|  |
| --- |
| Cost Function For Neural Network |
| h(x) & y = m \* k matrix   1. *sum over the difference in rows* 2. *then sum over columns)*   theta = (#layers-1) \* (#hypothesis \* # feature)   1. sum thetas of hypothesis into 1 hypothesis 2. sum all features into 1 3. sum theses number for theta between different layer |

**-----**

**m**

|  |
| --- |
| **Derivative term of neural network – Backpropagation algorithm** |
| **Intuition** |
| **Compute error for each output unit**  L = 4    **Use the error term of the last layer to compute the error in the earlier layers**    Note:  g’(3) = derivative of activation of function g by input value z(3)  g’(3) = a(3) .\* (1-a(3))  no delta(1) because x = input |
| **Backpropagation algorithm** |
| == |

**Unrolling matrices of theta into vectors**

|  |
| --- |
| **Recall** |
| These 2 functions only accept parameters(gradient and initialTheta) in vector |
| **Unrolling** |
| Assume the neural network looks like: 10elements 🡪 10 elements 🡪 1 elements    Put all element of matrices into one big long vector    Reshape the matrix according to the size of each layer  https://d3c33hcgiwev3.cloudfront.net/imageAssetProxy.v1/kdK7ubT2EeajLxLfjQiSjg_d35545b8d6b6940e8577b5a8d75c8657_Screenshot-2016-11-27-15.09.24.png?expiry=1535673600000&hmac=4BD8yRoaAoHp0xjYhyv9w-mT1a5vpRoIjm-WyLKxS9M  Vectorized matrix in costFunction |

**Gradient Checking**

|  |
| --- |
| **Purpose** |
| eliminate bugs in backpropagation/forward propagation |
| **Numerical estimation** |
| Where epsilon = 10 ^ -4 |
| **Parameter in numerical estimation** |
|  |
| **Code** |
|  |

**Random Initialization**

|  |
| --- |
| **Purpose** |
| Setting initial theta to all zero does not work for neural network, because the gradient descent will be computed to same value. |
| **Solution** |
|  |

Step:

1. initialize random theta
2. implement forward propagation to get h(x) for any x(i)
3. implement code to compute cost function J(teta)
4. implement backprop to compute partial derivatives of the cost function (store activations & delta for each layer)

for i = 1:m,

Perform forward propagation and backpropagation using example (x(i),y(i))

(Get activations a(l) and delta terms d(l) for l = 2,...,L

Get cap-delta with delta and activation

Compute partial derivative from cap-delta

1. use gradient checking to compare partial derivative make sure the partial derivative is computing the right number
2. use gradient descent with the computed partial derivative and costFunction to minimize J(theta) with theta