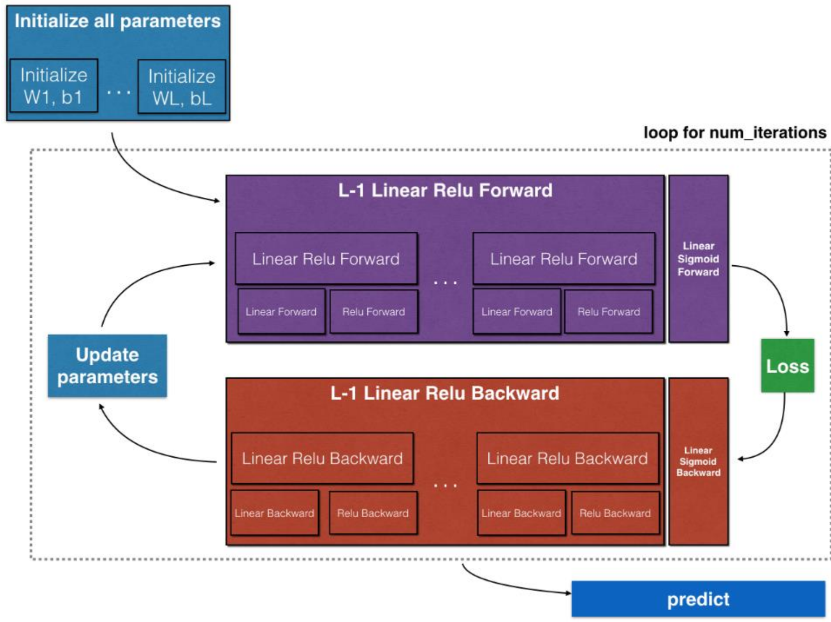
Building your deep neural network step by step



Initialization

2 helper functions will initialize the parameters for the model:

1) *initialize_parameters* – this function will be used to initialize parameters for a 2 – layer model.
2) *initialize_parameters_deep* - this function will be used to initialize parameters for a L – layer model.

Forward propagation module

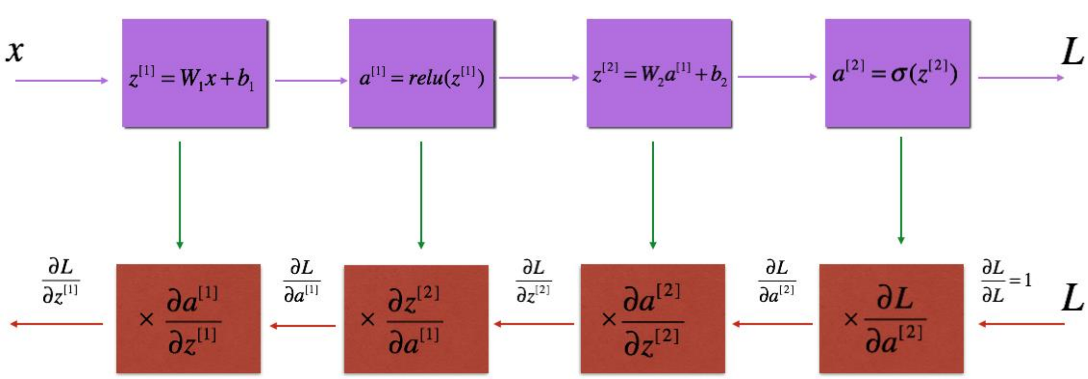
1) *linear_forward* – linear forward module (vectorized over all the examples). Computes the equation: $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$ (where $A^{[0]} = X$)
2) *linear_activation_forward* – uses the *linear_forward* function to compute $Z^{[l]}$, and computes $A^{[l]} = g^{[l]}(Z^{[l]})$. The activation function g can be ReLU or sigmoid (according to the function input).
3) *L_model_forward* – function that replicates the previous one (*linear_activation_forward* with ReLU) $[L-1]$ times, then follows that with one *linear_activation_forward* with sigmoid.

Cost function

compute_cost – Compute the cross-entropy cost J:

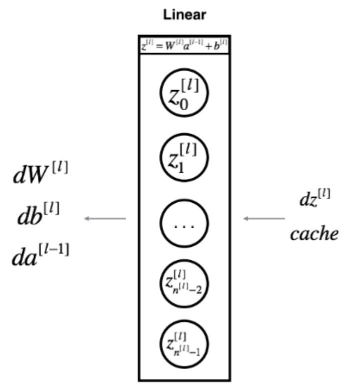
$$J = \frac{-1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + \left(1-y^{(i)}\right)\log\left(1-a^{[L](i)}\right)\right)$$

Backward propagation module



The purple blocks represent the forward propagation, and the red blocks represent the backward propagation

1) *linear_backward* – Implement the linear portion of backward propagation for a single layer l. For layer l the linear part is: $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$.
Suppose $dZ^{[l]} = \frac{\partial L}{\partial Z^{[l]}}$ is already calculated, we want to get $dW^{[l]}, db^{[l]}, dA^{[l-1]}$:



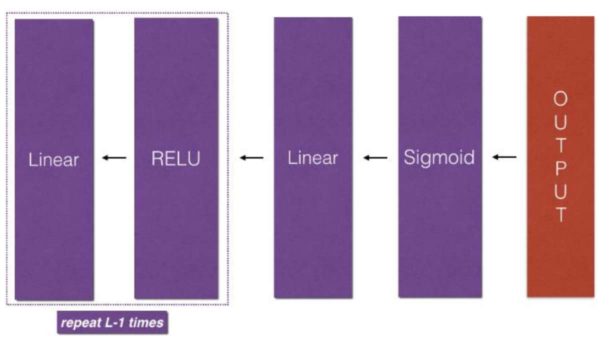
The 3 outputs $dW^{[l]}, db^{[l]}, dA^{[l-1]}$ are computed using the input $dZ^{[l]}$ according to the functions:

$$dW^{[l]} = \frac{\partial J}{\partial W^{[l]}} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$$

$$db^{[l]} = \frac{\partial J}{\partial b^{[l]}} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[l](i)}$$

$$dA^{[l-1]} = \frac{\partial L}{\partial A^{[l-1]}} = W^{[l]T} dZ^{[l]}$$

2) *linear_activation_backward* - uses the *linear_backward* function to compute $dZ^{[l]} = dA^{[l]} * g'\left(Z^{[l]}\right)$
3) *L_model_backward* – backward function for the whole network:



Initialize backpropagation: the output is $A^{[l]} = \sigma\left(Z^{[L]}\right)$, thus we need to compute $dAL = \frac{\partial L}{\partial A^{[L]}}$ by using the formula:

$$dAL = -(np.divide(Y, AL) - np.divide(1 - Y, 1 - AL))$$

derivative of cost with respect to $A^{[L]}$

Update parameters

update_parameters: Update the parameters of the model, using gradient descent:

$$W^{[l]} = W^{[l]} - \alpha dW^{[l]}$$

$$b^{[l]} = b^{[l]} - \alpha db^{[l]}$$

- α is the learning rate.