Suppose we want to try to minimize $J(\Theta)$ as a function of Θ, using one of the advanced optimization methods (fminunc, conjugate gradient, BFGS, L-BFGS, etc.). What do we need to supply code to compute (as a function of Θ)?

$J(\Theta)$ and the (partial) derivative terms $\frac{\partial}{\partial\Theta_{ij}^{(l)}}$ for every i, j, l

---

Suppose you have two training examples $(x^{(1)}, y^{(1)})(x(1), y(1))$ and $(x^{(2)}, y^{(2)})(x(2), y(2))$. Which of the following is a correct sequence of operations for computing the gradient? (Below, FP = forward propagation, BP = back propagation).

FP using $x^{(1)}x(1)$ followed by FP using $x^{(2)}x(2)$. Then BP using $y^{(1)}y(1)$ followed by BP using $y^{(2)}y(2)$.

FP using $x^{(1)}x(1)$ followed by BP using $y^{(2)}y(2)$. Then FP using $x^{(2)}x(2)$ followed by BP using $y^{(1)}y(1)$.

BP using $y^{(1)}y(1)$ followed by FP using $x^{(1)}x(1)$. Then BP using $y^{(2)}y(2)$ followed by FP using $x^{(2)}x(2)$.

FP using $x^{(1)}x(1)$ followed by BP using $y^{(1)}y(1)$. Then FP using $x^{(2)}x(2)$ followed by BP using $y^{(2)}y(2)$: **True**

Suppose D1 is a 10x6 matrix and D2 is a 1x11 matrix. You set:DVec = [D1(:); D2(:)];Which of the following would get D2 back from DVec?

reshape(DVec(60:71), 1, 11)

reshape(DVec(61:72), 1, 11)

reshape(DVec(61:71), 1, 11)

Correct

reshape(DVec(60:70), 11, 1)

Let $J(\theta) = \theta^3$. Furthermore, let $\theta = 1$ and $\epsilon = 0.01$. You use the formula: $\frac{J(\theta+\epsilon)-J(\theta-\epsilon)}{2\epsilon}$ to approximate the derivative. What value do you get using this approximation? (When $\theta = 1$, the true, exact derivative is $\frac{d}{d\theta}J(\theta) = 3$.

3.0000

3.0001

Correct

3.0301

6.0002

What is the main reason that we use the backpropagation algorithm rather than the numerical gradient computation method during learning?

The numerical gradient computation method is much harder to implement.

The numerical gradient algorithm is very slow.

Correct

Backpropagation does not require setting the parameter EPSILON.

None of the above.

Consider this procedure for initializing the parameters of a neural network:

1. Pick a random number r = rand(1,1) * (2 * INIT_EPSILON) - INIT_EPSILON;
2. Set $\Theta_{ij}^{(l)}$ = r for all i, j, l

Does this work?

Yes, because the parameters are chosen randomly.

Yes, unless we are unlucky and get r=0 (up to numerical precision).

Maybe, depending on the training set inputs x(i).

No, because this fails to break symmetry.

Correct

Suppose you are using gradient descent together with backpropagation to try to minimize $J(\Theta)$ as a function of Θ. Which of the following would be a useful step for verifying that the learning algorithm is running correctly?

Plot $J(\Theta)$ as a function of Θ, to make sure gradient descent is going downhill.

Plot $J(\Theta)$ as a function of the number of iterations and make sure it is increasing (or at least non-decreasing) with every iteration.

Plot $J(\Theta)$ as a function of the number of iterations and make sure it is decreasing (or at least non-increasing) with every iteration.

Correct

Plot $J(\Theta)$ as a function of the number of iterations to make sure the parameter values are improving in classification accuracy.

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, ji,j. Which of the following is a correct vectorization of this step?

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

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

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

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

=> 1

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{dJ(\theta)}{d\theta} = 12 .)

11.9988

12

12.0012

6

=> 3

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

For computational efficiency, after we have performed gradient checking toverify 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.

=> 1 and 3

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

Suppose you have a three layer network with parameters \Theta^{(1)}Θ(1) (controlling the function mapping from the inputs to the hidden units) and \Theta^{(2)}Θ(2) (controlling the mapping from the hidden units to the outputs). If we set all the elements of \Theta^{(1)}Θ(1) to be 0, and all the elements of \Theta^{(2)}Θ(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).

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)J(Θ) as a function of the number of iterations, and make sure it is decreasing (or at least non-increasing) after each iteration.

If we initialize all the parameters of a neural network to ones instead of zeros, this will suffice for the purpose of "symmetry breaking" because the parameters are no longer symmetrically equal to zero.

=> 1 and 3