# Implementation of Neural Network

Saturday, June 30, 2018
9:49 PM

We'll try to implement a L-layer neural network. The implementation could be divided into 5 steps:

0. First things first
   Layers_dims
       [12288, 20, 7, 5, 1]
   Use a dictionary to record parameters(W[l], b[l]) of DNN
       parameters = {"W1": W1, "b1": b1, "W2": W2, "b2": b2}
   Shape of parameter matrixes: ( Z[i] = W[i]*X[i] + b[i] )

| i | W[i] | X[i] = Z[i-1] | b[i] | Z[i] | |
|---|---|---|---|---|---|
| 1 | 20, 12888 | 12888, 209 | 20, 1 | 20, 209 | |
| 2 | 7, 20 | 20, 209 | 7, 1 | 7, 209 | |
| 3 | 5, 7 | 7, 209 | 5, 1 | 5, 209 | |
| 4 | 1, 5 | 5, 209 | 1, 1 | 1, 209 | |

| Functions | Usage | Input | Out |
|---|---|---|---|
| **initialize_parameters_deep** | Initialize parameter dictionary of DNN<br>(Refer to Initialization in course2) | Num of Layers | parameters{W[i], b[i]} |
| **L_model_forward** | Forward Propagation<br>For each layer,<br>Z[i]=W[i] • A[i]+b[i] | X, parameters{W[i], b[i]} | AL (The output layer)<br>cache(A_prev, W, b) |
| **compute_cost** | Compute cost | AL, Y | cost |
| **L_model_backward** | Backward Propagation<br>For each layer(in reversed order),<br>compute gradient dA, dW, db | AL, Y, cache(A_prev, W, b) | Grads{dA[i], dW[i], db[i]} |
| **update_parameters** | For each layer,<br>parameters[W[i]] = parameters[W[i]] - learning_rate * grads[dW[i]]<br>parameters[b[i]] = parameters[b[i]] - learning_rate * grads[db[i]] | parameters{W[i], b[i]},<br>Grads{dA[i], dW[i], db[i]}, learning_rate | parameters{W[i], b[i]}<br>(new value) |
| | | | |
| linear_activation_forward | Z, linear_cache = linear_forward(A_prev, W, b)<br>A, activation_cache = Activation_function(Z) | A_prev, W, b, activation | A, cache(linear_cache, activation_cache) |
| linear_forward | Z = W • A + b | A, W, b | Z, cache(A,W,b) |
| linear_activation_backward | Calculate dW, db for current layer | dA, cache(linear_cache, activation_cache), activation | dA_prev, dW, db |
| linear_backward | Calculate dW, db | dZ, cache(A_prev, W, b) | dA_prev, dW, db |

*W, b* indicates the *parameters['WL'], parameters['bL']* for a specific layer L

1. Load dataset

| | Shape | comment |
|---|---|---|
| Train_x_org | (209, 64, 64, 3) | 209 original 64x64-pixel colored pictures |
| Train_x | (12288, 209) | Train_x_org is reversed and flattened (12288 = 64x64x3 ). The values are standardized to range [0,1] |
| Train_y | (1, 209) | Train_y is reversed |

2. Initialize parameters
   Here we will use Random Initialization.

3. Train NN

   For each iteration:
       L_model_forward
       Compute_cost
       L_model_backward
       Update_parameters

4. Predict