ANN SUMMER CAMP ASSIGNMENT-1

Atreya Goswami

May 7, 2020

1 Forward Propagation and Backward Propagation

Forward Propagation refers to the process of training our model in which the input fed is linearly combined with weights and bias and then passed through a non-linear activation function to compute the activations at every neuron of the network. When an input enters the system the weights ,bias and activation functions basically try to map it to make guesses close to the ground truth. Thus the task of forward propagation is to calculate the activations at every step till we arrive at the output. It broadly consists of 2 steps : Linear forward step and Activation forward step. In the linear step,we multiply the inputs of the neuron by the weights and add bias, which is then passed through a nonlinear function to calculate the activations for the neuron.

Back Propagation refers to the process of propagating the error calculated at the given step in a backward direction through the neural network i.e from the output layers towars the input layers. The error calculated in the log-loss function, or more generally the Cost Function is passed on to the previous units by using the chain rule of calculus. In this way error can be calculated for all our assumed parameters and they can then be updated in such a way so as to minimise the error using gradient descent. It also consists of 2 steps : the Activation Backward Step and the Linear Backward Step. In the activation backward step we use the error in the activation of the neuron to calculate the error in the linear output obtained during forward propagation using the derivative of the activation function. In the linear backward step we calculate the error in our weights and bias using the error calculated in the previous step. Point to be noted is that in both the processes we need some of the quantities calculated in the forward propagation steps to be stored as cache

2 Vectorization

Given : Input vector $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^4$

Number of layers in hidden unit ($n^{[1]}$) = 5

Output : $y \in \{0, 1\}$

Parameters:

Weights in the first hidden layer (layer 1) : $w_j^{[1]} = \begin{bmatrix} w_{j,0}^{[1]} \\ w_{j,1}^{[1]} \\ w_{j,2}^{[1]} \\ w_{j,3}^{[1]} \end{bmatrix} \in \mathbb{R}^4$

Bias in the first hidden layer : $b_j^{[1]} \in \mathbb{R}$,

where j represents the j^{th} unit of the hidden layer.

Weights in the output layer (layer 2) : $w_1^{[2]} = \begin{bmatrix} w_{1,0}^{[2]} \\ w_{1,1}^{[2]} \\ w_{1,2}^{[2]} \\ w_{1,3}^{[2]} \\ w_{1,4}^{[2]} \end{bmatrix} \in \mathbb{R}^5$

Bias in the output layer : $b^{[2]} \in \mathbb{R}$

2.1 Non-vectorized Implementation for 1 training example

2.1.1 Forward Propagation

Hidden layer :

For the j^{th} unit in the layer :

- $z_j^{[1]} = (w_j^{[1]})^T x + b^{[1]} = \sum_{k=0}^{3} (w_{j,k}^{[1]} x_k) + b^{[1]}$ (Linear forward step)
- $a_j^{[1]} = g^{[1]}(z_j^{[1]})$ (Activation forward Step)

- $z^{[1]} = \begin{bmatrix} z_0^{[1]} \\ \vdots \\ z_4^{[1]} \end{bmatrix} \in \mathbb{R}^5$
- $a^{[1]} = \begin{bmatrix} a_0^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix} \in \mathbb{R}^5$

Output Layer :

- $z_1^{[2]} = (w^{[2]})^T a^{[1]} + b^{[2]} = \sum_{j=0}^{4} (w_{1,j}^{[2]} a_j^{[1]}) + b^{[2]}$ (Linear Forward Step)
- $a_1^{[2]} = \sigma(z^{[2]})$ (Activation Forward Step)

$l(a^{[2]}, y) = -\{ylog(a^{[2]}) + (1-y)log(1-a^{[2]})\}$ (Log-Loss Function)

2.1.2 Back Propagation

Output Layer :

- $da^{[2]} = \frac{a^{[2]} - y}{a^{[2]}(1-a^{[2]})}$
- $dz^{[2]} = a^{[2]} - y$ (Activation Backward Step)
- $dw_1^{[2]} = dz^{[2]}(a^{[1]}) = dz^{[2]} \begin{pmatrix} a_0^{[1]} \\ a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{pmatrix}$ (Linear Backward Step)
- $db^{[2]} = dz^{[2]}$

Hidden Layer :

- $da^{[1]} = dz^{[2]}(w_1^{[2]}) = dz^{[2]} \begin{pmatrix} w_0^{[2]} \\ w_1^{[2]} \\ \vdots \\ w_4^{[2]} \end{pmatrix}$
- $dz^{[1]} = da^{[1]} * g^{[1]'}(z^{[1]})$ (Activation Backward Step)
- $dw_j^{[1]} = dz_j^{[1]}(x) = dz_j^{[1]} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$ (Linear Backward Step)
- $db_j^{[1]} = dz_j^{[1]}$

2.2 Vectorized Implementation for 1 training example :

Parameters :

Hidden Layer :

- $W^{[1]} = \begin{bmatrix} \cdots & (w_0^{[1]})^T & \cdots \\ \cdots & (w_1^{[1]})^T & \cdots \\ \cdots & (w_2^{[1]})^T & \cdots \\ \cdots & (w_3^{[1]})^T & \cdots \\ \cdots & (w_4^{[1]})^T & \cdots \end{bmatrix} \in \mathbb{R}^{5\times 4}$
- $b^{[1]} = \begin{bmatrix} b_0^{[1]} \\ b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} \in \mathbb{R}^5$

Output Layer :

- $W^{[1]} = \begin{bmatrix} \cdots & (w_1^{[2]})^T \cdots \end{bmatrix} \in \mathbb{R}^{1\times 5}$
- $b^{[2]} \in \mathbb{R}$

2.2.1 Forward Propagation

Hidden Layer :

- $z^{[1]} = W^{[1]}x + b^{[1]}$ (Linear Forward Step)
- $a^{[1]} = g^{[1]}(z^{[1]})$ (Activation forward Step)

Output Layer :

- $z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$ (Linear Forward Step)
- $a^{[2]} = \sigma(z^{[2]})$ (Activation Forward Step)

2.2.2 Back Propagation

Notation : $dX = \nabla_X l(a^{[2]}, y)$

Output Layer :

- $da^{[2]} = \frac{a^{[2]}-y}{a^{[2]}(1-a^{[2]})}$
- $dz^{[2]} = a^{[2]} - y$ (Activation Backward Step)
- $dW^{[2]} = dz^{[2]}(a^{[1]})^T$ (Linear Backward Step)

- $db^{[2]} = dz^{[2]}$

Hidden Layer :

- $da^{[1]} = dz^{[2]}(dW^{[2]})^T$
- $dz^{[1]} = da^{[1]} * g^{[1]'}(z^{[1]})$ (Activation Backward Step)
- $dW^{[1]} = dz^{[1]}(x)^T$ (Linear Backward Step)
- $db^{[1]} = dz^{[1]}$

3 Activation Functions

- Sigmoid :

  $\sigma(z) = \frac{1}{1+e^{-z}} \; \forall \, z \in \mathbb{R}$

  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

- ReLu ($f(z)$):

  $f(z) = max(0, z) \; \forall \, z \in \mathbb{R}$

  $f'(z) = \begin{pmatrix} 0 & z \leq 0 \\ 1 & z > 0 \end{pmatrix}$

- Leaky ReLu ($h(z)$):

  $h(z) = max(az, z) \;\; \forall \, z \; \in \mathbb{R}$ where $a << 1$

  $h'(z) = \begin{pmatrix} a & z \leq 0 \\ 1 & z > 0 \end{pmatrix}$

- Tanh :

  $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

  $tanh'(z) = 1 - (tanh(z))^2$

- Softmax :

  $S_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$

  We compute $D_j S_i$ where i and j are arbitrary.

  $D_j S_i = \frac{\partial S_i}{\partial y_j} = \frac{\partial}{\partial y_j}\{\frac{e^{y_i}}{\sum_k e^{y_k}}\}$

  $D_j S_i = \begin{pmatrix} S_i(1 - S_i) & i = j \\ -S_i S_j & i \neq j \end{pmatrix}$