day6Logistic Regression

笔记本:	100days		
创建时间:	2019/1/11 20:32	**更新时间:**	2019/1/11 20:34
URL:	http://localhost:8888/notebooks/Day6/day6Logistic%20Regression.ipynb		

Logistic Regression¶

#100DaysOfMLCode
Day 4
©Avik Jain

逻辑回归

什么是逻辑回归

逻辑回归被用来处理不同的分类问题,这里的目的是预测当前被观察的对象属于哪个组。它会给你提供一个离散的二进制输出结果。一个简单的例子就是判断一个人是否会在即将到来的选举中进行投票。

如何工作

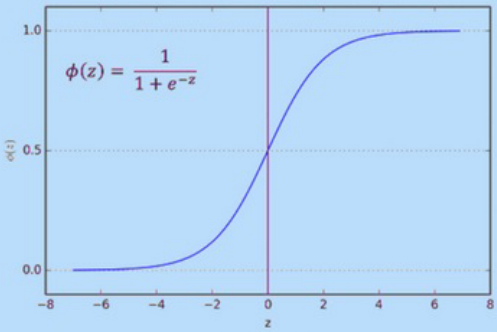
逻辑回归使用基础逻辑函数通过估算概率来测量因变量(我们想要预测的标签)和一个或者多个自变量之间的关系。

Sigmoid 函数

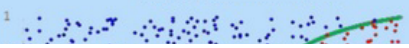
Sigmoid函数是一个S形曲线,可以实现将任意真实值映射为值域范围为0-1的值，但从来不局限于这些限制。

做出预测

这些概率值必须转换为二进制数，以便实际中进行预测。这是逻辑函数的任务，也被称为sigmoid 函数。然后使用阈值分类器将(0,1)范围的值转化成0和1的值来表示结果。



逻辑回归示例



Step 1 | Data Pre-Processing(数据预处理)¶

In [5]:

5

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read_csv('Social_Network_Ads.csv')

dataset
```

Out[5]:

	USER ID	GENDER	AGE	ESTIMATEDSALARY	PURCHASED
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0

4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0
16	15733883	Male	47	25000	1
17	15617482	Male	45	26000	1
18	15704583	Male	46	28000	1
19	15621083	Female	48	29000	1
20	15649487	Male	45	22000	1
21	15736760	Female	47	49000	1
22	15714658	Male	48	41000	1
23	15599081	Female	45	22000	1
24	15705113	Male	46	23000	1
25	15631159	Male	47	20000	1
26	15792818	Male	49	28000	1
27	15633531	Female	47	30000	1
28	15744529	Male	29	43000	0
29	15669656	Male	31	18000	0
...	...	...	...	...	...
370	15611430	Female	60	46000	1
371	15774744	Male	60	83000	1
372	15629885	Female	39	73000	0
373	15708791	Male	59	130000	1

374	15793890	Female	37	80000	0
375	15646091	Female	46	32000	1
376	15596984	Female	46	74000	0
377	15800215	Female	42	53000	0
378	15577806	Male	41	87000	1
379	15749381	Female	58	23000	1
380	15683758	Male	42	64000	0
381	15670615	Male	48	33000	1
382	15715622	Female	44	139000	1
383	15707634	Male	49	28000	1
384	15806901	Female	57	33000	1
385	15775335	Male	56	60000	1
386	15724150	Female	49	39000	1
387	15627220	Male	39	71000	0
388	15672330	Male	47	34000	1
389	15668521	Female	48	35000	1
390	15807837	Male	48	33000	1
391	15592570	Male	47	23000	1
392	15748589	Female	45	45000	1
393	15635893	Male	60	42000	1
394	15757632	Female	39	59000	0
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

. . .

In [6]:

```
xxxxxxxxxx
```

```
X = dataset.iloc[:,[2,3]].values
```

```
Y = dataset.iloc[:,4].values
```

```
X[:10], Y[:10]
```

Out[6]:

```
(array([[ 19, 19000],
[ 35, 20000],
[ 26, 43000],
[ 27, 57000],
[ 19, 76000],
[ 27, 58000],
[ 27, 84000],
[ 32, 150000],
[ 25, 33000],
[ 35, 65000]], dtype=int64),
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64))
```

. . .

In [9]:

2

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25,random_state=0)
```

. . .

Feature Scaling(特征缩放)¶

In [12]:

```
xxxxxxxxxx
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

```
X_train[:10],X_test[:10]
```

Out[12]:

```
(array([[ 0.58164944, -0.88670699],
[-0.60673761, 1.46173768],
[-0.01254409, -0.5677824 ],
[-0.60673761, 1.89663484],
[ 1.37390747, -1.40858358],
[ 1.47293972, 0.99784738],
[ 0.08648817, -0.79972756],
[-0.01254409, -0.24885782],
[-0.21060859, -0.5677824 ],
[-0.21060859, -0.19087153]]), array([[-0.80480212, 0.50496393],
[-0.01254409, -0.5677824 ],
[-0.30964085, 0.1570462 ],
[-0.80480212, 0.27301877],
[-0.30964085, -0.5677824 ],
[-1.10189888, -1.43757673],
[-0.70576986, -1.58254245],
[-0.21060859, 2.15757314],
[-1.99318916, -0.04590581],
[ 0.8787462 , -0.77073441]]))
```

. . .

Step 2 | Logistic Regression Model(逻辑回归模型)¶

Fitting Logistic Regression to the Training set(将逻辑回归应用于训练集)¶

In [16]:

```
from sklearn.linear_model import LogisticRegression
```

```
classifier = LogisticRegression()
```

```
classifier.fit(X_train,Y_train)
```

```
classifier
```

```
C:\Users\张帅\AppData\Roaming\Python\Python36\site-
packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
FutureWarning)
```

Out[16]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn',
tol=0.0001, verbose=0, warm_start=False)
```

. . .

Step 3 | Predection(预测)¶

Predicting the Test set results(预测测试集结果)¶

In [18]:

1

```
Y_pred = classifier.predict(X_test)
```

```
Y_pred
```

Out[18]:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
```

```
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

. . .

Step 4 | Evaluating The Predection(评估预测)¶

Making the Confusion Matrix(生成混淆矩阵)¶

In [19]:

```
xxxxxxxxxx
```

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(Y_test,Y_pred)
```

```
cm
```

Out[19]:

```
array([[65, 3],
[ 8, 24]], dtype=int64)
```

. . .

Visualization(可视化)¶

In [51]:

19

```
from matplotlib.colors import ListedColormap
```

```
X_set, Y_set = X_train, Y_train
```

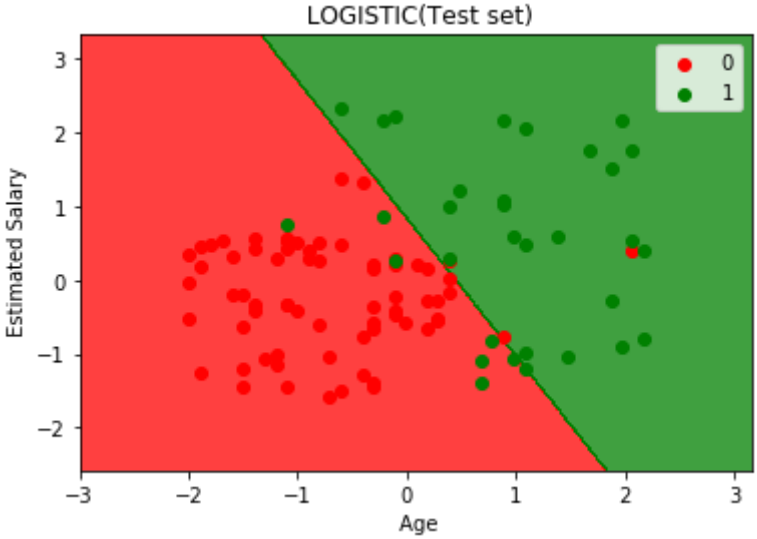
```
X1, X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1,stop=X_set[:,0].max()+1,step=0.01),
```

```
np.arange(start=X_set[:,1].min()-1,stop=X_set[:,1].max()+1,step=0.01))

plt.contourf(X1,X2,classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75,cmap = ListedColormap(('red','green')))

plt.xlim(X1.min(),X1.max())

plt.ylim(X2.min(),X2.max())

for i,j in enumerate(np. unique(Y_set)):

plt.scatter(X_set[Y_set==j,0],X_set[Y_set==j,1],

c = ListedColormap(('red', 'green'))(i), label=j)

plt. title(' LOGISTIC(Training set)')

plt. xlabel(' Age')

plt. ylabel(' Estimated Salary')

plt. legend()

plt. show()
```

. . .

In [54]:

```
X_set,Y_set=X_test,Y_test

X1,X2=np. meshgrid(np. arange(start=X_set[:,0].min()-1, stop=X_set[:, 0].max()+1, step=0.01),

np. arange(start=X_set[:,1].min()-1, stop=X_set[:,1].max()+1, step=0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(),X1.max())

plt.ylim(X2.min(),X2.max())

for i,j in enumerate(np. unique(Y_set)):

plt.scatter(X_set[Y_set==j,0],X_set[Y_set==j,1],

c = ListedColormap(('red', 'green'))(i), label=j)

plt. title(' LOGISTIC(Test set)')

plt. xlabel(' Age')

plt. ylabel(' Estimated Salary')

plt. legend()

plt. show()
```



. . .

In [ ]:

1

. . .