机器学习课题报告

1. 题目：使用Python实现对数几率回归模型
2. 问题描述：
3. 下载 Iris 数据集，并读懂数据集的相关说明，了解数据格式。
4. 使用python编程，实现对数几率回归模型，并对 Iris 数据集进行分类。
5. 验证模型的效能：

将数据集的 50%作为训练集，50%作为测试集，检验模型在测试集上的分类正确率

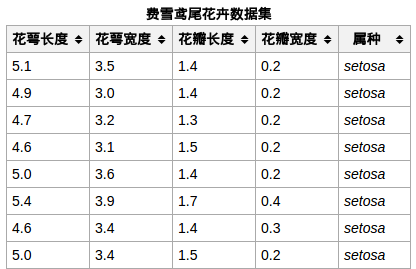
将数据集的 70%作为训练集，30%作为测试集，检验模型在测试集上的分类正确率

将数据集的 90%作为训练集，10%作为测试集，检验模型在测试集上的分类正确率

1. 数据集描述：

iris数据集包含150个样本，是一个150行5列的二维表，对应数据集的每行数据。每行数据包含每个样本的四个特征和样本的类别信息。

四个特征（前4列）分别是：Sepal.Length花萼长度、Sepal.Width花萼宽度、Pepal.Length花瓣长度、Pepal.Width花瓣宽度。类别变量分别对应鸢尾花的三个亚属，分别是：Iris-setosa山鸢尾，Iris-versicolor变色鸢尾和Iris-virginica维吉尼亚鸢尾。



1. 实验过程：
2. 导入iris数据集，并进行可视化

import numpy as np

import matplotlib.pyplot as plt

from sklearn import model\_selection

#数据导入

data\_t=[]

data\_f=np.loadtxt('E:/biancheng/ml/iris.data',delimiter=",",usecols=(0,1,2,3),dtype=float)

data\_s=np.loadtxt('E:/biancheng/ml/iris.data',delimiter=",",usecols=4,dtype=str)

for i in range(0, len(data\_s)):

if data\_s[i] =='Iris-setosa':

data\_t.append(1) #山鸢尾标记为1

else:

data\_t.append(0)

data=np.insert(data\_f,4,values=data\_t,axis=1)

x=data[:100,1:3] #选取其中两个特征

y=data[:100,4] #属种

m,n=np.shape(x)

1. 数据可视化：(绘制散点图显示数据，选取前一百行数据，X包含（Width, Length）两个变量，y为鸢尾花种类分类（二分），由此生成.csv数据文件，在Python中用Numpy读取数据并采用matplotlib库可视化数据)

seto=data[:50]

vers=data[51:100]

f1=plt.figure(1)

plt.title('Iris')

plt.xlabel('Width')

plt.ylabel('Length')

plt.scatter(seto[:,1],seto[:,2],marker='o',color='c',s=100,label='seto')

plt.scatter(vers[:,1],vers[:,2],marker='o',color='k',s=100,label='vers')

plt.legend(loc='upper right')

plt.show()

1. Sigmoid函数：即逻辑斯蒂回归分类器的基础模型

#Sigmoid

def sigmoid(x,beta): #课本3.1已经证得对数似然函数是凸函数，存在最优解

return 1.0/(1+np.math.exp(-np.dot(beta,x)))

1. 梯度下降法：

def Gradientdescent(x,y,theta): #theta是对变量应用逻辑函数之后的值

loop\_max=500 #最大迭代次数

alpha=0.01

xT=np.transpose(x)

for i in range(0,loop\_max):

hypothesis=np.dot(x,theta)

loss=hypothesis-y

gradient=np.dot(xT,loss)/m

theta=theta-alpha\*gradient

return theta

1. 建立Sigmoid预测函数，对测试集数据进预测

#预测函数

def predict (x,theta):

m,n=np.shape(x)

y=np.zeros(m)

for i in range(m):

if sigmoid(x[i],theta)>0.5:y[i]=1;

return y

1. 训练集和测试集：(取两列数据（即特征花萼长度与宽度）进行对数几率回归的分类。用train\_test\_split函数将原始数据集按5:5的比例分成训练集与测试集)

#划分训练集与测试集

m,n=np.shape(x)

theta = np.zeros(n)

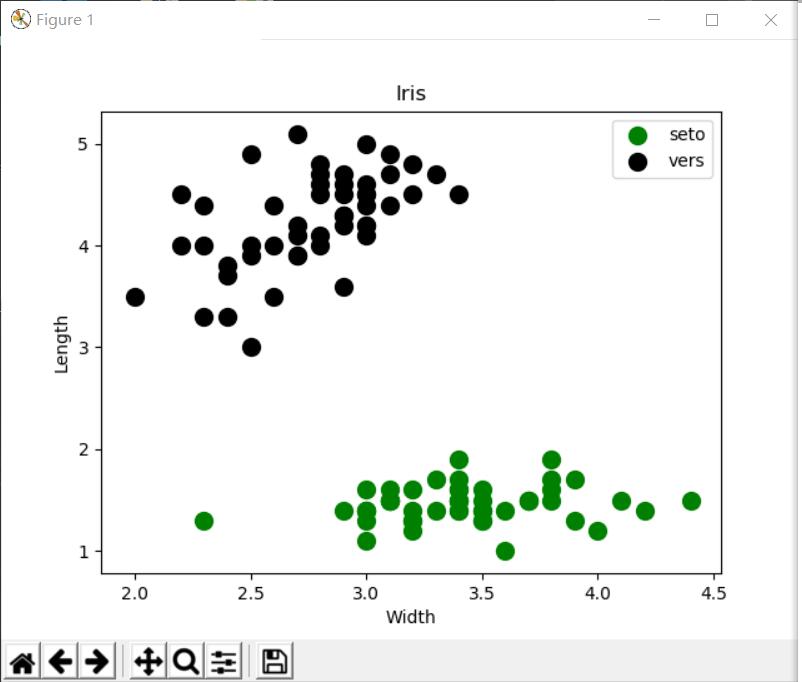
np.ones(n)

x\_train,x\_test,y\_train,y\_test=model\_selection.train\_test\_split(x,y,test\_size=0.5,random\_state=0)

theat=Gradientdescent(x\_test,y\_test,theta)

y\_p=predict(x\_test,theat)

print(y\_p)

1. 实验结果图：
2. 
3. 实验结果分析;

可看出线性回归有如下的特点：

优点：实现简单，计算简单

缺点：不能很好地拟合非线性数据

应用场景：线性回归简单、易于使用，但是现实生活中数据的特征和目标之间并不是简单的线性组合，所以并不能很好的解决具体问题。所以线性回归常用于数据特征稀疏，并且数据过大的问题中，可以通过线性回归进行特征筛选。

再者机器学习的算法非常多，但是不是拿来就用，应该首先了解算法的特性，使用范围，再看数据来源，什么特点，这个算法是否符合这种数据，再决定使用哪种模型。