Neural Networks: Learning

Question 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

$$\Delta^{(2)}_{ij} := \Delta^{(2)}_{ij} + \delta^{(3)}_i * (a^{(2)})_j$$

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

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

  > This version is correct, as it takes the "outer product" of the two verctors $\delta^{(3)}$ and $a^{(2)}$ which is a matrix such that the (i,j)-th entry is $\delta^{(3)}_i * (a^{(2)})_j$ as desired.

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

Question 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)**

  > This choice is correct, since **Theta1** has 15 elements, so **Theta2** begins at index 16 and ends at index 16 + 24 - 1 = 39.

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

Question 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 derivatove is $\frac{dJ(\theta)}{d\theta} = 12$.)

- [ ] 11.9988
- [ ] 12

- [ ] 6
- [✓] **12.0012**

> $\frac{(3(1.01)^4+4)-(3(0.99)^4+4)}{2(0.01)} = 12.0012$

Question 4

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

- [ ] 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.
- [ ] 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.
- [✓] **Using gradient checking can help verify if one's implementation of vackpropagation is bug-free**

> If the gradient computed by backpropogation is the same as one computed numerically with gradient checking, this is very strong evidence that you have a correct implementation of backprpogation.

- [✓] **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.**

> Checking the gradient numerically is debugging tool: it helps ensure a correct implementation, but it is too slow to use a method for actually computing gradients.

Question 5

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

- [ ] Supppose that the parameter $\Theta^{(1)}$ is a square matrix (meaning the number of rows equals the number 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 have a correct implementation of backpropagation, and are training a neural network using gradient descent. Suppose we plot $J(\Theta)$ as a function of the number of iterations, and find that it is increasing rather than decreasing. One possible cause of this is that the learning rate α is too large.**

> If the learning rate is too large, the cost function can diverge can diverge during gradient descent. Thus, you should select a smaller value of α.

☐ Suppose we are using gradient descent with learnign rate α. For logistic regression and linear regression, $J(\theta)$ was a convex optimization problem and thus we did not want to choose a learning rate α that is too large. For a neural network however, $J(\Theta)$ may not be convex, and thus choosing a very large value of α 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 afunction of the number of iterations, and make sure it is decreasing (or at least non-increasing) after each iteration.**

> Since gradient descent uses the gradient to take a step toward parameters with lower cost (ie, lower $J(\Theta)$), the value of $J(\Theta)$ should be equal or less at each iteration if the gradient computation is correct and the learning rate is set properly.