|  |
| --- |
| function numgrad = computeNumericalGradient(J, theta) |
|  | %COMPUTENUMERICALGRADIENT Computes the gradient using "finite differences" |
|  | %and gives us a numerical estimate of the gradient. |
|  | % numgrad = COMPUTENUMERICALGRADIENT(J, theta) computes the numerical |
|  | % gradient of the function J around theta. Calling y = J(theta) should |
|  | % return the function value at theta. |
|  |  |
|  | % Notes: The following code implements numerical gradient checking, and |
|  | % returns the numerical gradient.It sets numgrad(i) to (a numerical |
|  | % approximation of) the partial derivative of J with respect to the |
|  | % i-th input argument, evaluated at theta. (i.e., numgrad(i) should |
|  | % be the (approximately) the partial derivative of J with respect |
|  | % to theta(i).) |
|  | % |
|  |  |
|  | numgrad = zeros(size(theta)); |
|  | perturb = zeros(size(theta)); |
|  | e = 1e-4; |
|  | for p = 1:numel(theta) |
|  | % Set perturbation vector |
|  | perturb(p) = e; |
|  | loss1 = J(theta - perturb); |
|  | loss2 = J(theta + perturb); |
|  | % Compute Numerical Gradient |
|  | numgrad(p) = (loss2 - loss1) / (2\*e); |
|  | perturb(p) = 0; |
|  | end |
|  |  |
|  | end |