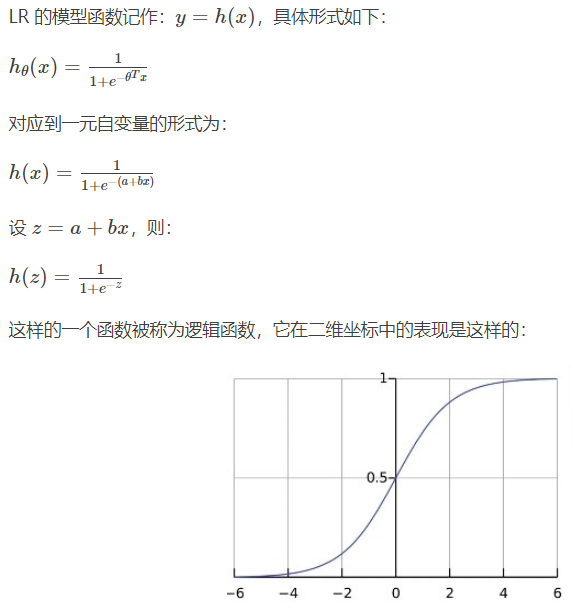
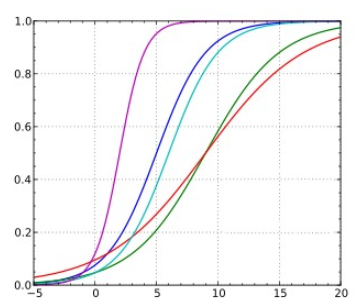
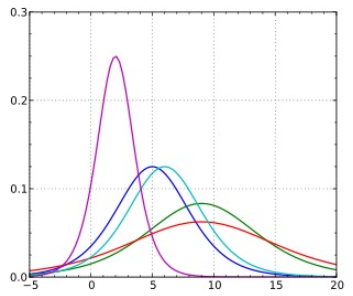
# 第14课：逻辑回归——非线性逻辑函数的由来

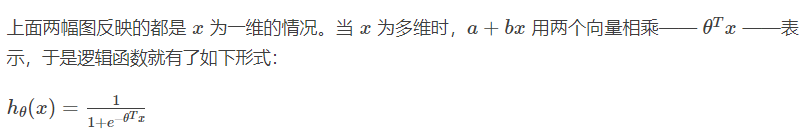
逻辑回归：Logistic Regression （LR）  


逻辑函数表示存量随时间增长渐增的关系。



逻辑函数的微分函数，表示增长率与时间的关系





## 借助数学工具解决问题的方法

1. 将目标问题定义为一个函数
2. 选取最简单的假设作为某具体形式
3. 用事实数据验证该形式，确认有效后沿用，形成数学模型
4. 一旦当前采用的数学模型出现问题，则对其进行修正，同样借助事实数据求取修正项的形式，形成新的数学模型

## 线性VS非线性

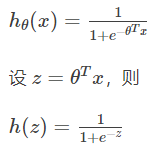
线性关系表达的是一种相关性。在二维坐标系中，体现为直线；非线性体现为曲线。

线性回归简单，容易理解和计算，被广泛应用，但是没有非线性准确，可非线性的计算复杂，成本高。

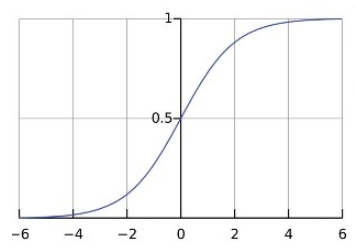
# 第15课：逻辑回归——用来做分类的回归模型

线性回归的预测结果是一个连续值域上的任意值，而朴素贝叶斯分类模型的预测结果则是一个离散值。

但是LR用来做分类，模型为：



图像：



上图中，z是自变量（横轴），最终计算出的因变量 y（纵轴），则是一个 [0,1] 区间之内的实数值。

一般而言，当 y>0.5时，z 被归类为真（True）或阳性（Positive），否则当 y<=0.5 时，z被归类为假（False）或阴性（Negative）。

所以，在模型输出预测结果时，不必输出 y 的具体取值，而是根据上述判别标准，输出1（真）或0（假）。

因此，LR 典型的应用是二分类问题上，也就是说，把所有的数据只分为两个类。

注意：

（1）LR也能处理多分类的问题。

（2）此模型在y=0.5附近非常敏感，自变量取值稍有不同，因变量的差异就非常大，

所以即使出现极端情况，也不会出现大量因细微特征差异而被归错类的情况。

## 逻辑回归的目标函数

从公式本身来看

h(x) 实际上是 xx 为阳性的分布概率，所以，才会在 h(x)>0.5 时将 xx归于阳性。也就是说 h(x)=P(y=1)。反之，样例是阴性的概率 P(y=0)=1−h(x)。

因此：



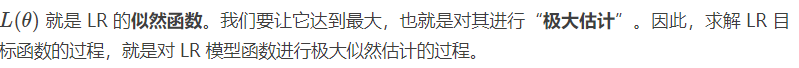
根据二项分布：



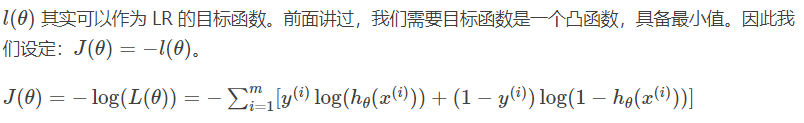
当有m个数据的联合概率为：



求取θ，让L的值最大



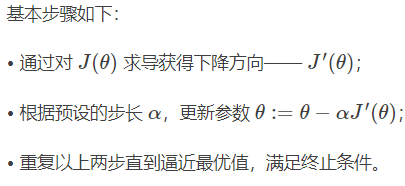
为了求出最大值，我们可以先取对数，再取负数，来求最小值，因为我们需要凸函数



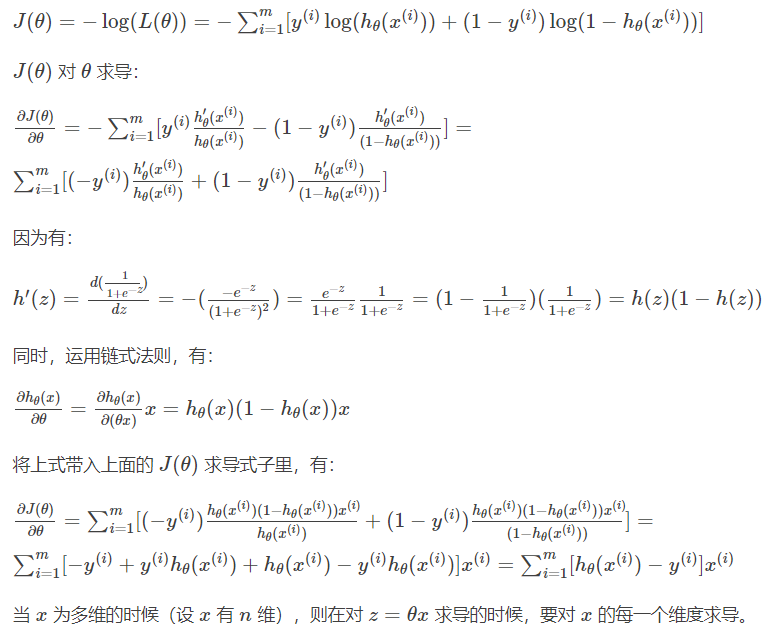
所以LR的目标函数，转变成J

## 优化算法

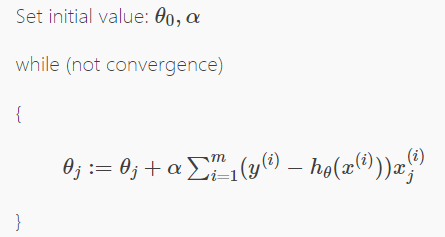
**梯度下降算法：**



**计算：**



伪代码：



## 实例



线性回归代码：

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.linear\_model **import** LinearRegression

**import** pandas **as** pd

*# Importing dataset*

data = pd.read\_csv('quiz.csv', delimiter=',')

used\_features = ["Last Score", "Hours Spent"]

X = data[used\_features].values

scores = data["Score"].values

X\_train = X[:11]

X\_test = X[11:]

*# Linear Regression - Regression*

y\_train = scores[:11]

y\_test = scores[11:]

regr = LinearRegression()

regr.fit(X\_train, y\_train)

y\_predict = regr.predict(X\_test)

**print**(y\_predict)

4号这种特殊情况，在数据量较小的情况下，会影响结果，因为线性回归，所有训练样本对目标的贡献是平均的。

把具体分数转变成是否合格，合格标志为1，不合格为0，然后再进行逻辑回归：

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.linear\_model **import** LinearRegression

**import** pandas **as** pd

*# Importing dataset*

data = pd.read\_csv('quiz.csv', delimiter=',')

used\_features = [ "Last Score", "Hours Spent"]

X = data[used\_features].values

scores = data["Score"].values

X\_train = X[:11]

X\_test = X[11:]

*# Logistic Regression – Binary Classification*

passed = []

**for** i **in** range(len(scores)):

**if**(scores[i] >= 60):

passed.append(1)

**else**:

passed.append(0)

y\_train = passed[:11]

y\_test = passed[11:]

classifier = LogisticRegression(C=1e5)

classifier.fit(X\_train, y\_train)

y\_predict = classifier.predict(X\_test)

print(y\_predict)

## LR处理多分类的问题

假设你一共有 n 个标签（类别），也就是说可能的分类一共有 n 个。那么就构造 n 个 LR 分类模型，第一个模型用来区分 label\_1和 non-label \_1（即所有不属于 label\_1 的都归属到一类），第二个模型用来区分 label\_2 和 non-label \_2……, 第 n 个模型用来区分 label\_n 和 non-label \_n。

如果有多个模型都得出了Positive，选最高的即可，因为LR是一个回归模型。

sklearn 已经为我们处理了多分类问题，我们用 sklearn 来做多分类的时候，只是需要把 y 准备好，其他的，都和做二分类一样就可以了。

比如还是上面的例子，现在我们需要区分：学生的本次成绩是优秀（>=85），及格，还是不及格。我们就在处理 y 的时候给它设置三个值：0 （不及格），1（及格）和2（优秀），然后再做 LR 分类就可以了。代码如下：

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.linear\_model **import** LinearRegression

**import** pandas **as** pd

*# Importing dataset*

data = pd.read\_csv('quiz.csv', delimiter=',')

used\_features = [ "Last Score", "Hours Spent"]

X = data[used\_features].values

scores = data["Score"].values

X\_train = X[:11]

X\_test = X[11:]

*# Logistic Regression - Multiple Classification*

level = []

**for** i **in** range(len(scores)):

**if**(scores[i] >= 85):

level.append(2)

**elif**(scores[i] >= 60):

level.append(1)

**else**:

level.append(0)

y\_train = level[:11]

y\_test = level[11:]

classifier = LogisticRegression(C=1e5)

classifier.fit(X\_train, y\_train)

y\_predict = classifier.predict(X\_test)

print(y\_predict)

## 附录

quiz.csv 文件：

Id,Last Score,Hours Spent,Score

1,90,117,89

2,85,109,78

3,75,113,82

4,98,20,95

5,62,116,61

6,36,34,32

7,87,120,88

8,89,132,92

9,60,83,52

10,72,92,65

11,73,112,71

12,56,143,62

13,57,97,52

14,91,119,93