Training the Network

Now that we have completed the parts to 1) initialize the network, 2) perform the forward pass, and 3) the backward pass, we are ready to train the network for an image classification task. As described in the first part of this assignment, we will train a network that can classify a given image into 20 different object classes such as "airplane", "bicycle","car", "sheep", etc.

You will be given a set of 336 dimensional visual features describing every image in the training and testing datasets. These features will be used as an input to the first layer of a neural network. For this lab, we will use a two layer network: 1) the first layer will contain 100 hidden units, and 2) the second layer will contain 20 output units, each of them corresponding to a probability for one of the 20 object classes.

Below is the image with a general pseudo code describing a general procedure how to train a neural network.

Algorithm 1 Training a Neural Network

```
1: Initialization: Randomly initialize the parameters of a network
2: for each training epoch do
3:    for each batch in the training dataset do
4:       Perform a forward pass
5:       Perform a backward pass
6:       Update the parameters in each layer
7:    end for
8: end for
```

Note that the training is done via a batch mode. Each batch represents a subset of data points from the original training dataset. This is a common way to train neural networks to make the training more efficient and also to reduce the memory storage required for training. The procedure for sampling data points and including them in a batch is provided in the code template so you won't need to worry about it.

Your task will be to write the code for lines 4-6 in the pseudo-code description above. For lines 4-5, you will use the matlab functions `ForwardPass`, and `BackwardPass` that we implemented before. Simply call those function with the appropriate input arguments. For line 6, you will have to loop through every layer of the network (for layers that have learnable parameters), access the gradients $\frac{\partial L}{\partial W^{(l)}}$ that were computed in the `BackwardPass` function, and update the parameters using the gradient descent update rule:

$$W^{(l)} = W^{(l)} - \gamma \frac{\partial L}{\partial W^{(l)}}$$

where $\gamma = 2$ is the learning rate. Your network should achieve above 90% accuracy on the training dataset.

Your Function

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

```
function nn=TrainNetwork(train_x,train_y)
    % Train the network
    %
    % Input:
    % - train_x: n x d matrix storing n data observations with d features
    % - train_y: a ground truth indicator matrix, where y(i,j)=1 indicates that a data point i belongs to an ob
    % Output:
    % - nn: a variable storing the neural network structure.

    %% Network Initialization
    nn = InitializeNetwork([336 100 20]);
    n_layers=numel(nn.W)+1;


    %% Hyperparameters
    num_epochs=50;
    batch_size=100;
    num_batches=size(train_x,1)/batch_size;
    gamma=2;

    for i = 1 : num_epochs
```

Code to call your function

Reset

```
load('X.mat');
load('Y.mat');
X=StandardizeData(X);
[train_x, train_y, test_x, test_y] = splitData(X, Y);
nn=TrainNetwork(train_x,train_y);
```

▶ Run Function

Previous Assessment: All Tests Passed

Submit

Does the Trained Network Achieve >90% Training Accuracy?