

逻辑函数

之前的分类方法对于某些情况表现十分不好，例如当分类多于两种，或者分类在0和1之间的情况。

针对这样的情况，我们用逻辑函数把分类转化为概率问题。

We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm 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 $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$. To fix this, let's change the form for our hypotheses $h_\theta(x)$ to satisfy $0 \leq h_\theta(x) \leq 1$. This is accomplished by plugging $\theta^T x$ into the Logistic Function.

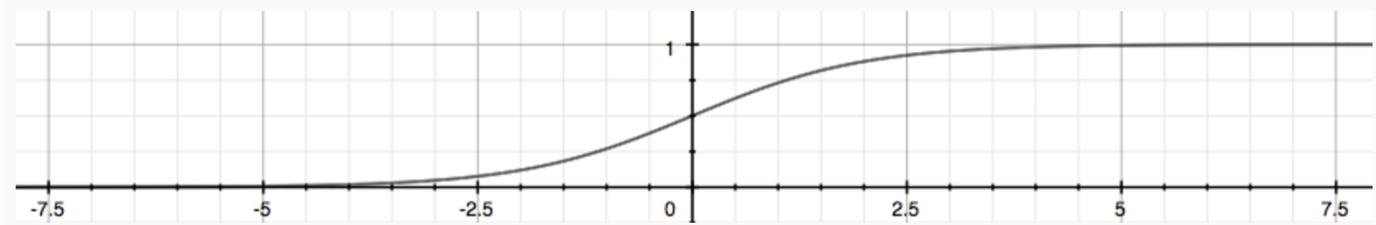
Our new form uses the "Sigmoid Function," also called the "Logistic Function":

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

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.

$h_\theta(x)$ will give us the **probability** that our output is 1. For example, $h_\theta(x) = 0.7$ gives us a probability of 70% that our output 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 0 is 30%).

$$h_\theta(x) = P(y = 1|x; \theta) = 1 - P(y = 0|x; \theta)$$

$$P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1$$

Decision Boundary

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

$$\begin{aligned} h_{\theta}(x) \geq 0.5 &\rightarrow y = 1 \\ h_{\theta}(x) < 0.5 &\rightarrow y = 0 \end{aligned}$$

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:

$$\begin{aligned} g(z) \geq 0.5 \\ \text{when } z \geq 0 \end{aligned}$$

Remember.

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

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

$$\begin{aligned} h_{\theta}(x) = g(\theta^T x) \geq 0.5 \\ \text{when } \theta^T x \geq 0 \end{aligned}$$

From these statements we can now say:

$$\begin{aligned} \theta^T x \geq 0 &\Rightarrow y = 1 \\ \theta^T x < 0 &\Rightarrow y = 0 \end{aligned}$$

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:

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

$$y = 1 \text{ if } 5 + (-1)x_1 + 0x_2 \geq 0$$

$$5 - x_1 \geq 0$$

$$-x_1 \geq -5$$

$$x_1 \leq 5$$

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. $\theta^T X$) doesn't need to be linear, and could be a function that describes a circle (e.g. $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$) or any shape to fit our data.

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:

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

$$\begin{aligned} \text{Cost}(h_\theta(x), y) &= -\log(h_\theta(x)) && \text{if } y = 1 \\ \text{Cost}(h_\theta(x), y) &= -\log(1 - h_\theta(x)) && \text{if } y = 0 \end{aligned}$$

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

$$\text{Cost}(h_\theta(x), y) \rightarrow \infty \text{ if } y = 0 \text{ and } h_\theta(x) \rightarrow 1$$

$$\text{Cost}(h_\theta(x), y) \rightarrow \infty \text{ if } y = 1 \text{ and } h_\theta(x) \rightarrow 0$$

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.

牛逼的地方来了，怎么把逻辑递归写成一行

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

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

因为 $y=0$ 或 $y=1$ 都会导致其中一项的系数为0，实际上就等于上述两行分开的方程式

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

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

A vectorized implementation is:

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

Gradient Descent

Remember that the general form of gradient descent is:

$$\begin{aligned} &\text{Repeat } \{ \\ &\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\ &\} \end{aligned}$$

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

$$\begin{aligned} &\text{Repeat } \{ \\ &\quad \theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \\ &\} \end{aligned}$$

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:

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

Advanced Optimization

优化函数在Octave中的写法（栗子）：

```
function [jVal, gradient] = costFunction(theta)
jVal = (theta(1) - 5)^2 + ... + (theta(2) - 5)^2;
gradient = zeros(2, 1);           ← 梯度向量的两个元素
gradient(1) = 2 * (theta(1) - 5);
gradient(2) = 2 * (theta(2) - 5);
```

options = optimset('GradObj', 'on', 'MaxIter', 100); ← 存储你想要的options的数据结构

设置梯度目标参数为打开 ↑ ↑ 设置迭代次数为100

initialTheta = zeros(2,1);

[optTheta, functionVal, exitFlag] ... = fminunc(@costFunction, initialTheta, options);

@: 指向刚定义的costFunction函数

The `@`-symbol means that what we are sending to the *fmincg* - function, *lrCostFunction*, is not a matrix, vector or number, but a function. And the `(t)` symbol means that it is a function of *t*. *lrCostFunction* is a function of four variables, but with the command `@(t) lrCostFunction(t, X, (y==c), lambda)` it becomes a function of one variable, *t*, and the rest of the variables are set to *X*, *(y==c)*, and *lambda*.

We use this notation because *fmincg* minimize a function with respect to one variable.

优点：可以自动选择学习速率

fminunc: 【高级函数】无约束最小化函数

多类分类问题 (one vs all)

Now we will approach the classification of data when we have more than two categories. Instead of $y = \{0,1\}$ we will expand our definition so that $y = \{0,1,\dots,n\}$.

Since $y = \{0,1,\dots,n\}$, we divide our problem into $n+1$ (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

$$\begin{aligned}
y &\in \{0, 1 \dots n\} \\
h_{\theta}^{(0)}(x) &= P(y = 0|x; \theta) \\
h_{\theta}^{(1)}(x) &= P(y = 1|x; \theta) \\
&\dots \\
h_{\theta}^{(n)}(x) &= P(y = n|x; \theta) \\
\text{prediction} &= \max_i(h_{\theta}^{(i)}(x))
\end{aligned}$$

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

To summarize:

Train a logistic regression classifier $h_{\theta}(x)$ for each class to predict the probability that $y = i$.

To make a prediction on a new x , pick the class that maximizes $h_{\theta}(x)$

The Problem of Overfitting

Overfitting, or high variance, is caused by a hypothesis function that fits the available data but does not generalize well to predict new data. It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

This terminology is applied to both linear and logistic regression. There are two main options to address the issue of overfitting:

1) Reduce the number of features:

- Manually select which features to keep.
- Use a model selection algorithm (studied later in the course).

2) Regularization

- Keep all the features, but reduce the magnitude of parameters
- Regularization works well when we have a lot of slightly useful features.

当特征很多又不知道哪个比较重要的时候，正规化登场了：

$$\min_{\theta} \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

lambda是正规化参数，用来平衡拟合度和theta值的大小，正规通过惩罚theta0以外的所有参数来达到使theta最小的目的

The λ , or lambda, is the **regularization parameter**. It determines how much the costs of our theta parameters are inflated.

Regularized Linear Regression

We can apply regularization to both linear regression and logistic regression. We will approach linear regression first.

Gradient Descent

We will modify our gradient descent function to separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0 .

Repeat {

$$\begin{aligned}\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \\ \theta_j &:= \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \quad j \in \{1, 2, \dots, n\} \\ \}\end{aligned}$$

The term $\frac{\lambda}{m} \theta_j$ performs our regularization. With some manipulation our update rule can also be represented as:

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

The first term in the above equation, $1 - \alpha \frac{\lambda}{m}$ will always be less than 1. Intuitively you can see it as reducing the value of θ_j by some amount on every update. Notice that the second term is now exactly the same as it was before.

Normal Equation

Now let's approach regularization using the alternate method of the non-iterative normal equation.

To add in regularization, the equation is the same as our original, except that we add another term inside the parentheses:

$$\theta = (X^T X + \lambda \cdot L)^{-1} X^T y$$

where $L = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix}$

L is a matrix with 0 at the top left and 1's down the diagonal, with 0's everywhere else. It should have dimension $(n+1) \times (n+1)$. Intuitively, this is the identity matrix (though we are not including x_0), multiplied with a single real number λ .

Recall that if $m \leq n$, then $X^T X$ is non-invertible. However, when we add the term $\lambda \cdot L$, then $X^T X + \lambda \cdot L$ becomes invertible.

Regularized Logistic Regression

Cost Function

Recall that our cost function for logistic regression was:

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

We can regularize this equation by adding a term to the end:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

The second sum, $\sum_{j=1}^n \theta_j^2$ means to explicitly exclude the bias term, θ_0 . I.e. the θ vector is indexed from 0 to n (holding $n+1$ values, θ_0 through θ_n), and this sum explicitly skips θ_0 , by running from 1 to n , skipping 0. Thus, when computing the equation, we should continuously update the two following equations:

Gradient descent

Repeat {

$$\rightarrow \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\rightarrow \theta_j := \theta_j - \alpha \left[\underbrace{\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\substack{(j = 1, 2, 3, \dots, n) \\ \theta_1, \dots, \theta_n}} + \frac{\lambda}{m} \theta_j \right] \leftarrow$$

}

$$\frac{\partial}{\partial \theta_j} \underline{\mathcal{J}(\theta)}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$