实验手册

试验任务：对散点数据的处理

使用模型：非线性逻辑回归

其中有：

对数据的处理：

数据升维

train\_1 = np.insert(train[:,:-1],0,[1],axis = 1)

test\_1 = np.insert(test,0,[1],axis = 1)

#添加高次项

temp\_1 = train\_1[...,1] \* train\_1[...,1]

train\_1 = np.insert(train\_1,3,temp\_1,axis = 1)

temp\_4 = test\_1[...,1] \* test\_1[...,1]

test\_1 = np.insert(test\_1,3,temp\_4,axis = 1)

temp\_2 = train\_1[...,1] \* train\_1[...,2]

train\_1 = np.insert(train\_1,4,temp\_2,axis = 1)

temp\_5 = test\_1[...,1] \* test\_1[...,2]

test\_1 = np.insert(test\_1,4,temp\_5,axis = 1)

temp\_3 = train\_1[...,2] \* train\_1[...,2]

train\_1 = np.insert(train\_1,5,temp\_3,axis = 1)

temp\_6 = test\_1[...,2] \* test\_1[...,2]

test\_1 = np.insert(test\_1,5,temp\_6,axis = 1)

数据的线性映射

train\_min = train\_1.min(axis = 0)

test\_min = test\_1.min(axis = 0)

train\_max = train\_1.max(axis = 0)

test\_max = test\_1.max(axis = 0)

for i in range(1,len(train\_1[0])):

train\_1[...,i] = (train\_1[...,i]-train\_min[i])/(train\_max[i] - train\_min[i])

test\_1[...,i] = (test\_1[...,i]-test\_min[i])/(test\_max[i] - test\_min[i])

构造函数的使用

def function(weights):

def function\_1(x):

y = []

for i in x:

if (np.mat(i) \* np.mat(weights)) >= 0:

y.append(1)

else:

y.append(0)

return y

return function\_1

梯度下降的应用

def grad\_descent(dataMathIn,classLabels):

# 把数组类型转化成矩阵类型

dataMatrix = np.mat(dataMathIn)

# 转化成矩阵类型并进行转置

labelMat = np.mat(classLabels).transpose()

# 获取特征矩阵的特征个数和样本个数

m, n = np.shape(dataMatrix)

# 构建相应的1矩阵

weights = np.ones((n, 1))

weights\_1 = np.zeros((n,1))

alpha = 0.75

lamda = 0.75

maxCycle = 10000

weights\_1[0] = 1

for i in range(1,len(weights\_1)):

weights\_1[i] = 1 \* (1 - alpha \* lamda / m)

# 每一个样本带入公式进行计算推导出相关系数值

for i in range(maxCycle):

h = sigmoid(dataMatrix \* weights)

weights = weights\_1 \* np.array(weights) - alpha / m \* dataMatrix.transpose() \* (h - labelMat)

return weights

实验结果：对数据进行了有效的区分正确率为0.81428