Logistic Regression

Perceptron disadvantages:

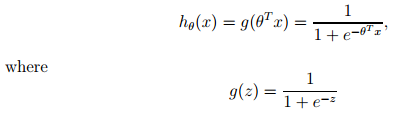
It never coverages if the classes are not perfectly linearly separable. Intuitively, we can think of the reason as the weights are continuously being updated since there is always at least one misclassified sample present in each epoch.

分类与回归的不同点在于分类的y值是离散值，而回归是连续值。

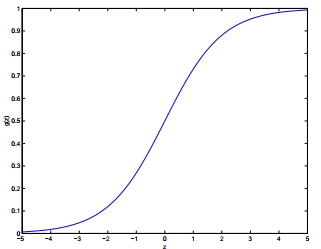
# Logistic regression (逻辑回归)

Simliar to the perceptron and Adaline, the logistic regression model in this chapter is also a linear mode for binary classification that can be extended to multiclass classification via the OvR technique.

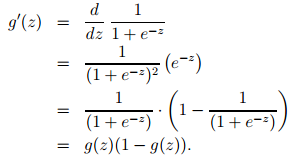
 逻辑回归中的假设（hypotheses）为:



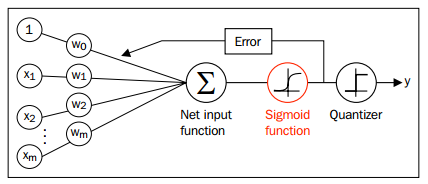
 该函数称为逻辑(logistic)函数或者S(sigmoid)型函数，函数具有如下图特点：



 并且该函数的导数有如下特点：



  In Adaline, we used the identity function as the activation function. In logistic regression, this activation function simply becomes the sigmoid function that we defined earlier, which is illustrated in the following figure:



The output of the sigmoid function is then interpreted as the probability of particular sample belong to class 1 ,given its feature x parameterized by the weights w. The predicted probability can then simply be converted into a binary outcome via a quantizer (unit step function):



    既然有了假设函数（hypotheses）,如何得到C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(3).png，进行如下的一些推导：

    \*假设

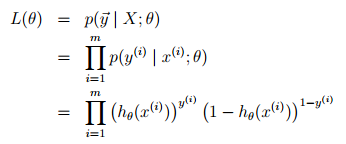
C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(4).png

      h = estimated probability that y = 1 on input x.

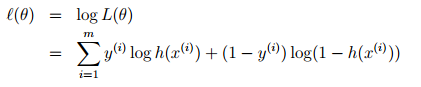
      该式可写成：

C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(5).png

       假设m个训练集是相互独立(independent of one another)的，这样参数的可能性可以写成：



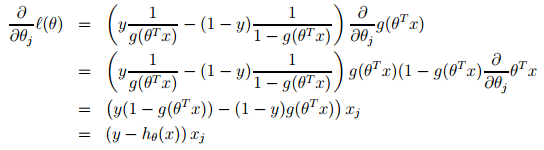
       然后写成对数的形式，这样最大化比较容易：



      这样可得到逻辑回归的cost function C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(10).png.

We can convert the product of factors into a summation of factors, which makes it easier to obtain the derivative of this function via the addition trick, as you may remember from calculus.

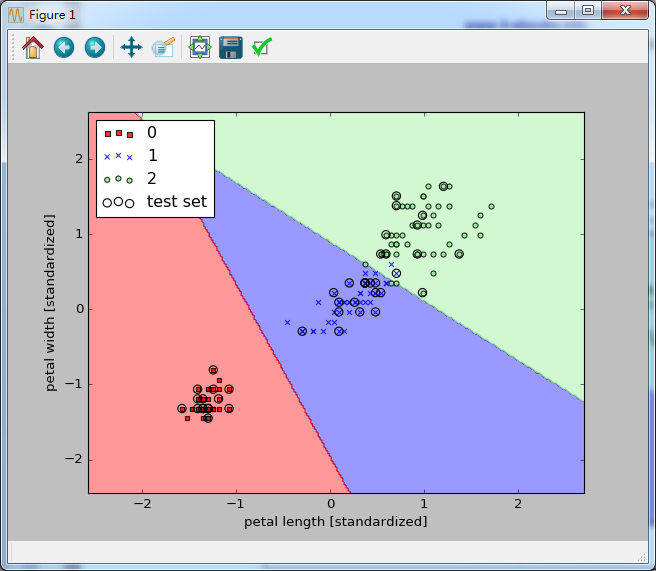
      如何来最大化这个函数呢，与前面最小化线性回归相似，可以使用梯度上升（gradient ascent）的形式,每一次迭代过程中改变C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(11).png值，C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(12).png。梯度上升的更新规则：



C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(14).png

C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(15).png(向量化)

Figures after classify using logistic regression:



# 2.最小化的（- C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(16).png）其它算法（逻辑回归中）

   -- Gradient descent

  -- Conjugate gradient

   -- BFGS

  -- L- BFGS

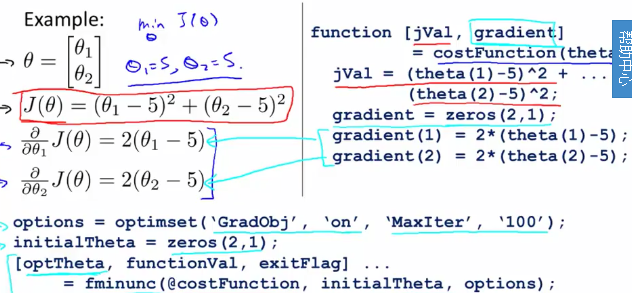
 Advantages:

  -- No need to manually pick a

  -- often faster than gradient descent

 Disadvantages:

 -- More complex



# 3.Multiclass Classification: one vs all

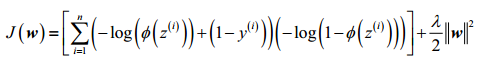
Train a logistic regression classifier C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(24).png for each class i to predict the probability that y = i.

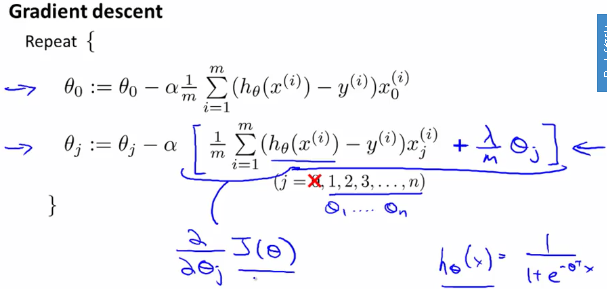
On a new input x, to make a prediction, pick the class i that maximizes

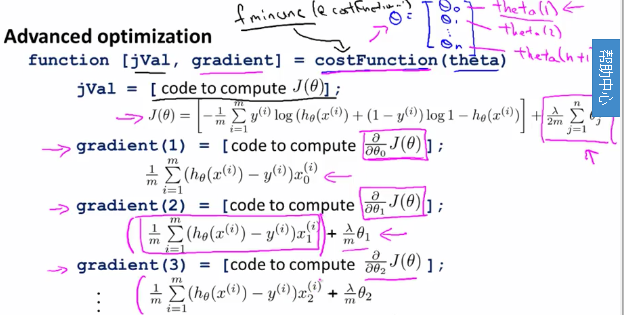
C:\Users\phenix\AppData\Local\Temp\enhtmlclip\Image(25).png

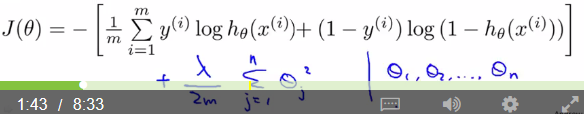
4. Regularized Logistic Regression

Cost Function:









Reference:

1. Machine learning : Ng Video:
2. Python machine learning, book