机器学习 week3

目录

1 Classification

To attempt classification, one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.

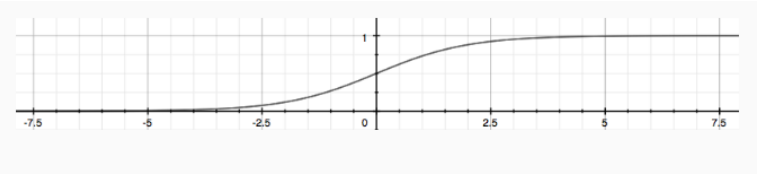
The classification problem is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values. For now, we will focus on the **binary classification** problem in which y can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.) For instance, if we are trying to build a spam classifier for email, then [Math Processing Error] may be some features of a piece of email, and y may be 1 if it is a piece of spam mail, and 0 otherwise. Hence, y 0,1. 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "-" and "+." Given [Math Processing Error], the corresponding [Math Processing Error] is also called the label for the training example.

2 Hypothesis Representation

We could approach the classification problem ignoring the fact that y is discrete-valued,and use our old linear regression algotithm to try to predict y given x.However,it is easy to construct examples where this method performs very poorly.Intuitively,it also doesn't make sense for[Math Processing Error]to take values larger than 1 or smaller than 0 when we know that y 0, 1.To fix this,let's change the form for our hypothese[Math Processing Error]to satisfy[Math Processing Error].This is accomplished by plugging [Math Processing Error]into the Logistic Function.

Our new form uses the "Sigmoid function,"also called the "Logistic Function":

The following image shows us what the sigmoid function looks like:



The function g(z),shown here,maps any real number to the (0,1)interval,making it useful for transforming an arbitrary-valued function into a function better suited for classification.

[Math Processing Error]will give us the **probability** that our output is 1.For example,[Math Processing Error]gives us a probability of 70% that our outputs is 1.Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%,then the probability that it is 30%).

3 Decision Boundary

In order to get our discrete 0 or 1 classification,we can translate the output of the hypothesis function as follows:

$h_\theta(x) \geq 0.5 \rightarrow$ y=1
$h_\theta(x) < 0.5 \rightarrow$ y=0

The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

g(z) ≥ 0.5
when z ≥ 0

Remember.

z=0,e^0=1 $\Rightarrow$ $g(z)$=1/2
z $\rightarrow \infty$,$e^{-\infty} \rightarrow 0 \Rightarrow$ $g(z)$=1
z $\rightarrow -\infty$,$e^{\infty} \rightarrow \infty \Rightarrow$ $g(z)$=0

So if our input to g is $\theta^T X$, then that means:

$h_\theta(x)$ = g($\theta^T x$) $\geq$0.5
when $\theta^T x \geq 0$

From these statements we can now say:

θ^Tx $\geq$0 $\Rightarrow$ y=1
θ^Tx <0 $\Rightarrow$ y=0

The **decision boundary** is the line that separates the area where y = 0 and where y = 1. It is created by our hypothesis function.

Example:

$$\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$$

y=1 if 5+(-1)x_1+0x_2 $\geq$0
5-x_1 $\geq$ 0
-x_1 $\geq$ -5
$x_1 \leqslant 5$

In this case, our decision boundary is a straight vertical line placed on the graph where x_1=5, and everything to the left of that denotes y = 1, while everything to the right denotes y = 0.

Again, the input to the sigmoid function $g(z)$ (e.g. θ^TX) doesn't need to be linear, and could be a function that describes a circle (e.g. z=θ_0+θ_1 x_1^2+θ_2 x_2^2) or any shape to fit our data.

4 Cost Function

We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

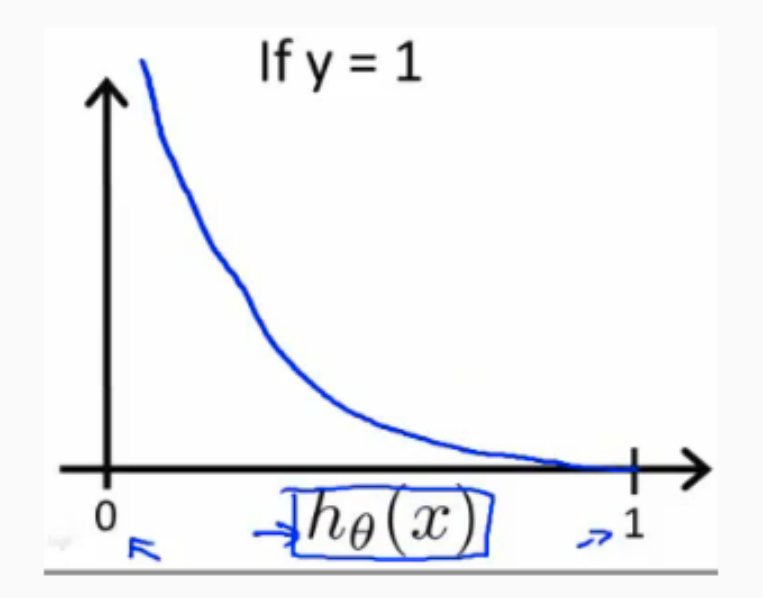
Instead, our cost function for logistic regression looks like:

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\text{Cost}(h_\theta(x^{(i)}),y^{(i)})$$

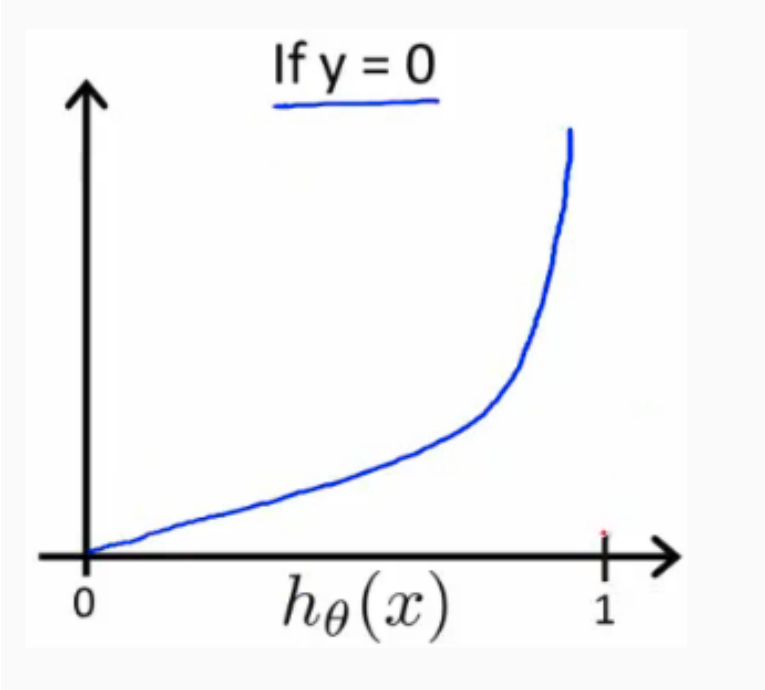
$$\text{Cost}(h_\theta(x),y)=-\log(h_\theta(x)) \quad \text{if } y=1$$

$$\text{Cost}(h_\theta(x),y)=-\log(1-h_\theta(x)) \quad \text{if } y=0$$

When y = 1, we get the following plot for $J()$ vs $h_\theta(x)$:



Similarly, when y = 0, we get the following plot for $J()$ vs $h_\theta(x)$:



Cost(h_θ(x),y)=0	if h_θ(x)=y
Cost(h_θ(x),y) $\to \infty$	if y=0 and h_θ(x) $\to$ 1
Cost(h_θ(x),y) $\to \infty$	if y=1 and h_θ(x) $\to$ 0

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

Note that writing the cost function in this way guarantees that $J(\theta)$ is convex for logistic regression.

5 Simplified Cost Function and Gradient Descent

Note: [6:53 - the gradient descent equation should have a 1/m factor]

We can compress our cost function's two conditional cases into one case:

Cost(h_θ(x),y)=-y log(h_θ(x))-(1-y)log(1-h_θ(x))

Notice that when y is equal to 1, then the second term (1−y)log(1−$h_\theta(x)$) will be zero and will not affect the result. If y is equal to 0, then the first term −ylog($h_\theta(x)$) will be zero and will not affect the result.

We can fully write out our entire cost function as follows:

$$\text{J}(\theta)=-\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)}))+(1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

A vectorized implementation is:

h=g(Xθ)

$$\text{J}(\theta)=\frac{1}{m}\cdot(-y^T\log(\text{h})-(1-y)^T\log(1-\text{h}))$$

5.1 Gradient Descent

Remember that the general form of gradient descent is:

Repeat

$$\theta_j:=\theta_j-\alpha\frac{\partial}{\partial\theta_j}\text{J}(\theta)$$

We can work out the derivative part using calculus to get:

Notice that this algorithm is identical to the one we used in linear regression. We still have to simultaneously update all values in theta.

A vectorized implementation is:

$$\theta:=\theta-\frac{\alpha}{m}X^T\ (\text{g}(\text{X}\theta)-\vec{y})$$

Repeat

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$