

# The Coders' Club

## Machine Learning: G1

### Week 5: Assignment

**Topic:**

Neural Networks: Learning

**Some Additional Courses:**

- Machine Learning Onramp (MathWorks)  
<https://www.mathworks.com/learn/tutorials/machine-learning-onramp.html>
- Deep Learning Onramp (MathWorks)  
<https://www.mathworks.com/learn/tutorials/deep-learning-onramp.html>
- AI From the Data Center to the Edge – An Optimized Path Using Intel® Architecture (Intel AI)  
<https://software.intel.com/en-us/ai/courses/data-center-to-edge>
- Machine Learning (Intel)  
<https://software.intel.com/en-us/ai/courses/machine-learning>
- Deep Learning (Intel)  
<https://software.intel.com/en-us/ai/courses/deep-learning>

## Neural Networks: Learning

Q.1. You are training a three layer neural network and would like to use backpropagation to compute the gradient of the cost function. In the backpropagation algorithm, one of the steps is to update

$$\Delta_{ij}^{(2)} = \Delta_{ij}^{(2)} + \delta_i^{(3)} * (a^{(2)})_j$$

for every  $i, j$ . Which of the following is a correct vectorization of this step?

- $\Delta^{(2)} := \Delta^{(2)} + \delta^{(3)} * (a^{(3)})^T$
- $\Delta^{(2)} := \Delta^{(2)} + (a^{(3)})^T * \delta^{(3)}$
- $\Delta^{(2)} := \Delta^{(2)} + \delta^{(2)} * (a^{(2)})^T$
- $\Delta^{(2)} := \Delta^{(2)} + \delta^{(3)} * (a^{(2)})^T$

Q.2. Suppose `Theta1` is a 5x3 matrix, and `Theta2` is a 4x6 matrix. You set `thetaVec = [ Theta1(:); Theta2(:) ]`. Which of the following correctly recovers `Theta2`?

- `reshape(thetaVec(16:39), 4, 6)`
- `reshape(thetaVec(15:38), 4, 6)`
- `reshape(thetaVec(16:24), 4, 6)`
- `reshape(thetaVec(15:39), 4, 6)`
- `reshape(thetaVec(16:39), 6, 4)`

Q.3. Let  $J(\theta) = 3\theta^4 + 4$ . Let  $\theta = 1$  and  $\epsilon = 0.01$ . Use the formula

$$\frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$

to numerically compute an approximation to the derivative at  $\theta = 1$ . What value do you get? (When  $\theta = 1$ , the true/exact derivative is,

$$\frac{d}{d\theta} J(\theta) = 12).$$

- 12
- 11.9988
- 6
- 12.0012

Q.4. Which of the following statements are true? Check all that apply.

- For computational efficiency, after we have performed gradient checking to verify that our backpropagation code is correct, we usually disable gradient checking before using backpropagation to train the network.
- Computing the gradient of the cost function in a neural network has the same efficiency when we use backpropagation or when we numerically compute it using the method of gradient checking.
- Using gradient checking can help verify if one's implementation of backpropagation is bug-free.
- Gradient checking is useful if we are using one of the advanced optimization methods (such as in `fminunc`) as our optimization algorithm. However, it serves little purpose if we are using gradient descent.

Q.5. Which of the following statements are true? Check all that apply.

- Suppose that the parameter  $\Theta^{(1)}$  is a square matrix (meaning the no. of rows equals the no. of columns). If we replace  $\Theta^{(1)}$  with its transpose  $(\Theta^{(1)})^T$  then we have not changed the function that the network is computing.
- Suppose we are using gradient descent with learning rate  $\alpha$ . For logistic regression and linear regression,  $J(\theta)$  was a convex optimization problem and thus we did not want to choose a learning rate  $\alpha$  that is too large. For a neural network however,  $J(\Theta)$  may not be convex, and thus choosing a very large value of  $\alpha$  can only speed up convergence.
- If we are training a neural network using gradient descent, one reasonable “debugging” step to make sure it is working is to plot  $J(\Theta)$  as a function of number of iterations, and make sure it is decreasing (or at least non-increasing) after each iteration.
- Suppose we have a correct implementation of backpropagation, and are training a neural network using gradient descent. Suppose we plot  $J(\Theta)$  as a function of number of iterations, and find that it is **increasing** rather than decreasing. One possible cause of this is that the learning rate  $\alpha$  is too large.