Feedback — IX. Neural Networks: Learning

Help

You submitted this quiz on **Fri 22 Nov 2013 6:43 PM PST**. You got a score of **4.00** out of **5.00**. You can attempt again in 10 minutes.

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 $\Delta_{ij}^{(2)} := \Delta_{ij}^{(2)} + \delta_i^{(3)} * (a^{(2)})_j$ for every i, j. Which of the following is a correct vectorization of this step?

Your Answer		Score	Explanation
○ $\Delta^{(2)} := \Delta^{(2)} + (a^{(2)})^T * \delta^{(2)}$			
○ $\Delta^{(2)} := \Delta^{(2)} + \delta^{(2)} * (a^{(2)})^T$			
● $\Delta^{(2)} := \Delta^{(2)} + \delta^{(3)} * (a^{(3)})^T$	✗	0.00	This choice incorrectly uses $a^{(3)}$ instead of $a^{(2)}$.
○ $\Delta^{(2)} := \Delta^{(2)} + \delta^{(3)} * (a^{(2)})^T$			
Total		0.00 / 1.00	

Question 2

Suppose `Theta1` is a 2x5 matrix, and `Theta2` is a 3x6 matrix. You set `thetaVec = [Theta1(:) ; Theta2(:)]`. Which of the following correctly recovers `Theta2`?

Your Answer	Score	Explanation

○ `reshape(thetaVec(10:27), 6, 3)`			
○ `reshape(thetaVec(11:20), 3, 6)`			
○ `reshape(thetaVec(10:27), 3, 6)`			
● `reshape(thetaVec(11:28), 3, 6)`	✓	1.00	This choice is correct, since `Theta1` has 10 elements, so `Theta2` begins at index 11 and ends at index 11 + 18 - 1 = 28.
Total		1.00 / 1.00	

Question 3

Let $J(\theta) = 2\theta^3 + 2$. 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 derivati ve is $\frac{dJ(\theta)}{d\theta} = 6$.)

Your Answer		Score	Explanation
○ 6			
● 6.0002	✓	1.00	We compute $\frac{(2(1.01)^3+2)-(2(0.99)^3+2)}{2(0.01)} = 6.0002$.
○ 8			
○ 5.9998			
Total		1.00 / 1.00	

Question 4

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

Your Answer		Score	Explanation
☑ Using gradient checking can help verify if one's implementation of backpropagation is bug-free.	✓	0.25	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.
☐ Gradient checking is useful if we are using one of the advanced optimization methods (such as in f minunc) as our optimization algorithm. However, it serves little purpose if we are using gradient descent.	✓	0.25	Gradient descent depends on the computation of correct gradient values at different parameter settings. Gradient checking ensures the computed values are correct.
☑ 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.	✓	0.25	Checking the gradient numerically is a debugging tool: it helps ensure a corre ct implementation, but it is too slow to use as a method for actually computing gradients.
☐ 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	✓	0.25	A large value of λ can be quite detrimental. If you set it too high, then the network will be underfit to the training data and give poor predictions on both training data and new, unseen test data.

problems.

Total	1.00 / 1.00	

Question 5

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

Your Answer		Score	Explanation
✓ 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.	✓	0.25	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 you have a three layer network with parameters $\Theta^{(1)}$ (controlling the function mapping from the inputs to the hidden units) and $\Theta^{(2)}$ (controlling the mapping from the hidden units to the outputs). If we set all the elements of	✓	0.25	Since the parameters are the same within layers, every unit in each layer will receive the same update during backpropagation. The result is that such an initialization does not break symmetry.

$\Theta^{(1)}$ to be 0, and all the elements of $\Theta^{(2)}$ to be 1, then this suffices for symmetry breaking, since the neurons are no longer all computing the same function of the input.			
☑ Suppose you are training a neural network using gradient descent. Depending on your random initialization, your algorithm may converge to different local optima (i.e., if you run the algorithm twice with different random initializations, gradient descent may converge to two different solutions).	✓	0.25	The cost function for a neural network is non-convex, so it may have multiple minima. Which minimum you find with gradient descent depends on the initialization.
☐ 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	✓	0.25	$\Theta^{(1)}$ can be an arbitrary matrix, so when you compute $a^{(2)} = g(\Theta^{(1)}a^{(1)})$, replacing $\Theta^{(1)}$ with its transpose will compute a different value.

computing.

Total	1.00 / 1.00