机器学习算法_基于逻辑回归的分类预测

> 学习及学习笔记
>
> 另外github不太支持latex语法，于是使用图片推导

> 课前学习：高等数学的微积分部分

逻辑回归的介绍

逻辑回归（Logistic regression，简称LR）虽然其中带有"回归"两个字，但逻辑回归其实是一个**分类模型**，并且广泛应用于各个领域之中。虽然现在深度学习相对于这些传统方法更为火热，但实则这些传统方法由于其独特的优势依然广泛应用于各个领域中。

> (备注：机器学习包括符号方面的也算，在好几年前，符号机器学习占主要地位，所以来说相对还是比较重要，只是进些年来讲，深度学习能解决的问题基于大数据，更容易发挥其作用。这里要提到算力，计算能力。)

而对于逻辑回归而言，最为突出的两点就是其**模型简单和模型的可解释性强。**

逻辑回归模型的优劣势:

优点：实现简单，易于理解和实现；计算代价不高，速度很快，存储资源低；

> 计算代价不高(这里的意思是计算机所要求的算力不用太强，这里类比一下后面的图像识别的机器学习，卷积轴所需要的就是极大的算力，计算大量数据的能力，神经网络也是)

缺点：容易欠拟合

> （这里我们可以使用正则的方式降低数据过拟合，数学建模中也常常使用），分类精度可能不高

Logistic 曲线，就是我们所说的种族生长曲线，就是s型曲线。在数学建模的领域中也常常被使用到，可以作为一种预测模型。

正则化的内容

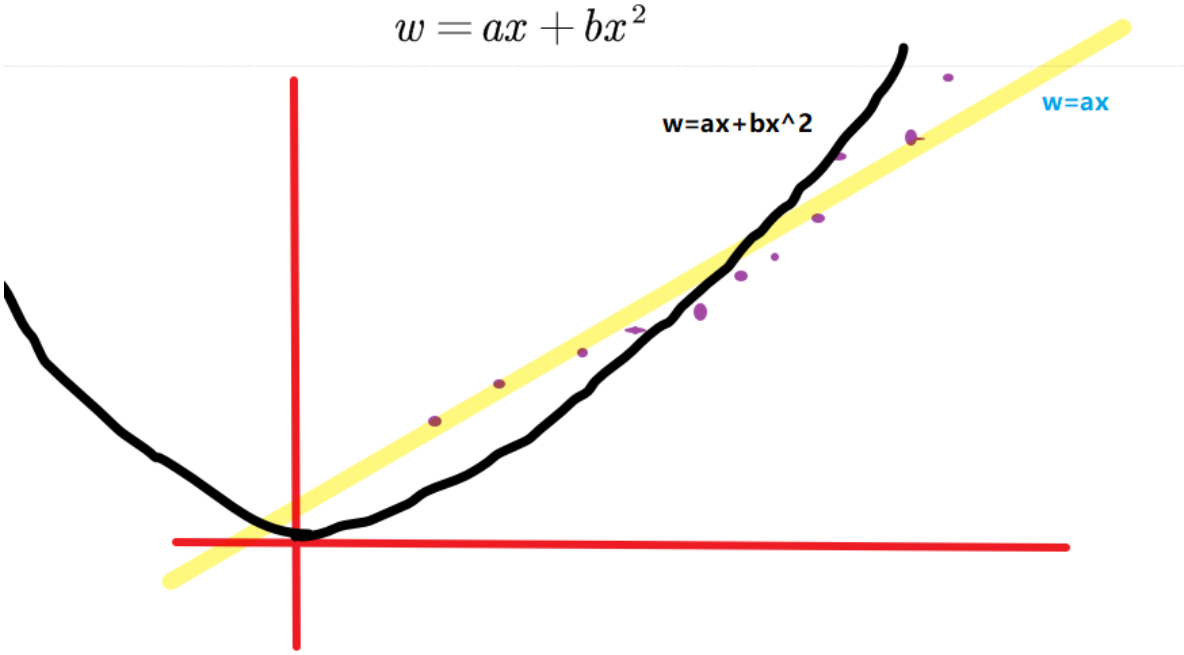
数据拟合的一类，这里我多提供一个正则化的内容（减少过拟合）：

法一：岭回归

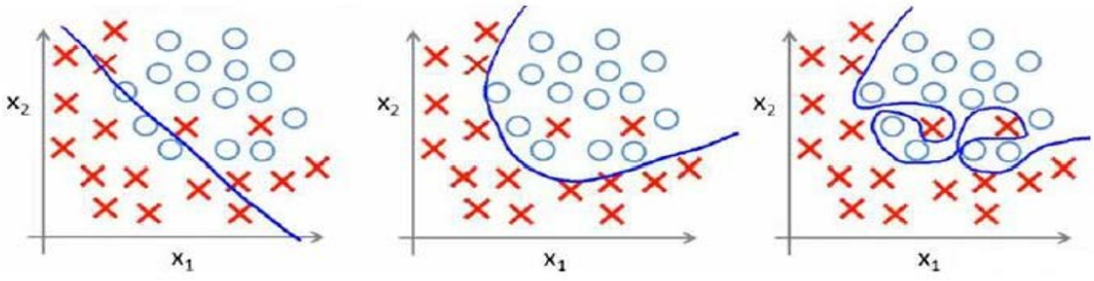
```
基于数组构建并拟合模型的调用格式为:
import statsmodels.api as sm
sm.OLS(y,X).fit()
```

稍微介绍下正则化：相当于减少了高次方的数据，导致拟合不会过度。这里只是简单的举例，了解了解是可以的，未来我们会学到。不过稍微不太一样。

$$w = ax + bx^2$$



这里我再次使用吴恩达老师上课时的做法：



x的次数越高，拟合的越好，但相应的预测的能力就可能变差。正则化： 保留所有的特征，但是减少参数的大小。

逻辑回归的应用

逻辑回归模型广泛用于各个领域，包括机器学习，大多数医学领域和社会科学。例如，最初由Boyd 等人开发的创伤和损伤严重度评分（TRISS）被广泛用于**预测受伤患者的死亡率**，使用逻辑回归 基于观察到的**患者特征**（年龄，性别，体重指数,各种血液检查的结果等）**分析预测**发生特定疾病（例如糖尿病，冠心病）的风险。逻辑回归模型也用于预测在给定的过程中，系统或产品的故障的可能性。还用于市场营销应用程序，例如**预测客户购买产品或中止订购的倾向**等。在经济学中它可以用来预测一个人选择进入劳动力市场的可能性，而商业应用则可以用来预测房主拖欠抵押贷款的可能性。条件随机字段是逻辑回归到顺序数据的扩展，用于自然语言处理。

逻辑回归模型现在同样是很多分类算法的基础组件,比如 分类任务中基于GBDT算法+LR逻辑回归实现的信用卡交易反欺诈，CTR(点击通过率)预估等，其好处在于输出值自然地落在0到1之间，并且有概率意义。模型清晰，有对应的概率学理论基础。它拟合出来的参数就代表了每一个特征(feature)对结果的影响。也是一个理解数据的好工具。但同时由于其本质上是一个线性的分类器，所以不能应对较为复杂的数据情况。很多时候我们也会拿逻辑回归模型去做一些任务尝试的基线（基础水平）。

> 总结：一般根据学习者的数据属性，每一个变量相当于一种属性，根据属性的不同，控制函数分类的区域不同，然后分类预测不同的人差不多的数据属性的值。

> > 本次任务：了解逻辑回归理论，使用sklearn函数，运用到鸢尾花数据中

2.1 代码流程

Part1 Demo实践

Step1:库函数导入
Step2:模型训练
Step3:模型参数查看
Step4:数据和模型可视化
Step5:模型预测

Part2 基于鸢尾花（iris）数据集的逻辑回归分类实践

Step1:库函数导入
Step2:数据读取/载入
Step3:数据信息简单查看
Step4:可视化描述
Step5:利用 逻辑回归模型 在二分类上 进行训练和预测
Step6:利用 逻辑回归模型 在三分类(多分类)上 进行训练和预测

代码实现

库文件的导入

```
##   基础函数库
import numpy as np

## 导入画图库
import matplotlib.pyplot as plt
import seaborn as sns

## 导入逻辑回归模型函数  Logistic 训练模型
from sklearn.linear_model import
LogisticRegression  #这个是导入sklearn里面的
LogisticRegression库
```

模型训练

```
##Demo演示LogisticRegression分类

## 构造数据集
x_fearures = np.array([[-1,-2],
                       [-2,-1],
                       [-3,-2],
                       [ 1, 3],
                       [ 2, 1],
                       [ 3, 2]])

y_label = np.array([0,0,0,1,1,1])

## 调用逻辑回归模型
lr_clf = LogisticRegression()

## 用逻辑回归模型拟合构造的数据集
lr_clf = lr_clf.fit(x_fearures, y_label)    #一般.fit()都是训练机器模型
#其拟合方程为 y=w0+w1*x1+w2*x2
```

这里的x矩阵为

$$y = w_0 + w_1 \times x_1 + w_2 \times x_2;\ x = \begin{pmatrix} -1 & -2 \\ -2 & -1 \\ -3 & -2 \\ 1 & 3 \\ 2 & 1 \\ 3 & 2 \end{pmatrix}; y = (0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1);$$

注：这里的x要加一列1，因为为常数项

所以要设w矩阵

$$w=(w_1 \quad w_2 \quad w_3);\quad y^T=x\times w$$

$$y^T\begin{pmatrix}0\\0\\0\\1\\1\\1\end{pmatrix}=x\begin{pmatrix}1&-1&-2\\1&-2&-1\\1&-3&-2\\1&1&3\\1&2&1\\1&3&2\end{pmatrix}\times w(w_1 \quad w_2 \quad w_3)$$

这里的x矩阵为

$$y=w_0+w_1\times x_1+w_2\times x_2;\ x=\begin{pmatrix}-1&-2\\-2&-1\\-3&-2\\1&3\\2&1\\3&2\end{pmatrix};y=(0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1);$$

注：这里的x要加一列1，因为为常数项

所以要设w矩阵

$$w=(w_1 \quad w_2 \quad w_3);\quad y^T=x\times w$$

$$y^T\begin{pmatrix}0\\0\\0\\1\\1\\1\end{pmatrix}=x\begin{pmatrix}1&-1&-2\\1&-2&-1\\1&-3&-2\\1&1&3\\1&2&1\\1&3&2\end{pmatrix}\times w(w_1 \quad w_2 \quad w_3)$$

类似训练方法，求出恰当的w值，这里只用保证接近就好，不是一定能算出准确的值。

模型参数查看

```
##查看其对应模型的w
print('the weight of Logistic Regression:',lr_clf.coef_)
#>>> the weight of Logistic Regression:
[[0.73462087 0.6947908]]

##查看其对应模型的w0
print('the intercept(w0) of Logistic Regression:',lr_clf.intercept_)
#>>> the intercept(w0) of Logistic Regression:
[-0.03643213]
```

数据和模型可视化

```
## 可视化构造的数据样本点

plt.figure() # 初始化窗口

#散点图设置  在附录有相关设置
plt.scatter(x_fearures[:,0],x_fearures[:,1], c=y_label, s=50, cmap='viridis')
#设置标题
plt.title('Dataset')
#展示
plt.show()

##画出决策边界
plt.figure()
plt.scatter(x_fearures[:,0],x_fearures[:,1], c=y_label, s=50, cmap='viridis')
plt.title('Dataset')

nx, ny = 200, 100              #控制步长
x_min, x_max = plt.xlim()    #设置坐标轴的范围
y_min, y_max = plt.ylim()
```

```
                                    #下：设置网格，这里控制网格的格数
x_grid, y_grid = np.meshgrid(np.linspace(x_min, x_max, nx),np.linspace(y_min, y_max, ny))

#这为训练得到一条较为好的坐标点，分割数据的线
z_proba = lr_clf.predict_proba(np.c_[x_grid.ravel(), y_grid.ravel()])

#这为获取矩阵的维数来改变矩阵形状（可能是概率矩阵）
z_proba = z_proba[:, 1].reshape(x_grid.shape)

# 等高线 contour 线的设置与绘图
plt.contour(x_grid, y_grid, z_proba, [0.5], linewidths=2., colors='blue')

plt.show()

### 可视化预测新样本
plt.figure()

## new point 1
x_fearures_new1 = np.array([[0, -1]])
plt.scatter(x_fearures_new1[:,0],x_fearures_new1[:,1], s=50, cmap='viridis')

#设置注释：s\text（这里修改，3.3以上为text）内容，xy坐标，文字坐标xytext，color颜色，
##plt.annotate(s='New point 1',xy=(0,-1),xytext=(-2,0),color='blue',arrowprops=dict(arrowstyle='-|>',connectionstyle='arc3',color='red'))
plt.annotate(text='New point 1',xy=(0,-1),xytext=(-2,0),color='blue',arrowprops=dict(arrowstyle='-|>',connectionstyle='arc3',color='red'))

```

```
## new point 2  同上
x_fearures_new2 = np.array([[1, 2]])
plt.scatter(x_fearures_new2[:,0],x_fearures_new2[:,1], s=50, cmap='viridis')

##plt.annotate(s='New point 2',xy=(1,2),xytext=(-1.5,2.5),color='red',arrowprops=dict(arrowstyle='-|>',connectionstyle='arc3',color='red'))
plt.annotate(text='New point 2',xy=(1,2),xytext=(-1.5,2.5),color='red',arrowprops=dict(arrowstyle='-|>',connectionstyle='arc3',color='red'))

## 训练样本
plt.scatter(x_fearures[:,0],x_fearures[:,1], c=y_label, s=50, cmap='viridis')
plt.title('Dataset')

# 可视化决策边界
plt.contour(x_grid, y_grid, z_proba, [0.5], linewidths=2., colors='blue')

plt.show()

```

模型预测

```
##在训练集和测试集上分布利用训练好的模型进行预测
y_label_new1_predict=lr_clf.predict(x_fearures_new1)#(0,-1)->属0区
y_label_new2_predict=lr_clf.predict(x_fearures_new2)#(1,2)->属1区
print('The New point 1 predict class:\n',y_label_new1_predict)
print('The New point 2 predict class:\n',y_label_new2_predict)
```

```
##TheNewpoint1predictclass:
##[0]
##ThseNewpoint2predictclass:
##[1]

##由于逻辑回归模型是概率预测模型（前文介绍的p = p(y=1|x,\theta)）,所有我们可以利用predict_proba函数预测其概率
y_label_new1_predict_proba=lr_clf.predict_proba(x_fearures_new1)
y_label_new2_predict_proba=lr_clf.predict_proba(x_fearures_new2)
print('The New point 1 predict Probability of each class:\n',y_label_new1_predict_proba)
print('The New point 2 predict Probability of each class:\n',y_label_new2_predict_proba)
##TheNewpoint1predictProbabilityofeachclass:
##[[0.69567724  0.30432276]]
##TheNewpoint2predictProbabilityofeachclass:
##[[0.11983936  0.88016064]]
      为0的概率          为1的概率
###可以发现训练好的回归模型将X_new1预测为了类别0（判别面左下侧），X_new2预测为了类别1（判别面右上侧）。其训练得到的逻辑回归模型的概率为0.5的判别面为上图中蓝色的线。
```

附录

阿里实验室的使用：（预防忘记）

```
# 查看数据文件目录  list datalab files
!ls datalab/

# 查看个人永久空间文件  list files in your permanent storage
!ls /home/tianchi/myspace/

# 查看当前kernel下已安装的包  list packages
!pip list --format=columns

# 安装扩展包时请使用阿里云镜像源  install packages
!pip install pyodps -i "https://mirrors.aliyun.com/pypi/simple/"

# 绘图案例 an example of matplotlib
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import jn
from IPython.display import display, clear_output
import time
x = np.linspace(0,5)
f, ax = plt.subplots()
ax.set_title("Bessel functions")

for n in range(1,10):
```

```
    time.sleep(1)
    ax.plot(x, jn(x,n))
    clear_output(wait=True)
    display(f)

# close the figure at the end, so we don't get a duplicate
# of the last plot
plt.close()
```

库的安装

> 1.打开cmd输入$ `pip install seaborn`
>
> 2.或者在`https://www.lfd.uci.edu/~gohlke/pythonlibs`
>
> 下轮子 cmd中输入`pip install c:/name/.../pythonlibs_filename.whl`

散点图设置

```
scatter的参数
plt.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, *, data=None, **kwargs)
```

x，y:　　表示的是大小为(n,)的数组，数据点

s:　　点的大小

c:　　颜色

marker：标记的样式

alpha：　透明度，0-1之间

其他的不常用（主要是我也不是很懂，很愿意一起分享）

无注释代码

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
x_fearures = np.array([[-1,-2],[-2,-1],[-3,-2],[1, 3],[ 2, 1],[ 3, 2]])
y_label = np.array([0,0,0,1,1,1])
lr_clf = LogisticRegression()
lr_clf = lr_clf.fit(x_fearures, y_label)
print('the weight of Logistic Regression:',lr_clf.coef_)
print('the intercept(w0) of Logistic Regression:',lr_clf.intercept_)
plt.figure()
plt.scatter(x_fearures[:,0],x_fearures[:,1], c=y_label, s=50, cmap='viridis')
plt.title('Dataset')
plt.show()
plt.figure()
plt.scatter(x_fearures[:,0],x_fearures[:,1], c=y_label, s=50, cmap='viridis')
plt.title('Dataset')
nx, ny = 200, 100
x_min, x_max = plt.xlim()
y_min, y_max = plt.ylim()
x_grid, y_grid = np.meshgrid(np.linspace(x_min, x_max, nx),np.linspace(y_min, y_max, ny))
```

```
z_proba = lr_clf.predict_proba(np.c_[x_grid.ravel(), y_grid.ravel()])
z_proba = z_proba[:, 1].reshape(x_grid.shape)
plt.contour(x_grid, y_grid, z_proba, [0.5], linewidths=2., colors='blue')
plt.show()
plt.figure()
x_fearures_new1 = np.array([[0, -1]])
plt.scatter(x_fearures_new1[:,0],x_fearures_new1[:,1], s=50, cmap='viridis')
plt.annotate(text='New point 1',xy=(0,-1),xytext=(-2,0),color='blue',arrowprops=dict(arrowstyle='-|>',connectionstyle='arc3',color='red'))
x_fearures_new2 = np.array([[1, 2]])
plt.scatter(x_fearures_new2[:,0],x_fearures_new2[:,1], s=50, cmap='viridis')
plt.annotate(text='New point 2',xy=(1,2),xytext=(-1.5,2.5),color='red',arrowprops=dict(arrowstyle='-|>',connectionstyle='arc3',color='red'))
plt.scatter(x_fearures[:,0],x_fearures[:,1], c=y_label, s=50, cmap='viridis')
plt.title('Dataset')
plt.contour(x_grid, y_grid, z_proba, [0.5], linewidths=2., colors='blue')
plt.show()
y_label_new1_predict=lr_clf.predict(x_fearures_new1)
y_label_new2_predict=lr_clf.predict(x_fearures_new2)
print('The New point 1 predict class:\n',y_label_new1_predict)
print('The New point 2 predict class:\n',y_label_new2_predict)
y_label_new1_predict_proba=lr_clf.predict_proba(x_fearures_new1)
```

```
41 y_label_new2_predict_proba=lr_clf.predict_proba(
   x_fearures_new2)
42 print('The New point 1 predict Probability of
   each class:\n',y_label_new1_predict_proba)
43 print('The New point 2 predict Probability of
   each class:\n',y_label_new2_predict_proba)
44
```