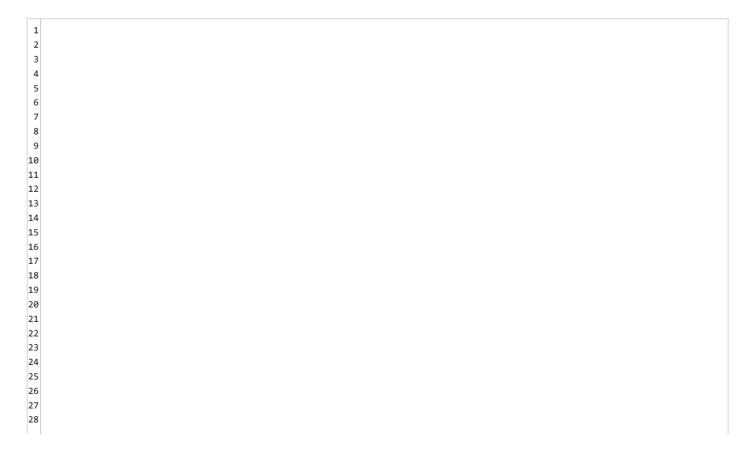
Backward Pass

In this part of the lab, we will implement a neural net's capability to propagate the error gradients from the last layer all the way to the first layer. We will refer to this procedure as a backward pass inside a neural network. At every fully connected layer l, we will need to do four things:

- 1. Compute the derivative of a non-linear sigmoid function $f(z) = \frac{1}{1 + \exp(-z)}$, which can be computed as $\frac{\partial f(z)}{\partial z} = f(z)(1-f(z))$. Note that your implementation needs to handle multi-dimensional vector inputs (i.e. use '.*' operator in matlab instead of '*'). The computed result $\frac{\partial f(z)}{\partial z}$ corresponding to layer l should be stored into nn.gradSigm{I} variable.
- 2. Use the error gradients $\frac{\partial L}{\partial z^{(l+1)}}$ coming from layer l+1 to compute the error gradients $\frac{\partial L}{\partial f(z^{(l)})}$ at the current layer l. This can be done as: $\frac{\partial L}{\partial f(z^{(l)})} = (W^{(l)})^T \frac{\partial L}{\partial z^{(l+1)}}$ where $(W^{(l)})^T$ depicts the transposed parameter matrix from layer l. The computed result $\frac{\partial L}{\partial f(z^{(l)})}$ should be stored into nn.gradA{I} variable.
- 3. Use the previously computed quantities $\frac{\partial L}{\partial f(z^{(l)})}$ and $\frac{\partial f(z)}{\partial z}$ to compute $\frac{\partial L}{\partial z^{(l)}} = \frac{\partial L}{\partial f(z^{(l)})} \odot \frac{\partial f(z^{(l)})}{\partial z^{(l)}}$ where \odot is an elementwise multiplication implemented as '.*' in matlab. The computed quantity $\frac{\partial L}{\partial z^{(l)}}$ should be stored into the variable nn.gradZ{I} for every laver l in the network.
- 4. Finally, we need to compute the error gradients with respect to the fully connected layer parameters: $\frac{\partial L}{\partial W^{(l)}}$ for every layer l. This can be done as: $\frac{\partial L}{\partial W^{(l)}} = \frac{\partial L}{\partial z^{(l+1)}} (a^{(l)})^T$ where $(a^{(l)})^T$ denotes the transposed activation units from layer l. Note that $a^{(l)} = f(z^l)$. Then, the computed result $\frac{\partial L}{\partial W^{(l)}}$ should be stored into the variable nn.gradW{I} for every layer l in the network.

Your Function

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



```
function nn = BackwardPass(nn,y)
       % perform the forward pass inside the network
31
32
       % Input:
33
       \ensuremath{\text{\%}} - nn: a structure storing the parameters of the network
34
       % - y: a ground truth indicator matrix, where y(i,j)=1 indicates that a data point i belongs to an object of
35
36
       % - nn: a new neural network variable where the values nn.gradZ{1} and nn.gradW{1} are updated for every la
37
38
39
       %number of layers in a network
40
       n_layers=numel(nn.W)+1;
41
42
       %computing the overall error of the network
43
       error= nn.a{n_layers} - y';
44
45
       %computing the error gradient in the last layer of the network
46
       nn.gradZ{n_layers} = error .* (nn.a{n_layers} .* (1 - nn.a{n_layers}));
47
48
       %looping through the layers backwards
```

Code to call your function

C Reset

```
1 architecture=[336 100 20];
2 nn = InitializeNetwork(architecture);
3 random_feature=rand(1,336);
4 random_label=rand(1,20);
5 nn = ForwardPass(nn, random_feature);
6 nn = BackwardPass(nn, random_label);
```

► Run Function

Previous Assessment: All Tests Passed

Submit

- Is the First Step of the Backward Pass Implemented Correctly?
- Is the Second Step of the Backward Pass Implemented Correctly?
- Is the Third Step of the Backward Pass Implemented Correctly?
- Is the Fourth Step of the Backward Pass Implemented Correctly?