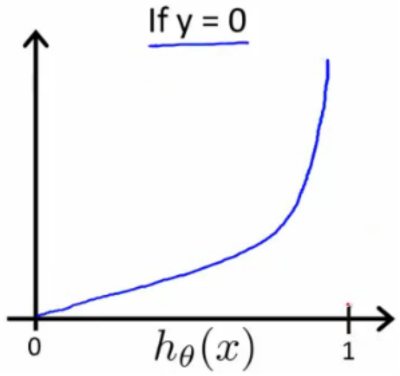

The more our hypothesis is off from y, the larger the cost function output. If our hypothesis is equal to y, then our cost is 0:

$$\begin{aligned}&\mathrm{Cost}(h_\theta(x),y)=0\text{ if }h_\theta(x)=y\\&\mathrm{Cost}(h_\theta(x),y)\to\infty\text{ if }y=0\text{ and }h_\theta(x)\to 1\\&\mathrm{Cost}(h_\theta(x),y)\to\infty\text{ if }y=1\text{ and }h_\theta(x)\to 0\end{aligned}$$

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(θ) is convex for logistic regression.

Simplified Cost Function and Gradient Descent

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

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

Notice that when y is equal to 1, then the second term $(1-y)\log(1-h_\theta(x))$ 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:

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

A vectorized implementation is:

$$\begin{aligned}&h=g(X\theta)\\&J(\theta)=\frac{1}{m}\cdot\left(-y^T\log(h)-(1-y)^T\log(1-h)\right)\end{aligned}$$

Gradient Descent

Remember that the general form of gradient descent is:

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

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

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

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:

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

Partial derivative of J(θ)

First calculate derivative of sigmoid function (it will be useful while finding partial derivative of J(θ)):

$$
\begin{aligned}
\sigma(x)' &= \left(\frac{1}{1+e^{-x}}\right)' = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \frac{-1' - (e^{-x})'}{(1+e^{-x})^2} = \frac{0 - (-x)'(e^{-x})}{(1+e^{-x})^2} = \frac{-(-1)(e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}}{1+e^{-x}}\right) = \sigma(x)\left(\frac{+1-1+e^{-x}}{1+e^{-x}}\right) = \sigma(x)\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x))
\end{aligned}
$$

Now we are ready to find out resulting partial derivative:

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{-1}{m} \sum_{i=1}^{m} \left[y^{(i)} log(h_\theta(x^{(i)})) + (1-y^{(i)}) log(1-h_\theta(x^{(i)})) \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \frac{\partial}{\partial \theta_j} log(h_\theta(x^{(i)})) + (1-y^{(i)}) \frac{\partial}{\partial \theta_j} log(1-h_\theta(x^{(i)})) \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} \frac{\partial}{\partial \theta_j} h_\theta(x^{(i)})}{h_\theta(x^{(i)})} + \frac{(1-y^{(i)}) \frac{\partial}{\partial \theta_j} (1-h_\theta(x^{(i)}))}{1-h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} \frac{\partial}{\partial \theta_j} \sigma(\theta^T x^{(i)})}{h_\theta(x^{(i)})} + \frac{(1-y^{(i)}) \frac{\partial}{\partial \theta_j} (1-\sigma(\theta^T x^{(i)}))}{1-h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} \sigma(\theta^T x^{(i)})(1-\sigma(\theta^T x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{h_\theta(x^{(i)})} + \frac{-(1-y^{(i)}) \sigma(\theta^T x^{(i)})(1-\sigma(\theta^T x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{1-h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} h_\theta(x^{(i)})(1-h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{h_\theta(x^{(i)})} - \frac{(1-y^{(i)}) h_\theta(x^{(i)})(1-h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{1-h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} (1-h_\theta(x^{(i)})) x_j^{(i)} - (1-y^{(i)}) h_\theta(x^{(i)}) x_j^{(i)} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} (1-h_\theta(x^{(i)})) - (1-y^{(i)}) h_\theta(x^{(i)}) \right] x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} - y^{(i)} h_\theta(x^{(i)}) - h_\theta(x^{(i)}) + y^{(i)} h_\theta(x^{(i)}) \right] x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} - h_\theta(x^{(i)}) \right] x_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x_j^{(i)}
\end{aligned}
$$

The vectorized version;

$\nabla J(\theta) = \frac{1}{m} \cdot X^T \cdot \left(g(X \cdot \theta) - \vec{y}\right)$

Advanced Optimization

"Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize θ that can be used instead of gradient descent. A. Ng suggests not to write these more sophisticated algorithms yourself (unless you are an expert in numerical computing) but use the libraries instead, as they're already tested and highly optimized. Octave provides them.

We first need to provide a function that evaluates the following two functions for a given input value θ:

$$
\begin{aligned}
& J(\theta) \\
& \frac{\partial}{\partial \theta_j} J(\theta)
\end{aligned}
$$

We can write a single function that returns both of these: