data, te_label)
```

3.4 逻辑回归模型

基于2.3中讨论的逻辑回归模型的推导过程，完成基于梯度下降法的逻辑回归模型的代码实现。

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# 定义逻辑回归模型
class logistic_regression:
    def __init__(self, lr, epoch, data):
        self.lr = lr
        self.epoch = epoch
        self.data = np.hstack([np.ones((len(data), 1)), data])
        self.weight = np.random.randn(self.data.shape[1])
        self.loss = []

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    # 定义loss function
    def j_loss(self, label):
        pred = self.sigmoid(np.dot(self.data, self.weight))
        return - np.sum(label * np.log(pred) + (1 - label) * np.log(1 - pred)) /
label.shape[0]

    def d_j(self, label):
        pred = self.sigmoid(np.dot(self.data, self.weight))
        return np.dot(self.data.T, (pred-label))/label.shape

    # 权重梯度计算
    def train(self, label):
        """

        :rtype: object
        """
        for i in range(self.epoch):
            self.weight = self.weight - self.lr*self.d_j(label)
            self.loss.append(self.j_loss(label))
        return self

    # 预测结果
    def prediction(self, test):
        test_in = np.hstack([np.ones((len(test), 1)), test])
        pred_out = sigmoid(np.dot(test_in, self.weight))
        return pred_out

    # 分类结果
    def prediction_classification(self, test):
        test_in = np.hstack([np.ones((len(test), 1)), test])
        pred_out = self.sigmoid(np.dot(test_in, self.weight))
        pred_out = np.array(pred_out >= 0.5, dtype='int')
        return pred_out

    def f_acc(self, test, t_label):
        pred_out = self.prediction_classification(test)
        acc = np.sum(pred_out == t_label)/t_label.shape
        return acc
```

```
#运行逻辑回归模型
demo = logistic_regression(lr=0.01, epoch=10, data=tr_data).train(tr_label)
result = demo.prediction_classification(te_data)
result2 = demo.prediction(te_data)
```

3.5 回归模型性能评价

在模型性能评价的指标选取过程中，我们选择了平均绝对误差(MAE)，均方误差（MSE），均方根误差（RMSE）与R2_score作为回归模型的模型性能评价指标。

```
# 性能测试函数，包括：平均绝对误差(MAE)，均方误差（MSE），均方根误差（RMSE）与R2_score
def MAE(label, pre):
    return np.mean(np.abs(label - pre))


def MSE(label, pre):
    return np.mean((label - pre) ** 2)


def RMSE(label, pre):
    return np.sqrt(MSE(label, pre))


def R2(label, pre):
    u = np.sum((label - pre) ** 2)
    v = np.sum((label - np.mean(label)) ** 2)
    return 1 - (u / v)
```

4 模型表现

4.1 葡萄酒分类任务的模型表现

在葡萄酒类型分类任务中，我们选择了固定酸度，挥发性酸度，柠檬酸，残糖，氯化物，游离二氧化硫，总二氧化硫，密度，pH，硫酸盐含量与酒精含量这11种理化参数作为特征数据，以红、白两种葡萄酒类型作为分类标准，以（0，1）作为分类标签，希望通过以上的理化特征，利用线性回归模型与逻辑回归模型完成对葡萄酒种类的区分。

4.1.1 线性回归模型在分类问题中的表现

在模型中设定学习率为0.01，训练轮数为2500轮。由Fig1可知，在训练50轮之后，线性回归模型的损失函数收敛，表明此时模型的参数已经基本不再迭代，模型收敛。

之后，分别以固定学习率为0.01，训练轮数为[0,2500]，间隔数为100，观察随训练轮数变化所得到的模型分类准确率的变化。线性回归模型在训练300轮之后的分类准确性达到稳定，模型收敛。线性回归模型对红、白葡萄酒的二分类问题的**分类准确性为0.5259**。考虑到二分类问题的模型分类准确率基线为0.5，可以分析得到线性回归模型在葡萄酒类型分类任务中的表现较差，无法通过已有的理化特征参数实现分类任务。

4.1.2 逻辑回归模型在分类问题中的表现

在逻辑回归模型中设定学习率为0.01，训练轮数为2500轮。由Fig3可知，在训练100轮之后，线性回归模型的损失函数收敛，表明此时模型的参数已经基本不再迭代，模型收敛。

之后，分别以固定学习率为0.01，训练轮数为[0，5000]，间隔数为100，观察随训练轮数变化所得到的模型分类准确率的变化。当逻辑回归模型训练1000轮之后，模型的分类准确性达到稳定，模型收敛。逻辑回归模型对红、白葡萄酒的二分类问题的**分类准确性为0.98765432。**模型的分类准确性接近于1，这一结果表明逻辑回归模型能够有效的基于葡萄酒的各类理化特征参数，对葡萄酒的种类进行区分，也体现出逻辑回归模型对二分类任务的有效性与准确性。

4.2 葡萄酒品质预测任务的模型表现

在葡萄酒品质预测过程中，我们选择上述11种理化特征指标作为训练参数，将葡萄酒的评价质量作为预测的指标，利用线性回归模型与逻辑回归模型对单一类型的葡萄酒质量进行预测。之后，利用在3.5中提到的平均绝对误差(MAE），均方误差（MSE）,均方根误差（RMSE）与R2_score作为回归模型的模型性能评价指标，对训练完的模型表现进行评价。

4.2.1 线性回归模型在回归预测中的表现

在线性回归模型中，在回归预测任务中，设定固定学习率0.001，训练轮数为250000轮，间隔数为10000，观察随训练轮数变化所得到得到模型预测表现的变化。

Fig5 线性回归模型性能评价指标与训练轮数的关系

	MAE	MSE	RMSE:	R^2
线性回归	0.507	0.449	0.670	0.253

表一 线性回归模型性能评价指标

如Fig5所示，在训练轮数达到200000轮之后，模型各评价指标达到稳定，模型收敛。线性回归模型对葡萄酒品质预测问题的**R2_score为0.253**，**平均绝对误差(MAE）为0.507,均方误差（MSE）为0.449,均方根误差（RMSE）为0.670。**模型的R2_score处于较为合适的范围中，同时MAE、MSE与RMSE的值都较小，表明模型能够在一定程度上基于现有的理化特征预测葡萄酒品质的好坏。

4.2.2线性回归模型在回归预测中的表现

在逻辑回归模型中，在回归预测任务中，设定学习率为0.001，训练轮数为5000轮，以100轮为间隔，观察随训练轮数变化所得到的模型预测表现得变化。在训练完成后，将各模型性能评价指标与训练轮数之间的关系绘制成下图所示。

	MAE	MSE	RMSE:	R^2
逻辑回归	0.0032	0.00326	3.9e-05	--8.9e-05

表二 逻辑回归模型性能评价指标

如Fig5与表二中的数据所示，虽然逻辑回归对于评价标准MAE、MSE与RMSE的值都极小，但对于关键的R2_score却为负数，这表明逻辑回归模型无法通过当前已有的理化特征参数预测葡萄酒的品质，无法完成回归预测。

5 总结与展望

在本文中，以两个葡萄酒理化性质数据集为基础，我们利用线性回归与逻辑回归的方法，通过对葡萄酒理化性质的特征参数的运算，对葡萄酒的类型进行分类，并基于理化特征参数的计算，预测葡萄酒的品质好坏。通过对模型的实验，可以得出以下结论：

- 在分类任务中，逻辑回归模型能够有效的基于已有理化特征参数，对葡萄酒的种类进行区分，在分类任务中表现出色。而对比之下，线性回归模型无法有效的基于已有特征完成分类任务，在分类任务中表现较差。

- 在回归预测任务中，线性回归模型能够在一定程度上通过已有的理化特征预测葡萄酒的品质好坏。而逻辑回归模型的R2_score为负数，表明该模型无法通过已有的特征参数完成回归预测任务。

参考文献

[1]P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.