**Saisha Kashyap - Progress Report**

**25.12.18**

**Machine Learning**

Completed week 5 of the Machine Learning course on Coursera

**Neural Networks**

Binary classification: y=0 or 1 and number of units in layer L is 1

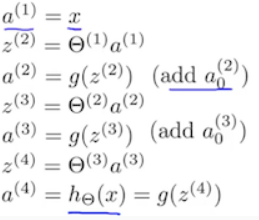
Multiclass classification: y is a vector and number of units in layer L is equal to the number of output units

**Cost function:**

There are additional nested summations that loop through the number of output nodes. In the regularisation part, we account for multiple theta matrices. The number of columns in the current theta matrix is equal to the number of nodes in the current layer (including the bias unit). The number of rows is equal to the number of nodes in the next layer (excluding the bias unit).

**Forward & Backward Computation:**

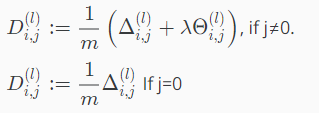
**Forward:**

****

**Backward:** Used to find error values and thus, partial derivative of the cost function







The error values (del) for the last layer are the differences of the actual results in the last layer and the correct output y.

The delta values of layer L are calculated by multiplying the delta values in the next layer with the theta matrix of layer L. It is then element-wise multiplied with g’ i.e. a.\*(1-a) which is the derivative if the activation function g.

The capital-delta matrix (D) is used as an accumulator to add up the values as we go along and eventually compute the partial derivative term.

**Unrolling & Reshaping:**

The theta values can be accumulated as a vector as follows:

thetaVector[theta1(:);theta(:);theta3(:)]

The individual theta values theta1, theta2 and theta3 can be recovered with the help of reshape

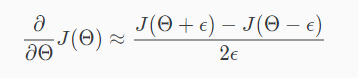
Theta1=reshape(thetaVector(1:110),10,11)

Theta2=reshape(thetaVector(111:220),10,11)

Theta3=reshape(thetaVector(221:231),1,11)

**Gradient checking:**

Gradient checking is done in order to cross-check the values computed through back propagation. The following formula is used:



The code to compute gradApprox is extremely slow and thus, should be turned off once the value obtained through back propagation is checked.

Also, the value of theta is randomly initialised in order to prevent redundancy by breaking symmetry. This is done with the help of rand function which initiates a matrix of random real numbers between 0 and 1.

**Neural networks algorithm stepwise:**

1. Picking a network architecture
2. Randomly initializing the weights
3. Implementing forward propagation to get h(x) for any x
4. Implementing the cost function
5. Implementing backward propagation to compute partial derivatives
6. Gradient checking to confirm and then disable
7. Gradient descent/ optimisation function to minimise the cost function with the weights in theta

**Python**

Completed course 2 of 5 in the Python for Everybody series offered by University of Michigan.