Simplified Cost Function and Gradient Descent

Logistic Regression cost function

The following ecuation is our cost function for logistic regression. Our overall cost function is 1 over m times the sum over the training set of the cost of making different predictions on the different examples of labels $y^{(i)}$. For classification problems in our training sets, and in fact even for examples, now that our training set y is always equal to zero or one.

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

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

$$\text{Note: } y = 0 \text{ or } 1 \text{ always}$$

Rather than writing out this cost function on two separate lines with two separate cases, we can take these two lines and compress them into one equation, and this would make it more convenient to write out a cost function and derive gradient descent. Concretely, we can write out the cost function as:

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

We know that there are only two possible cases:

- If $y = 1$ the second term is equal to 0 by $(1 - 1) \log(1 - h_\theta(x))$ = $(0) \log(1 - h_\theta(x))$ = 0

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

- If $y = 0$ the first term is equal to 0 by $-(0) \log(h_\theta(x)) = 0$

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

So this shows that this definition for the cost function is just a more compact way of taking both of these expressions, the cases $y = 1$ and $y = 0$, and writing them in a more convenient form with just one line. We can therefore write all our cost functions for logistic regression as:

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

Given this cost function, in order to fit the parameters, what we're going to do then is try to find the parameters theta that minimize $J(\theta)$.

To fit parameters θ: $min_\theta J(\theta)$

Finally, if we're given a new example with some set of features x (to make a prediction given new x), we can then take the θ that we fit to our training set and output our prediction as:

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

And the output of our hypothesis we're going to interpret as: $P(y = 1 | x; \theta)$

The way we're going to minimize the cost function is using gradient descent:

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

Want: $min_\theta J(\theta)$:

Repeat {

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

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

} (simultaneously update all θ_j)

So if we have n features, we would have a parameter vector theta: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix}$ which with parameters θ_0, θ_1, down to θ_n. And we will use this update to simultaneously update all of our values of theta (the Algorithm looks identical to linear regression). So, are linear regression and logistic regression different algorithms or not? This is resolved by observing that for logistic regression, what has changed is that the definition for the hypothesis function has changed

Video Question 1: Suppose you are running gradient descent to fit a logistic regression model with parameter $\theta \in \mathbb{R}^{n+1}$. Which of the following is a reasonable way to make sure the learning rate α is set properly and that gradient descent is running correctly?

- Plot $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ as a function of the number of iterations (i.e. the horizontal axis is the iteration number) and make sure $J(\theta)$ is decreasing on every iteration.

> Plot $J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$ as a function of the number of iterations and make sure $J(\theta)$ is decreasing on every iteration.

- Plot $J(\theta)$ as a function of θ and make sure it is decreasing on every iteration.
- Plot $J(\theta)$ as a function of θ and make sure it is convex.

Video Question 2: One iteration of gradient descent simultaneously performs these updates: One iteration of gradient descent simultaneously performs these updates:

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

$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$

...

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

We would like a vectorized implementation of the form $\theta := \theta - \alpha\delta$ (for some vector $\delta \in \mathbb{R}^{n+1}$).

What should the vectorized implementation be?

> $\theta := \theta - \alpha \frac{1}{m} \sum_{i=1}^{m} [(h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}]$

- $\theta := \theta - \alpha \frac{1}{m} [\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})] \cdot x^{(i)}$
- $\theta := \theta - \alpha \frac{1}{m} x^{(i)} [\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})]$
- All of the above are correct implementations.

Finally, the idea of feature scaling also applies to gradient descent for logistic regression. If we have features that are on very different scale, then applying feature scaling can also make gradient descent run faster for logistic regression.

Summary

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

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

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

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:

$$\text{Repeat}\{$$

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

$$\}$$

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

$$\text{Repeat}\{$$

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

$$\}$$

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

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