✓ **Congratulations! You passed!**
TO PASS 80% or higher

Keep Learning

GRADE
100%

Neural Networks: Learning

LATEST SUBMISSION GRADE
100%

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**

1 / 1 point

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

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

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

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

2. **Suppose $$\tt{Theta1}$$ is a 5x3 matrix, and $$\tt{Theta2}$$ is a 4x6 matrix. You set $$\tt{thetaVec = [Theta1(:) ; Theta2(:)]}$$. Which of the following correctly recovers $$\tt{Theta2}$$?**

1 / 1 point

- (●) $$\tt{reshape(thetaVec(16:39), 4, 6)}$$
- ( ) $$\tt{reshape(thetaVec(15:38), 4, 6)}$$
- ( ) $$\tt{reshape(thetaVec(16:24), 4, 6)}$$
- ( ) $$\tt{reshape(thetaVec(15:39), 4, 6)}$$
- ( ) \tt{reshape(thetaVec(16:39), 6, 4)}

✓ **Correct**
This choice is correct, since $$\tt{Theta1}$$ has 15 elements, so $$\tt{Theta2}$$ begins at index 16 and ends at index 16 + 24 - 1 = 39.

3. **Let $J(\theta) = 2\theta^4 + 2$. 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 $\frac{dJ(\theta)}{d\theta} = 8$.)**

1 / 1 point

- ( ) 7.9992
- ( ) 10
- (●) 8.0008
- ( ) 8

✓ **Correct**
We compute $\frac{(2(1.01)^4+2) - (2(0.99)^4+2)}{2(0.01)} = 8.0008.$

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

1 / 1 point

- [x] Using gradient checking can help verify if one's implementation of backpropagation is bug-free.

✓ **Correct**
If the gradient computed by backpropagation is the same as one computed numerically with gradient checking, this is very strong evidence that you have a correct implementation of backpropagation.

- [ ] Using a large value of λ cannot hurt the performance of your neural network; the only reason we do not set λ to be too large is to avoid numerical problems.
- [x] If our neural network overfits the training set, one reasonable step to take is to increase the regularization parameter λ.

regularization parameter λ.

✓ **Correct**
Just as with logistic regression, a large value of λ will penalize large parameter values, thereby reducing the changes of overfitting the training set.

- [ ] Gradient checking is useful if we are using gradient descent as our optimization algorithm. However, it serves little purpose if we are using one of the advanced optimization methods (such as in fminunc).

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

1 / 1 point

- [ ] Suppose we are using gradient descent with learning 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.

- [x] 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 the number of iterations, and make sure it is decreasing (or at least non-increasing) after each iteration.

✓ **Correct**
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.

- [ ] Suppose 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.

- [x] 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.

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