#训练数据前期处理程序  
  
**import** pandas **as** pd # 数据分析  
  
# 读入训练数据，sheetname=0表示第一个表单  
data\_train = pd.read\_excel(**"C:/Users/Tang/Desktop/模式识别作业/homework2018/ML\_data2.xlsx"**,sheetname=0)  
#将国籍为美国的标为1，非美国标为0  
data\_train.native\_country.loc[data\_train.native\_country == **'United-States'**]= 1  
data\_train.native\_country.loc[data\_train.native\_country != 1]= 0  
  
#将性别为男性的标为1，女性标为0  
data\_train.sex.loc[data\_train.sex == **'Male'**]= 1  
data\_train.sex.loc[data\_train.sex != 1]= 0  
  
  
# 因为逻辑回归建模时，需要输入的特征都是数值型特征  
# 我们先对类目型的特征离散/因子化，采用one-hot编码  
# 平展属性  
# 我们使用pandas的get\_dummies来完成这个工作，并拼接在原来的data\_train之上，如下所示  
dummies\_workClass = pd.get\_dummies(data\_train[**'workClass'**], prefix= **'workClass'**)  
dummies\_education = pd.get\_dummies(data\_train[**'education'**], prefix= **'education'**)  
dummies\_marital\_status = pd.get\_dummies(data\_train[**'marital\_status'**], prefix= **'marital\_status'**)  
dummies\_occupation = pd.get\_dummies(data\_train[**'occupation'**], prefix= **'occupation'**)  
dummies\_relationship = pd.get\_dummies(data\_train[**'relationship'**], prefix= **'relationship'**)  
dummies\_race = pd.get\_dummies(data\_train[**'race'**], prefix= **'race'**)  
  
#拼接  
df = pd.concat([data\_train, dummies\_workClass, dummies\_education, dummies\_marital\_status, dummies\_occupation,  
 dummies\_relationship,dummies\_race], axis=1)  
#丢弃  
df.drop([**'workClass'**, **'education'**, **'marital\_status'**, **'occupation'**, **'relationship'**, **'race'**], axis=1, inplace=**True**)  
#归一化，为了加快梯度下降时的迭代速度  
df.age=(df.age-df.age.min())/(df.age.max()-df.age.min())  
df.fnlwgt=(df.fnlwgt-df.fnlwgt.min())/(df.fnlwgt.max()-df.fnlwgt.min())  
df.education\_num=(df.education\_num-df.education\_num.min())/(df.education\_num.max()-df.education\_num.min())  
df.capital\_gain=(df.capital\_gain-df.capital\_gain.min())/(df.capital\_gain.max()-df.capital\_gain.min())  
df.capital\_loss=(df.capital\_loss-df.capital\_loss.min())/(df.capital\_loss.max()-df.capital\_loss.min())  
df.hours\_per\_week=(df.hours\_per\_week-df.hours\_per\_week.min())/(df.hours\_per\_week.max()-df.hours\_per\_week.min())  
  
#保存新数据  
df.to\_csv(**"C:/Users/Tang/Desktop/模式识别作业/homework2018/data\_train.csv"**, index=**False**)

#测试数据执行与训练数据相同的前期处理  
  
**import** pandas **as** pd # 数据分析  
  
# 读入训练数据，sheetname=1表示第二个表单  
test\_train = pd.read\_excel(**"C:/Users/Tang/Desktop/模式识别作业/homework2018/ML\_data2.xlsx"**, sheetname=1)  
  
#将国籍为美国的标为1，非美国标为0  
test\_train.native\_country.loc[test\_train.native\_country == **'United-States'**]= 1  
test\_train.native\_country.loc[test\_train.native\_country != 1]= 0  
  
#将性别为男性的标为1，女性标为0  
test\_train.sex.loc[test\_train.sex == **'Male'**]= 1  
test\_train.sex.loc[test\_train.sex != 1]= 0  
  
  
# 因为逻辑回归建模时，需要输入的特征都是数值型特征  
# 我们先对类目型的特征离散/因子化，采用one-hot编码  
# 平展属性  
# 我们使用pandas的get\_dummies来完成这个工作，并拼接在原来的data\_train之上，如下所示  
dummies\_workClass = pd.get\_dummies(test\_train[**'workClass'**], prefix=**'workClass'**)  
dummies\_education = pd.get\_dummies(test\_train[**'education'**], prefix=**'education'**)  
dummies\_marital\_status = pd.get\_dummies(test\_train[**'marital\_status'**], prefix=**'marital\_status'**)  
dummies\_occupation = pd.get\_dummies(test\_train[**'occupation'**], prefix=**'occupation'**)  
dummies\_relationship = pd.get\_dummies(test\_train[**'relationship'**], prefix=**'relationship'**)  
dummies\_race = pd.get\_dummies(test\_train[**'race'**], prefix=**'race'**)  
  
#拼接  
df = pd.concat([test\_train, dummies\_workClass, dummies\_education, dummies\_marital\_status, dummies\_occupation,  
 dummies\_relationship, dummies\_race], axis=1)  
#丢弃  
df.drop([**'workClass'**, **'education'**, **'marital\_status'**, **'occupation'**, **'relationship'**, **'race'**], axis=1, inplace=**True**)  
#归一化，为了加快梯度下降时的迭代速度  
df.age=(df.age-df.age.min())/(df.age.max()-df.age.min())  
df.fnlwgt=(df.fnlwgt-df.fnlwgt.min())/(df.fnlwgt.max()-df.fnlwgt.min())  
df.education\_num=(df.education\_num-df.education\_num.min())/(df.education\_num.max()-df.education\_num.min())  
df.capital\_gain=(df.capital\_gain-df.capital\_gain.min())/(df.capital\_gain.max()-df.capital\_gain.min())  
df.capital\_loss=(df.capital\_loss-df.capital\_loss.min())/(df.capital\_loss.max()-df.capital\_loss.min())  
df.hours\_per\_week=(df.hours\_per\_week-df.hours\_per\_week.min())/(df.hours\_per\_week.max()-df.hours\_per\_week.min())  
  
#保存新数据  
df.to\_csv(**"C:/Users/Tang/Desktop/模式识别作业/homework2018/data\_test.csv"**, index=**False**)

# 逻辑回归分类器程序  
  
**import** numpy **as** np  
**import** pandas **as** pd  
  
  
# 定义sigmoid函数  
**def** sigmoid(z):  
 gz = (1 / (1 + np.exp(-z)))  
 **return** gz  
  
  
# 梯度下降法求theta  
**def** gradientDescent(x, y, theta, alpha, m, numIterations): # theta要求的参数，alpha学习率，m样本数，numIterations迭代次数  
 xTran = np.transpose(x) # 转置，便于计算  
 h = sigmoid(xTran.dot(theta))  
 **for** i **in** range(numIterations):  
 hypothesis = np.dot(xTran, theta) # x与theta的内积  
 loss = hypothesis - y # 偏差  
 cost = -1 \* (1 / m) \* (np.log(h).T.dot(y) + np.log(1 - h).T.dot(1 - y)) + (lamb / (2 \* m)) \* np.sum(  
 np.square(theta[1:])) # 带正则化项的代价函数  
 gradient = np.dot(x, loss) / m # 梯度  
 theta[0] = theta[0] - alpha \* gradient[0] # 更新theta 0  
 theta[1:] = theta[1:] - alpha \* (gradient[1:] + (lamb / m) \* theta[1:]) # 更新theta1及之后的theta  
 print(**"Iteration %d | cost :%f"** % (i, cost)) # 输出代价值  
 **return** theta # 返回theta  
  
  
**def** predict(x): # 预测函数  
 result = np.dot(theta.T, x) + b  
 result = 1 / (1 + np.exp(-result))  
 **for** j **in** result:  
 **if** j > 0.61: # 当result>0.61时判断为1，否则为0  
 prediction.append(1)  
 **else**:  
 prediction.append(0)  
  
 **return** prediction # 返回预测结果  
  
  
# 训练  
prediction = [] # 将预测值装入prediction数组  
lamb = 1000 # 正则项常数  
b = 0 # 截距  
df\_train = pd.read\_csv(**'data\_train.csv'**) # 读入训练数据  
y = np.array(df\_train)[:, 8] # 将分类标签保存在y中  
X = df\_train.drop([**'salary'**], axis=1) # 将分类标签项丢弃  
X = np.array(X) # 转为数组类型  
numIterations = 1000 # 迭代1000次  
m, n = X.shape # 数据大小  
theta = np.ones(n) # 初始theta全为1  
alpha = 0.1 # 设置更新速率  
X = np.transpose(X) # 转置，便于计算  
theta = gradientDescent(X, y, theta, alpha, m, numIterations) # 训练得到theta  
# 预测  
df\_test = pd.read\_csv(**'data\_test.csv'**) # 读入测试数据  
y\_test = np.array(df\_test)[:, 8] # 将分类标签保存在y\_test中  
X\_test = df\_test.drop([**'salary'**], axis=1) # 将分类标签项丢弃  
X\_test = np.transpose(X\_test) # 转置，便于计算  
predict\_result = predict(X\_test) # 预测  
  
# 保存预测结果  
dataframe = pd.DataFrame({**'salary'**: predict\_result})  
# 将DataFrame存储为csv,index表示是否显示行名，default=True  
dataframe.to\_csv(**"C:/Users/Tang/Desktop/模式识别作业/homework2018/predict\_result0.csv"**, index=**False**)  
  
# 计算准确率  
**from** sklearn.metrics **import** accuracy\_score  
  
print(**'准确率:'**, accuracy\_score(y\_test, predict\_result))  
print(**'预测正确的数目'**, accuracy\_score(y\_test, predict\_result, normalize=**False**))