1 ML Week 5: Neural Network Learning

Overfitting and Regularization

- If one neural network overfits the training set, one reasonable step is to increase the regularization parameter λ.
- For computational efficiency, after we performed gradient checking to verify that our back-propagation code is correct, we usually disable gradient checkking before using back-propagation to train the network

2 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?

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

Question 2.

Suppose Theta1 is a 5×3 matrix, and Theta2 is a 4×6 matrix. You set `thetaVec=[Theta1(:);Theta2(:)]`.

Which of the following correctly recovers Theta2?

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

Question 3.

Let $J(\theta) = 3\theta^3 + 2$.

Let $\theta = 1$, and $\epsilon = 0.01$.

Use the formula $J(\theta + \epsilon) - J(\theta - \epsilon)2\epsilon$ to numerically compute an approximation to the derivative at $\theta = 1$.

What value do you get? (When $\theta = 1$, the true/exact derivative is $\frac{dJ(\theta)}{d\theta} = 9$.)

- 9
- 8.9997
- 11
- 9.0003

Exercise

Let $J(\theta) = 3\theta^2 + 2$

Let $\theta = 1$ and $\epsilon = 0.01$

Use the formula to numerically compute an approximation to the derivative of θ at $theta = 1$

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

$$= \frac{(3(1.01)^2 + 2) - (3(0.99)^2 + 2)}{0.002} = 9.003$$

Question 4.

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

- WRONG 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.
- 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.
- 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.

- Using gradient checking can help verify if one's implementation of backpropagation is bug-free.
- 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.
- If our neural network overfits the training set, one reasonable step to take is to increase the regularization parameter λ.
- 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).

Question 5.

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

- 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.
- CORRECT 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 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.
- 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.
- WRONG 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 $\theta(1)$ 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.
- WRONG 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.