|  |
| --- |
| **Github账号：https://github.com/ZmA0/Coursera\_machine\_learing\_exercise** |
| **实验题目:逻辑回归中梯度下降算法及代价函数的应用** |
| **实验摘要：**   1. 实验目的：   利用matlab或者octave，掌握对机器学习中的逻辑回归的梯度下降算法和代价函数的代码的编写，以及逻辑回归特征的归一化等知识点的编写。   1. 实验要求：   利用matlab或者octave的数学编程语言进行书写，按照代码文件中的要求编写并提交文件，使得程序运行通过并且得到足够的分数即为通过。   1. 代码文件要求：   需要编写的文件为  sigmoid.m,  predict.m,  costFunction.m  costFunctionReg.m |
| **题目描述**   1. sigmoid.m:   这道题的instructions部分说明我们需要编写逻辑回归中的S型函数的主体部分，而且诶为了按照整个课程的要求，最好利用向量化的编程思想来实现。   1. predict.m   这道题需要计算通过两个参数theta和X以及函数predict来进行y的判断，即逻辑函数中通过S型函数的结果以0.5作为分界线进行分类。  这题的工作就是在上一题的基础上，进行一个简单的if判断，如果s型函数的返回值为大于0.5，则返回y=1，否则令y=0   1. costFunction.m   首先这道题开始定义了一个返回值有两个的函数，参数有X,y，theta。这道题需要计算两个值，一个是逻辑回归中的代价函数即J，还有一个是J的对每个theta的偏导数grad。J和grad我们都可以通过现有的公式编写出来，需要注意的就是必须要利用向量化的思想，才能方便的对每个theta都可以计算到，而不需要用循环来实现。  4.costFunctionReg.m  这道题大体与上面相似，不同的是这里需要进行特征的归一化，也就是需要在原来代价函数的后面加上关于theta的惩罚项，但是惩罚项是关于除了theta(0)之外的所有theta，因此我们在进行完特征归一化的编写后，需要对theta（0）即matlab中的theta（1）进行减去的操作才能满足题意。 |
| **实验过程**   1. sigmoid.m   根据上面的题目描述可以知道，这道题需要编写S型函数，通过已知的S型公式可知，这里面涉及的点一个就是在matlab中自然底数e的表示，以及函数的这个向量参数中对每个元素单独处理的要求。对于第一个我们可以知道exp（-z）即可满足题意，而对于第二点在进行具体运算时改为点运算即可满足题意。代码如下：  t =exp(-z);  g = 1./(1+t);  2.predict.m  这道题可以直接利用上一题的成果，再加上一个判断返回值的if结构，判断条件是函数返回值是否大于0.5，这样一来就可以满足题意。另外一个就是注意X和theta作为矩阵和向量是的维数以及乘法运算。  代码入下：  res=sigmoid( X \*theta);  if (res>0.5)  p=1;  else  p=0;  end  后来知道可以有一种更为简洁的写法  res=sigmoid(X\*theta);  p=(res>0.5)；  利用布尔值直接进行了赋值，更加简洁   1. costFunction.m   这道题根据之前学习的内容可以知道在逻辑回归中代价函数J可以简化为，这个公式里面我们需要注意几个点，一个就是累加的表示，还有一个就是同样的需要进行矩阵元素中的单独运算。其中累加我们同样利用matlab中的sum函数，而这道题中y，theta仍然是一个列向量，h函数的结果也是一系列的值组成的列向量，因此很明显这里仍然需要先利用点运算算出一系列结果值，再将整体进行sum运算，得到J的值。计算J的偏导数时，与上一实验的结构思路相同，即利用X转置所带来的便利，省去了sum和  的变小而变大；当实际的  y=0  且  h  θ  也为  0  时代价为  0  ，当  y=0  但  h  θ  不为  0  时误差随着  h  θ  的变大而变大。    将构建的  Cost(h  θ  (x),y)  简化如下： |
| 和X的导数附加项。因此综上可以知道代码如下：  J= (-1)/m \* sum(( y .\* log(sigmoid(X \* theta)) +(1-y) .\* log(1-sigmoid(X\*theta))))  grad=(1/m) \* X' \* (sigmoid(X\*theta)-y);  4.costFunctionReg.m  这道题的分析过程与上题基本相同，不过在代码编写的时候需要多加一个sum函数构成的累加项，同时还应该注意J的归一化时要在累加项中减去theta(1)这一项的平方，而在grad这个偏导数中同样的需要在进行完有关lambda和theta的累加后再减去theta(1),这二者的原因都是这里的theta（1）是实际参数中的theta0，而theta0是我们为了方便运算而进行的一个偏差额外项，不需要进行额外的惩罚，因此需要减掉。综上所述，代码如下：  J1= (-1)/m \* sum(( y .\* log(sigmoid(X \* theta)) +(1-y) .\* log(1-sigmoid(X\*theta))));  J2=lambda/(2\*m) \* (sum(theta.^2)-theta(1)^2);  J = J1+J2;  grad1=(1/m) \* X' \* (sigmoid(X\*theta)-y);  grad2=lambda/m \* theta;  grad=grad1+grad2;  grad(1)=grad(1)-lambda/m \* theta(1); |
| **实验总结**  第二次编程基本上来讲和第一次的思想体系基本一样，主要是函数内容不一样，在逻辑回归中因为输出是离散值，因此需要更加复杂和严谨的规范才能保证函数的正确输出。在编程过程中犯错最多的是在进行特征归一化书写的时候，忘记了对附加项不进行惩罚而导致多次没有通过，后来仔细阅读了一遍网上课程的笔记，才发现自己的错误。同样的这次也算是向量化编程的第二次练习，自己对其中向量的运算例如点运算，转置乘法有了更好的理解，这次编写的时候明显感觉比上次要熟悉了许多，而且也更能理解参数之间的转换关系。 |
| **参考文献**  **<https://wenku.baidu.com/view/f328b62b69dc5022abea0068.html>**  （同样参考了这个笔记）  <https://wenku.baidu.com/view/48509ca5dd3383c4bb4cd23b.html>  （一个ppt，部分内容与cousera相同，还有一些细节） |