Suppose we want to minimize $J(\theta)$ as a function of θ. For this example, suppose $J : \Re \mapsto \Re$, so that $\theta \in \Re$. If we are using `minFunc` or some other optimization algorithm, then we usually have implemented some function $g(\theta)$ that purportedly computes $\frac{d}{d\theta}J(\theta)$.

How can we check if our implementation of g is correct?

Recall the mathematical definition of the derivative as:

$$\frac{d}{d\theta}J(\theta) = \lim_{\epsilon \to 0} \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}.$$

Thus, at any specific value of θ, we can numerically approximate the derivative as follows:

$$\frac{J(\theta + \text{EPSILON}) - J(\theta - \text{EPSILON})}{2 \times \text{EPSILON}}$$

In practice, we set EPSILON to a small constant, say around 10^{-4}. (There's a large range of values of EPSILON that should work well, but we don't set EPSILON to be "extremely" small, say 10^{-20}, as that would lead to numerical roundoff errors.)

Thus, given a function $g(\theta)$ that is supposedly computing $\frac{d}{d\theta}J(\theta)$, we can now numerically verify its correctness by checking that

$$g(\theta) \approx \frac{J(\theta + \text{EPSILON}) - J(\theta - \text{EPSILON})}{2 \times \text{EPSILON}}.$$

The degree to which these two values should approximate each other will depend on the details of J. But assuming $\text{EPSILON} = 10^{-4}$, you'll usually find that the left- and right-hand sides of the above will agree to at least 4 significant digits (and often many more).

Now, consider the case where $\theta \in \Re^n$ is a vector rather than a single real number (so that we have n parameters that we want to learn), and $J : \Re^n \mapsto \Re$. We now generalize our derivative checking procedure to the case where θ may be a vector (as in our linear regression and logistic regression examples). If ever we are optimizing over several variables or over matrices, we can always pack these parameters into a long vector and use the same method here to check our derivatives. (This will often need to be done anyway if you want to use off-the-shelf optimization packages.)

Suppose we have a function $g_i(\theta)$ that purportedly computes $\frac{\partial}{\partial\theta_i}J(\theta)$; we'd like to check if g_i is outputting correct derivative values. Let $\theta^{(i+)} = \theta + \text{EPSILON} \times \vec{e_i}$, where

$$g_i(\theta) \approx \frac{J(\theta^{(i+)}) - J(\theta^{(i-)})}{2 \times \mathrm{EPSILON}}.$$

Gradient checker code

As an exercise, try implementing the above method to check the gradient of your linear regression and logistic regression functions. Alternatively, you can use the provided `ex1/grad_check.m` file (which takes arguments similar to `minFunc`) and will check $\frac{\partial J(\theta)}{\partial \theta_i}$ for many random choices of i.