REPORT

Assignment 2:

Learning a simple deep forward network

MR.CHALERMKIAT CHANAHCHAN

ID: 62070701602

King Mongkut's University of Technology Thonburi

Faculty of Engineering, Department of Computer Engineering

CPE 663 Special Topic: Deep Learning, 1/2019

Assignment 2: Learning a simple deep forward network

Due: 17 October 2019. Please submit your report in PDF to LEB2

Introduction

In this assignment, we will implement a backpropagation algorithm to learn a simple deep forward network for XOR function. Unlike the lecture which the XOR network uses sigmoid as the activation function. In this assignment we will use ReLU for the hidden units and sigmoid for the output unit.

These are the architectural specification:

Connection type: Fully connected

Number of input units: 2

Number of hidden layers: 1

Number of units in hidden layer: 2

Activation function of hidden layer: ReLU

Number of output unit: 1

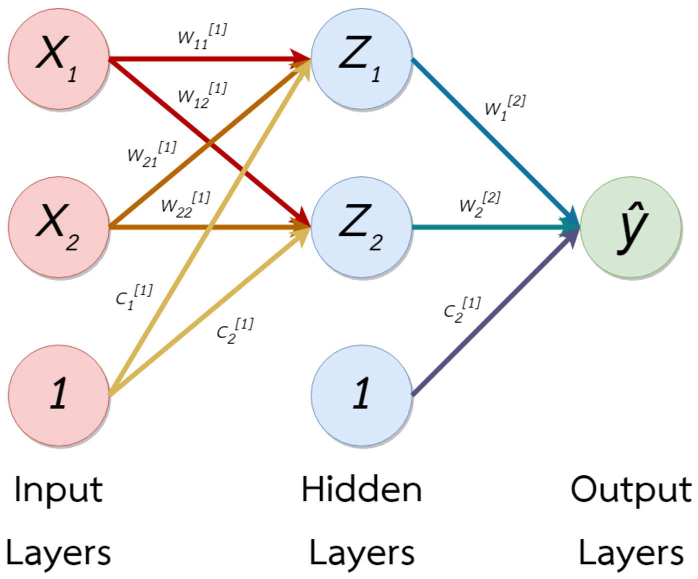
Activation function of output layer: Sigmoid

Tasks

1. Draw the diagram showing this deep forward network.
2. Write the equation of each units, showing all of the components in the network
3. Write the error function.
4. Derive the gradient of weights in output layers. Show your work.
5. Derive the gradient of weights in hidden layers. Show your work.
6. Prepare the XOR data point.
7. Implement the gradient descent using all of the dataset. The training should be done instance by instance and repeat for a number of iterations. Weights should be initialized randomly or fixed to small values.
8. Show how the error changes by iterations.
9. Demonstrate that the network has finally learned the XOR function.

Hint: Use matrix operation to simplify the implementation

Task 1: Draw diagram showing this deep forward network.



The Deep Forward Network Diagram

X = the input features

Z = Hidden Function

$\hat{y}$ = Predict Output

W = Weight

C = Bias

Task 2: Write the equation of each units, showing all of the components in the network

Equation of Hidden Layers or First Layers:

Regression Equation:

$$Z^{[1]} = W^{[1]}X + C^{[1]}$$

Equations of each hidden layers:

Node 1: $Z_1^{[1]} = W_{11}^{[1]}X_1 + W_{21}^{[1]}X_2 + C_1^{[1]}$

Node 2: $Z_2^{[1]} = W_{12}^{[1]}X_1 + W_{22}^{[1]}X_2 + C_2^{[1]}$

Activation Function of hidden layer: ReLU (Rectifier Linear Unit)

ReLU Function: $y = ReLU(Z)$

$y = max\{0, Z\}$

Adding $Z^{[1]}$ in Activation Function

$$A^{[1]} = ReLU(Z^{[1]})$$

$$A^{[1]} = max\{0, (Z^{[1]})\}$$

$$A^{[1]} = max\{0, (W^{[1]}X + C^{[1]})\}$$

Equations of each activation function of hidden layer:

Node 1: $A_1^{[1]} = ReLU\left(Z_1^{[1]}\right)$

$$A_1^{[1]} = ReLU\left(W_{11}^{[1]}X_1 + W_{21}^{[1]}X_2 + C_1^{[1]}\right)$$

$$A_1^{[1]} = max\left\{0, \left(W_{11}^{[1]}X_1 + W_{21}^{[1]}X_2 + C_1^{[1]}\right)\right\}$$

Node 2: $A_2^{[1]} = ReLU\left(Z_2^{[1]}\right)$

$$A_2^{[1]} = ReLU\left(W_{12}^{[1]}X_1 + W_{22}^{[1]}X_2 + C_2^{[1]}\right)$$

$$A_2^{[1]} = max\left\{0, \left(W_{12}^{[1]}X_1 + W_{22}^{[1]}X_2 + C_2^{[1]}\right)\right\}$$

Equation of Output Layer:

Regression Equation:

$$Z^{[2]} = W^{[2]}A^{[1]} + C^{[2]}$$

$$Z^{[2]} = W_1^{[2]}A_1^{[1]} + W_2^{[2]}A_2^{[1]} + C^{[2]}$$

Activation Function of Output Layer: Sigmoid Function

Sigmoid Function: $y = \sigma(Z)$

$$y = \frac{1}{1+e^{-(Z)}}$$

Adding $Z^{[2]}$ in Activation Function

$$A^{[2]} = \sigma\left(Z^{[2]}\right)$$

$$A^{[2]} = \sigma\left(W^{[2]}A^{[1]} + C^{[2]}\right)$$

$$A^{[2]} = \sigma\left(W^{[2]}(max\{0, \left(Z^{[1]}\right)\}) + C^{[2]}\right)$$

$$A^{[2]} = \frac{1}{1 + e^{-\left(W^{[2]}(max\{0,(Z^{[1]})\}) + C^{[2]}\right)}}$$

$$A^{[2]} = \frac{1}{1 + e^{-\left(W^{[2]}(max\{0,(W^{[1]}X + C^{[1]})\}) + C^{[2]}\right)}}$$

Equations activation function of output layer: ($\hat{y}$ = predict output)

$$A^{[2]} = \frac{1}{1 + e^{-\left(W^{[2]}(max\{0,(W^{[1]}X + C^{[1]})\}) + C^{[2]}\right)}}$$

And Predict Output ($\hat{y}$) is

$$\hat{y} = A^{[2]} = \frac{1}{1 + e^{-\left(W^{[2]}(max\left\{0,\left(W^{[1]}X + C^{[1]}\right)\right\}) + C^{[2]}\right)}}$$

Task 3: Write the error function.

Loss Function Is used in this assignment: Cross-Entropy or Log Loss

$$Loss(\hat{\boldsymbol{y}}, \mathrm{y}) = -\boldsymbol{y}\log(\hat{\boldsymbol{y}}) + (1-\boldsymbol{y})\log(1-\hat{\boldsymbol{y}})$$

Determined when y ∈ {0,1}

$$Loss(\hat{\boldsymbol{y}}, \mathrm{y}) = \begin{cases} -log(\hat{\boldsymbol{y}}) \text{ if } \mathrm{y} = 1 \\ -log(1-\hat{\boldsymbol{y}}) \text{ if } \mathrm{y} = 0 \end{cases}$$

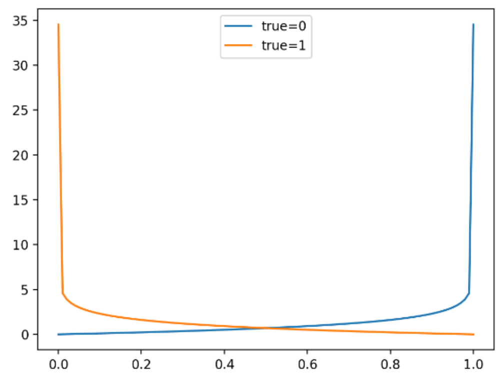
When predict

if $\boldsymbol{y} = 1$ and $\hat{\boldsymbol{y}} = 1$, then $Loss(\hat{\boldsymbol{y}}, \mathrm{y}) = 0$ (Loss Value will converge to 0)

if $\boldsymbol{y} = 1$ and $\hat{\boldsymbol{y}} \rightarrow 0$, then $Loss(\hat{\boldsymbol{y}}, \mathrm{y}) \rightarrow \infty$ (Loss Value will converge to infinite)

if $\boldsymbol{y} = 0$ and $\hat{\boldsymbol{y}} = 0$, then $Loss(\hat{\boldsymbol{y}}, \mathrm{y}) = 0$ (Loss Value will converge to 0)

if $\boldsymbol{y} = 0$ and $\hat{\boldsymbol{y}} \rightarrow 1$, then $Loss(\hat{\boldsymbol{y}}, \mathrm{y}) \rightarrow \infty$ (Loss Value will converge to 0)



From https://machinelearningmastery.com/how-to-score-probability-predictions-in-python/

$Loss(\hat{\boldsymbol{y}}, \mathrm{y}) = 0$, The output and predict output are almost correct.

$Loss(\hat{\boldsymbol{y}}, \mathrm{y}) \rightarrow \infty$, The output and predict output are wrong.

Then can be written as,

$$Loss(\hat{\boldsymbol{y}}, \mathrm{y}) = -\frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{y}\log(\hat{\boldsymbol{y}}) + (1-\boldsymbol{y})\log(1-\hat{\boldsymbol{y}}))$$

Task 4: Derive the gradient of weights in output layers. Show your work.

We find the gradient of weights in output layers from derivative of loss function ($\frac{\partial}{\partial W^{[2]}}(Loss)$) using chain rule method:

$$\frac{\partial}{\partial W^{[2]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial W^{[2]}}(Z^{[2]})$$

Loss function from task 3,

$$Loss = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

Find $\frac{\partial}{\partial \hat{y}}(Loss)$:

$$\frac{\partial}{\partial \hat{\boldsymbol{y}}}(Loss) = \frac{\partial}{\partial \hat{\boldsymbol{y}}}\left(-(\boldsymbol{y}\log(\hat{\boldsymbol{y}}) + (1-\boldsymbol{y})\log(1-\hat{\boldsymbol{y}}))\right)$$

$$\frac{\partial}{\partial \hat{\boldsymbol{y}}}(Loss) = -\left[\frac{\partial}{\partial \hat{\boldsymbol{y}}}(\boldsymbol{y}\log(\hat{\boldsymbol{y}})) + \frac{\partial}{\partial \hat{\boldsymbol{y}}}((1-\boldsymbol{y})\log(1-\hat{\boldsymbol{y}}))\right]$$

$$\frac{\partial}{\partial \hat{\boldsymbol{y}}}(Loss) = -\left[\boldsymbol{y}\frac{\partial}{\partial \hat{\boldsymbol{y}}}(\log(\hat{\boldsymbol{y}})) + (1-\boldsymbol{y})\frac{\partial}{\partial \hat{\boldsymbol{y}}}(\log(1-\hat{\boldsymbol{y}}))\right]$$

Derivative of $log(z)$:

$$\frac{\partial}{\partial z}log(z) = \frac{1}{z}$$

and adding in equation,

$$\frac{\partial}{\partial \hat{y}}(Loss) = -\left[\frac{y}{\hat{y}}\frac{\partial}{\partial \hat{y}}(\hat{y}) + \frac{(1-y)}{(1-\hat{y})}\frac{\partial}{\partial \hat{y}}(1-\hat{y})\right]$$

$$\frac{\partial}{\partial \hat{y}}(Loss) = -\left[\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right]$$

From Sigmoid Function:

$$Sigmoid(z) = \sigma(z) = \frac{1}{1+e^{-z}}$$

Derivative of Sigmoid Function

$$\frac{\partial}{\partial z}\sigma(z) = \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right)$$

$$\frac{\partial}{\partial z}\sigma(z) = -1(1+e^{-z})^{-2}\frac{\partial}{\partial z}(1+e^{-z})$$

$$\frac{\partial}{\partial z}\sigma(z) = -1(1+e^{-z})^{-2}(\frac{\partial}{\partial z}(1) + \frac{\partial}{\partial z}(e^{-z}))$$

$$\frac{\partial}{\partial z}\sigma(z) = -1(1+e^{-z})^{-2}(0 + (-e^{-z}))$$

$$\frac{\partial}{\partial z}\sigma(z) = -\frac{1}{(1+e^{-z})^2}(-e^{-z})$$

$$\frac{\partial}{\partial z}\sigma(z) = -\left(\frac{1}{1+e^{-z}}\right)\left(\frac{-e^{-z}}{1+e^{-z}}\right)$$

$$\frac{\partial}{\partial z}\sigma(z) = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}+1-1}{1+e^{-z}}\right)$$

$$\frac{\partial}{\partial z}\sigma(z) = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}+1}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right)$$

$$\frac{\partial}{\partial z}\sigma(z) = \big(\sigma(z)\big)\big(1 - \sigma(z)\big)$$

Then we derivative $\hat{y}$ and $Z^{[2]}$,

Find $\frac{\partial}{\partial Z^{[2]}}(\hat{y})$:

From task 2, $\hat{y} = A^{[2]} = \sigma\big(Z^{[2]}\big)$

$$\frac{\partial}{\partial Z^{[2]}}(\hat{y}) = \frac{\partial}{\partial Z^{[2]}}(\sigma\big(Z^{[2]}\big))$$

And derivative of Sigmoid Function, $\frac{\partial}{\partial z}\sigma(z) = \big(\sigma(z)\big)\big(1 - \sigma(z)\big)$, adding in equation:

$$\frac{\partial}{\partial Z^{[2]}}(\hat{y}) = \left(\sigma\big(Z^{[2]}\big)\right)\left(1 - \sigma\big(Z^{[2]}\big)\right)$$

And $\hat{y} = A^{[2]} = \sigma\big(Z^{[2]}\big)$

$$\frac{\partial}{\partial Z^{[2]}}(\hat{y}) = (\hat{y})(1-\hat{y})$$

Find the final part of chain rule equation ($\frac{\partial}{\partial w^{[2]}}\big(Z^{[2]}\big)$)

Find $\frac{\partial}{\partial w^{[2]}}\big(Z^{[2]}\big)$:

From task 2, $Z^{[2]} = W^{[2]}A^{[1]} + C^{[2]}$

$$\frac{\partial}{\partial W^{[2]}}\big(Z^{[2]}\big) = \frac{\partial}{\partial W^{[2]}}(W^{[2]}A^{[1]} + C^{[2]})$$

$$\frac{\partial}{\partial W^{[2]}}\big(Z^{[2]}\big) = \frac{\partial}{\partial W^{[2]}}(W^{[2]}A^{[1]}) + \frac{\partial}{\partial W^{[2]}}(C^{[2]})$$

$$\frac{\partial}{\partial W^{[2]}}\big(Z^{[2]}\big) = A^{[1]}\frac{\partial}{\partial W^{[2]}}(W^{[2]}) + 0$$

$$\frac{\partial}{\partial W^{[2]}}\big(Z^{[2]}\big) = A^{[1]}(1) = A^{[1]}$$

And adding 3 part ($\frac{\partial}{\partial \hat{y}}(Loss), \frac{\partial}{\partial Z^{[2]}}(\hat{y}), \frac{\partial}{\partial W^{[2]}}(Z^{[2]})$) to equation:

$$\frac{\partial}{\partial W^{[2]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial W^{[2]}}(Z^{[2]})$$

$$\frac{\partial}{\partial W^{[2]}}(Loss) = -\left[\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right](\hat{y})(1-\hat{y})(A^{[1]})$$

$$\frac{\partial}{\partial W^{[2]}}(Loss) = \left[-\frac{\boldsymbol{y}}{\boldsymbol{\hat{y}}} + \frac{(1-\boldsymbol{y})}{(1-\boldsymbol{\hat{y}})}\right](\hat{y})(1-\hat{y})(A^{[1]})$$

$$\frac{\partial}{\partial W^{[2]}}(Loss) = \left[-\frac{\boldsymbol{y}(\cancel{\boldsymbol{\hat{y}}})(1-\boldsymbol{\hat{y}})}{\cancel{\boldsymbol{\hat{y}}}} + \frac{(1-\boldsymbol{y})(\boldsymbol{\hat{y}})\cancel{(1-\boldsymbol{\hat{y}})}}{\cancel{(1-\boldsymbol{\hat{y}})}}\right](A^{[1]})$$

$$\frac{\partial}{\partial W^{[2]}}(Loss) = [-\boldsymbol{y} + \cancel{\boldsymbol{y\hat{y}}} + \boldsymbol{\hat{y}} - \cancel{\boldsymbol{y\hat{y}}}](A^{[1]})$$

$$\frac{\partial}{\partial W^{[2]}}(Loss) = [\boldsymbol{\hat{y}} - \boldsymbol{y}](A^{[1]})$$

And find gradient of bias from derivative of loss function ($\frac{\partial}{\partial C^{[2]}}(Loss)$) using chain rule method:

$$\frac{\partial}{\partial C^{[2]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial C^{[2]}}(Z^{[2]})$$

Find $\frac{\partial}{\partial C^{[2]}}(Z^{[2]})$:

From task 2, $Z^{[2]} = W^{[2]}A^{[1]} + C^{[2]}$

$$\frac{\partial}{\partial C^{[2]}}(Z^{[2]}) = \frac{\partial}{\partial W^{[2]}}(W^{[2]}A^{[1]}) + \frac{\partial}{\partial C^{[2]}}(C^{[2]})$$

$$\frac{\partial}{\partial C^{[2]}}(Z^{[2]}) = 0 + \frac{\partial}{\partial C^{[2]}}(C^{[2]})$$

$$\frac{\partial}{\partial C^{[2]}}(Z^{[2]}) = \mathbf{1}$$

Adding $\frac{\partial}{\partial C^{[2]}}(Z^{[2]}) = \mathbf{1}$ in chain rule equation:

$$\frac{\partial}{\partial b^{[2]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial C^{[2]}}(Z^{[2]})$$

$$\frac{\partial}{\partial C^{[2]}}(Loss) = -\left[\frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}\right](\hat{y})(1-\hat{y})(1)$$

$$\frac{\partial}{\partial C^{[2]}}(Loss) = \left[-\frac{\boldsymbol{y}}{\boldsymbol{\hat{y}}} + \frac{(1-\boldsymbol{y})}{(1-\boldsymbol{\hat{y}})}\right](\hat{y})(1-\hat{y})$$

$$\frac{\partial}{\partial C^{[2]}}(Loss) = \left[-\frac{\boldsymbol{y}(\cancel{\boldsymbol{\hat{y}}})(1-\boldsymbol{\hat{y}})}{\cancel{\boldsymbol{\hat{y}}}} + \frac{(1-\boldsymbol{y})(\boldsymbol{\hat{y}})\cancel{(1-\boldsymbol{\hat{y}})}}{\cancel{(1-\boldsymbol{\hat{y}})}}\right]$$

$$\frac{\partial}{\partial C^{[2]}}(Loss) = [-\boldsymbol{y} + \cancel{\boldsymbol{y\hat{y}}} + \boldsymbol{\hat{y}} - \cancel{\boldsymbol{y\hat{y}}}]$$

$$\frac{\partial}{\partial C^{[2]}}(Loss) = [\boldsymbol{\hat{y}} - \boldsymbol{y}]$$

Finally, gradient of weight and bias of output Layer:

Gradient of weights $= \frac{\partial}{\partial W^{[2]}}(Loss) = [\hat{\boldsymbol{y}} - \boldsymbol{y}](A^{[1]})$

Gradient of bias $= \frac{\partial}{\partial C^{[2]}}(Loss) = [\hat{\boldsymbol{y}} - \boldsymbol{y}]$

Represent in Matrix Form (For Checking Matrix Dimensions)**:**

Regression Equation from Task 2, $Z^{[1]} = W^{[1]}X + C^{[1]}$

Calculate $W^{[1]}X$,

Matrix of Input, $X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$, Matrix of Weights on Layer 1, $W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix}$

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & W_{22}^{[1]} \\ W_{11}^{[1]} & 0 \\ \left(W_{11}^{[1]} + W_{21}^{[1]}\right) & \left(W_{12}^{[1]} + W_{22}^{[1]}\right) \end{bmatrix}$$

Adding $C^{[1]}$ in Matrix, $C^{[1]} = \begin{bmatrix} C_1^{[1]} & C_2^{[1]} \end{bmatrix}$

$$\begin{bmatrix} 0 & 0 \\ 0 & W_{22}^{[1]} \\ W_{11}^{[1]} & 0 \\ \left(W_{11}^{[1]} + W_{21}^{[1]}\right) & \left(W_{12}^{[1]} + W_{22}^{[1]}\right) \end{bmatrix} + C^{[1]} = \begin{bmatrix} C_1^{[1]} & C_2^{[1]} \\ C_1^{[1]} & W_{22}^{[1]} + C_2^{[1]} \\ W_{11}^{[1]} + C_1^{[1]} & C_2^{[1]} \\ \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_1^{[1]} & \left(W_{12}^{[1]} + W_{22}^{[1]}\right) + C_2^{[1]} \end{bmatrix}$$

Using activation function, reLU function ($y = max\{0, Z\}$):

$$ReLU\left(\begin{bmatrix} C_1^{[1]} & C_2^{[1]} \\ C_1^{[1]} & W_{22}^{[1]} + C_2^{[1]} \\ W_{11}^{[1]} + C_1^{[1]} & C_2^{[1]} \\ (W_{11}^{[1]} + W_{21}^{[1]}) + C_1^{[1]} & (W_{12}^{[1]} + W_{22}^{[1]}) + C_2^{[1]} \end{bmatrix}\right) = \begin{bmatrix} max\{0, C_1^{[1]}\} & max\{0, C_2^{[1]}\} \\ max\{0, C_1^{[1]}\} & max\{0, W_{22}^{[1]} + C_2^{[1]}\} \\ max\{0, W_{11}^{[1]} + C_1^{[1]}\} & max\{0, C_2^{[1]}\} \\ max\{0, (W_{11}^{[1]} + W_{21}^{[1]}) + C_1^{[1]}\} & max\{0, (W_{12}^{[1]} + W_{22}^{[1]}) + C_2^{[1]}\} \end{bmatrix}$$

Adding result of reLU function ($A^{[1]}$) to this equation:

$$Z^{[2]} = W^{[2]}A^{[1]} + C^{[2]}$$

And Matrix of Weight in Output Layer, $W^{[2]} = \begin{bmatrix} W_1^{[2]} \\ W_2^{[2]} \end{bmatrix}$

Calculate $W^{[2]}A^{[1]}$,

$$\begin{bmatrix} max\left\{0, C_1^{[1]}\right\} & max\left\{0, C_2^{[1]}\right\} \\ max\left\{0, C_1^{[1]}\right\} & max\left\{0, W_{22}^{[1]} + C_2^{[1]}\right\} \\ max\left\{0, W_{11}^{[1]} + C_1^{[1]}\right\} & max\left\{0, C_2^{[1]}\right\} \\ max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_1^{[1]}\right\} & max\left\{0, \left(W_{12}^{[1]} + W_{22}^{[1]}\right) + C_2^{[1]}\right\} \end{bmatrix} \begin{bmatrix} W_1^{[2]} \\ W_2^{[2]} \end{bmatrix}$$

$$= \begin{bmatrix} \left(max\{0, C_1^{[1]}\}W_1^{[2]} + max\{0, C_2^{[1]}\}W_2^{[2]}\right) \\ \left(max\{0, C_1^{[1]}\}W_1^{[2]} + max\left\{0, W_{22}^{[1]} + C_2^{[1]}\right\}W_2^{[2]}\right) \\ \left(max\left\{0, W_{11}^{[1]} + C_1^{[1]}\right\}W_1^{[2]} + max\{0, C_2^{[1]}\}W_2^{[2]}\right) \\ \left(max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_1^{[1]}\right\}W_1^{[2]} + max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_2^{[1]}\right\}W_2^{[2]}\right) \end{bmatrix}$$

Adding $C^{[2]}$ in this Matrix, $C^{[2]} = \left[C_1^{[2]}\right]$,

$$\begin{bmatrix} \left(max\{0, C_1^{[1]}\}W_1^{[2]} + max\{0, C_2^{[1]}\}W_2^{[2]}\right) \\ \left(max\{0, C_1^{[1]}\}W_1^{[2]} + max\left\{0, W_{22}^{[1]} + C_2^{[1]}\right\}W_2^{[2]}\right) \\ \left(max\left\{0, W_{11}^{[1]} + C_1^{[1]}\right\}W_1^{[2]} + max\{0, C_2^{[1]}\}W_2^{[2]}\right) \\ \left(max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_1^{[1]}\right\}W_1^{[2]} + max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_2^{[1]}\right\}W_2^{[2]}\right) \end{bmatrix} + C^{[2]}$$

$$= \begin{bmatrix} \left(max\{0, C_1^{[1]}\}W_1^{[2]} + max\{0, C_2^{[1]}\}W_2^{[2]}\right) + C_1^{[2]} \\ \left(max\{0, C_1^{[1]}\}W_1^{[2]} + max\left\{0, W_{22}^{[1]} + C_2^{[1]}\right\}W_2^{[2]}\right) + C_1^{[2]} \\ \left(max\left\{0, W_{11}^{[1]} + C_1^{[1]}\right\}W_1^{[2]} + max\{0, C_2^{[1]}\}W_2^{[2]}\right) + C_1^{[2]} \\ \left(max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_1^{[1]}\right\}W_1^{[2]} + max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_2^{[1]}\right\}W_2^{[2]}\right) + C_1^{[2]} \end{bmatrix}$$

Using activation function, sigmoid function ($y = Sigmoid(z) \; = \; \sigma(z) = \frac{1}{1+e^{-z}}$):

$$\sigma\left(\begin{bmatrix} \left(max\left\{0, C_1^{[1]}\right\}W_1^{[2]} + max\left\{0, C_2^{[1]}\right\}W_2^{[2]}\right) + C_1^{[2]} \\ \left(max\left\{0, C_1^{[1]}\right\}W_1^{[2]} + max\left\{0, W_{22}^{[1]} + C_2^{[1]}\right\}W_2^{[2]}\right) + C_1^{[2]} \\ \left(max\left\{0, W_{11}^{[1]} + C_1^{[1]}\right\}W_1^{[2]} + max\left\{0, C_2^{[1]}\right\}W_2^{[2]}\right) + C_1^{[2]} \\ \left(max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_1^{[1]}\right\}W_1^{[2]} + max\left\{0, \left(W_{11}^{[1]} + W_{21}^{[1]}\right) + C_2^{[1]}\right\}W_2^{[2]}\right) + C_1^{[2]} \end{bmatrix}\right)$$

$$= \begin{bmatrix} \dfrac{1}{1 + e^{-\left(\left(max\left\{0,C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \dfrac{1}{1 + e^{-\left(\left(max\left\{0,C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,W_{22}^{[1]}+C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \dfrac{1}{1 + e^{-\left(\left(max\left\{0,W_{11}^{[1]}+C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \dfrac{1}{1 + e^{-\left(\left(max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \end{bmatrix}$$

And predict output $\hat{y} = A^{[2]} \; = \; \sigma(Z^{[2]})$

Matrix of Predict Output,

$$\widehat{\boldsymbol{y}} = \begin{bmatrix} \frac{1}{1+e^{-\left(\left(max\{0,C_1^{[1]}\}W_1^{[2]}+max\{0,C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \frac{1}{1+e^{-\left(\left(max\{0,C_1^{[1]}\}W_1^{[2]}+max\{0,W_{22}^{[1]}+C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \frac{1}{1+e^{-\left(\left(max\{0,W_{11}^{[1]}+C_1^{[1]}\}W_1^{[2]}+max\{0,C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \frac{1}{1+e^{-\left(\left(max\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1^{[1]}\}W_1^{[2]}+max\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{y}}_{\boldsymbol{1}} \\ \widehat{\boldsymbol{y}}_{\boldsymbol{2}} \\ \widehat{\boldsymbol{y}}_{\boldsymbol{3}} \\ \widehat{\boldsymbol{y}}_{\boldsymbol{4}} \end{bmatrix}$$

$$\widehat{\boldsymbol{y}}_{\boldsymbol{1}} = \frac{1}{1+e^{-\left(\left(max\{0,C_1^{[1]}\}W_1^{[2]}+max\{0,C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}}$$

$$\widehat{\boldsymbol{y}}_{\boldsymbol{2}} = \frac{1}{1+e^{-\left(\left(max\{0,C_1^{[1]}\}W_1^{[2]}+max\{0,W_{22}^{[1]}+C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}}$$

$$\widehat{\boldsymbol{y}}_{\boldsymbol{3}} = \frac{1}{1+e^{-\left(\left(max\{0,W_{11}^{[1]}+C_1^{[1]}\}W_1^{[2]}+max\{0,C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}}$$

$$\widehat{\boldsymbol{y}}_{\boldsymbol{4}} = \frac{1}{1+e^{-\left(\left(max\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1^{[1]}\}W_1^{[2]}+max\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}}$$

From Gradient of weights $= \frac{\partial}{\partial W^{[2]}}(Loss) = [\widehat{\boldsymbol{y}} - \boldsymbol{y}]\left(A^{[1]}\right)$

Change in Matrix Form,

$$\frac{\partial}{\partial W^{[2]}}(Loss) = [\widehat{\boldsymbol{y}} - \boldsymbol{y}] \times \left[A^{[1]}\right]$$

And Matrix of Output, $y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Calculate $[\widehat{\boldsymbol{y}} - \boldsymbol{y}]$ in Matrix,

$$[\widehat{\boldsymbol{y}} - \boldsymbol{y}] = \begin{bmatrix} \frac{1}{1+e^{-\left(\left(max\{0,C_1^{[1]}\}W_1^{[2]}+max\{0,C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \frac{1}{1+e^{-\left(\left(max\{0,C_1^{[1]}\}W_1^{[2]}+max\{0,W_{22}^{[1]}+C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \frac{1}{1+e^{-\left(\left(max\{0,W_{11}^{[1]}+C_1^{[1]}\}W_1^{[2]}+max\{0,C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \frac{1}{1+e^{-\left(\left(max\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1^{[1]}\}W_1^{[2]}+max\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_2^{[1]}\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Calculate $[\hat{\boldsymbol{y}} - \boldsymbol{y}] \times [A^{[1]}]$

$$[\hat{\boldsymbol{y}} - \boldsymbol{y}]^T = \begin{bmatrix} \dfrac{1}{1+e^{-\left(\left(max\left\{0,C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \dfrac{1}{1+e^{-\left(\left(max\left\{0,C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,W_{22}^{[1]}+C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \dfrac{1}{1+e^{-\left(\left(max\left\{0,W_{11}^{[1]}+C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \\ \dfrac{1}{1+e^{-\left(\left(max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1^{[1]}\right\}W_1^{[2]}+max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_2^{[1]}\right\}W_2^{[2]}\right)+C_1^{[2]}\right)}} \end{bmatrix}^T$$

$[\hat{\boldsymbol{y}} - \boldsymbol{y}]^T \times [A^{[1]}]$

$$= \begin{bmatrix} \dfrac{1}{1+e^{-\left(\left(max\{0,C_1\}W_1^{[2]}+max\{0,C_1\}W_2^{[2]}\right)+C_2\right)}} \\ \dfrac{1}{1+e^{-\left(\left(max\{0,C_1\}W_1^{[2]}+max\left\{0,W_{22}^{[1]}+C_1\right\}W_2^{[2]}\right)+C_2\right)}} - 1 \\ \dfrac{1}{1+e^{-\left(\left(max\left\{0,W_{11}^{[1]}+C_1\right\}W_1^{[2]}+max\{0,C_1\}W_2^{[2]}\right)+C_2\right)}} - 1 \\ \dfrac{1}{1+e^{-\left(\left(max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1\right\}W_1^{[2]}+max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1\right\}W_2^{[2]}\right)+C_2\right)}} \end{bmatrix}^T \begin{bmatrix} max\{0, C_1^{[1]}\} & max\{0, C_2^{[1]}\} \\ max\{0, C_1^{[1]}\} & max\{0, W_{22}^{[1]} + C_2^{[1]}\} \\ max\{0, W_{11}^{[1]} + C_1^{[1]}\} & max\{0, C_2^{[1]}\} \\ max\{0, (W_{11}^{[1]} + W_{21}^{[1]}) + C_1^{[1]}\} & max\{0, (W_{12}^{[1]} + W_{22}^{[1]}) + C_2^{[1]}\} \end{bmatrix}$$

$$= [W_1^{[2]} \quad W_2^{[2]}]$$

Calculate each weight in Matrix:

$$W_1^{[2]} = \left(\frac{1}{1+e^{-\left(\left(max\{0,C_1\}W_1^{[2]}+max\{0,C_1\}W_2^{[2]}\right)+C_2\right)}}\right)\left(max\{0, C_1^{[1]}\}\right)$$
$$+\left(\frac{1}{1+e^{-\left(\left(max\{0,C_1\}W_1^{[2]}+max\left\{0,W_{22}^{[1]}+C_1\right\}W_2^{[2]}\right)+C_2\right)}} - 1\right)\left(max\{0, C_1^{[1]}\}\right)$$
$$+\left(\frac{1}{1+e^{-\left(\left(max\left\{0,W_{11}^{[1]}+C_1\right\}W_1^{[2]}+max\{0,C_1\}W_2^{[2]}\right)+C_2\right)}} - 1\right)\left(max\{0, W_{11}^{[1]} + C_1^{[1]}\}\right)$$
$$+\left(\frac{1}{1+e^{-\left(\left(max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1\right\}W_1^{[2]}+max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1\right\}W_2^{[2]}\right)+C_2\right)}}\right)\left(max\{0, (W_{11}^{[1]} + W_{21}^{[1]}) + C_1^{[1]}\}\right)$$

$$W_2^{[2]} = \left(\frac{1}{1+e^{-\left(\left(max\{0,C_1\}W_1^{[2]}+max\{0,C_1\}W_2^{[2]}\right)+C_2\right)}}\right)\left(max\{0, C_1^{[1]}\}\right)$$
$$+\left(\frac{1}{1+e^{-\left(\left(max\{0,C_1\}W_1^{[2]}+max\left\{0,W_{22}^{[1]}+C_1\right\}W_2^{[2]}\right)+C_2\right)}} - 1\right)\left(max\{0, C_1^{[1]}\}\right)$$
$$+\left(\frac{1}{1+e^{-\left(\left(max\left\{0,W_{11}^{[1]}+C_1\right\}W_1^{[2]}+max\{0,C_1\}W_2^{[2]}\right)+C_2\right)}} - 1\right)\left(max\{0, W_{11}^{[1]} + C_1^{[1]}\}\right)$$
$$+\left(\frac{1}{1+e^{-\left(\left(max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1\right\}W_1^{[2]}+max\left\{0,\left(W_{11}^{[1]}+W_{21}^{[1]}\right)+C_1\right\}W_2^{[2]}\right)+C_2\right)}}\right)\left(max\{0, (W_{11}^{[1]} + W_{21}^{[1]}) + C_1^{[1]}\}\right)$$

Task 5: Derive the gradient of weights in hidden layers. Show your work.

We find the gradient of weights in hidden layers from derivative of activation function in hidden layers ($\frac{\partial}{\partial W^{[1]}}(Loss)$) using chain rule method:

$$\frac{\partial}{\partial W^{[1]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right)\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right)\frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right)$$

We know $\frac{\partial}{\partial \hat{y}}(Loss), \frac{\partial}{\partial Z^{[2]}}(\hat{y})$ from task 4, then we find other part of chain rule method,

$$\frac{\partial}{\partial Z^{[2]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y}) = [\hat{\boldsymbol{y}} - \boldsymbol{y}]$$

Find $\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right)$:

From Task 2, $Z^{[2]} = W^{[2]}A^{[1]} + C^{[2]}$

$$\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right) = \frac{\partial}{\partial A^{[1]}}\left(W^{[2]}A^{[1]} + C^{[2]}\right)$$

$$\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right) = \frac{\partial}{\partial A^{[1]}}(W^{[2]}A^{[1]}) + \frac{\partial}{\partial A^{[1]}}(C^{[2]})$$

$$\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right) = W^{[2]}\frac{\partial}{\partial A^{[1]}}(A^{[1]}) + 0$$

$$\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right) = W^{[2]}(1) = W^{[2]}$$

Find $\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right)$:

From Task 2, $A^{[1]} = ReLU\left(Z^{[1]}\right) = max\left\{0, \left(Z^{[1]}\right)\right\}$

Derivative of $A^{[1]}$,

$$\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right) = \frac{\partial}{\partial Z^{[1]}}\left(ReLU\left(Z^{[1]}\right)\right)$$

$$\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right) = \frac{\partial}{\partial Z^{[1]}}\left(max\left\{0, \left(Z^{[1]}\right)\right\}\right)$$

$$\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right) = \begin{cases} 0, & if\ A^{[1]} \le 0 \\ \frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right), & if\ A^{[1]} > 0 \end{cases}$$

$$\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right) = reLU'\left(A^{[1]}\right) = \begin{cases} 0, & if\ A^{[1]} \le 0 \\ 1, & if\ A^{[1]} > 0 \end{cases}$$

Find $\frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right)$:

From Task 2, $Z^{[1]} = W^{[1]}X + C^{[1]}$

$$\frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right) = \frac{\partial}{\partial W^{[1]}}\left(W^{[1]}X\ +\ C^{[1]}\right)$$

$$\frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right) = \frac{\partial}{\partial W^{[1]}}(W^{[1]}X)\ +\ \frac{\partial}{\partial W^{[1]}}(C^{[1]})$$

$$\frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right) = X\frac{\partial}{\partial W^{[1]}}(W^{[1]})\ +\ 0$$

$$\frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right) = X(1) = X$$

And adding $\frac{\partial}{\partial Z^{[2]}}(Loss), \frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right), \frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right), \frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right)$ in chain rule equation:

$$\frac{\partial}{\partial W^{[1]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right)\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right)\frac{\partial}{\partial W^{[1]}}\left(Z^{[1]}\right)$$

$$\frac{\partial}{\partial W^{[1]}}(Loss) = [\hat{\boldsymbol{y}} - \boldsymbol{y}] \times W^{[2]} \times reLU'\left(A^{[1]}\right) \times (X)$$

And find gradient of bias from derivative of $A^{[1]}$ ($\frac{\partial}{\partial C^{[1]}}(Loss)$) using chain rule method:

$$\frac{\partial}{\partial C^{[1]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right)\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right)\frac{\partial}{\partial C^{[1]}}\left(Z^{[1]}\right)$$

Find $\frac{\partial}{\partial C^{[1]}}\left(Z^{[1]}\right)$:

From Task 2, $Z^{[1]} = W^{[1]}X\ +\ C^{[1]}$

$$\frac{\partial}{\partial C^{[1]}}\left(Z^{[1]}\right) = \frac{\partial}{\partial C^{[1]}}\left(W^{[1]}X\ +\ C^{[1]}\right)$$

$$\frac{\partial}{\partial C^{[1]}}\left(Z^{[1]}\right) = \frac{\partial}{\partial C^{[1]}}(W^{[1]}X)\ +\ \frac{\partial}{\partial C^{[1]}}(C^{[1]})$$

$$\frac{\partial}{\partial C^{[1]}}\left(Z^{[1]}\right) = (0)\ +\ (1) = 1$$

And adding $\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right), \frac{\partial}{\partial C^{[1]}}\left(Z^{[1]}\right)$ in chain rule equation:

$$\frac{\partial}{\partial C^{[1]}}(Loss) = \frac{\partial}{\partial \hat{y}}(Loss)\frac{\partial}{\partial Z^{[2]}}(\hat{y})\frac{\partial}{\partial A^{[1]}}\left(Z^{[2]}\right)\frac{\partial}{\partial Z^{[1]}}\left(A^{[1]}\right)\frac{\partial}{\partial C^{[1]}}\left(Z^{[1]}\right)$$

$$\frac{\partial}{\partial C^{[1]}}(Loss) = [\hat{\boldsymbol{y}} - \boldsymbol{y}] \times W^{[2]} \times reLU'\left(A^{[1]}\right)$$

Finally, gradient of weight and bias of hidden Layer:

Gradient of weights $= \frac{\partial}{\partial W^{[1]}}(Loss) = [\hat{\boldsymbol{y}} - \boldsymbol{y}] \times W^{[2]} \times reLU'\left(A^{[1]}\right) \times (X)$

Gradient of bias $= \frac{\partial}{\partial C^{[1]}}(Loss) = [\hat{\boldsymbol{y}} - \boldsymbol{y}] \times W^{[2]} \times reLU'\left(A^{[1]}\right)$

Task 6: Prepare the XOR data point.

XOR Function Table

Input		Output
X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Matrix of Input, $X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$

Matrix of Weights in Hidden Layer, $W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix}$

Matrix of Bias in Hidden Layer, $C^{[1]} = \begin{bmatrix} C_1^{[1]} & C_2^{[1]} \end{bmatrix}$

Matrix of Weight in Output Layer, $W^{[2]} = \begin{bmatrix} W_1^{[2]} \\ W_2^{[2]} \end{bmatrix}$

Matrix of Bias in Output Layer, $C^{[2]} = \begin{bmatrix} C_1^{[2]} \end{bmatrix}$

Matrix of Output, $y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Code:

```
x = np.array(([0, 0], [0, 1], [1, 0], [1, 1]), dtype=float) # input data
y = np.array(([0], [1], [1], [0]), dtype=float) # output

#parameters
self.input_Layer = inputLayer
self.output_Layer = outputLayer
self.hidden_Layer = hiddenLayer

# Initial weights set in small values
# (2x2) weight matrix from input to hidden layer
self.W1 = np.array(np.random.randn(self.input_Layer, self.hidden_Layer)) * 0.0
1
# (2x1) weight matrix from hidden to output layer
self.W2 = np.array(np.random.randn(self.hidden_Layer, self.output_Layer)) * 0.
01

# Initial bias
# (1X2) bias matrix from input to hidden layer
self.C1 = np.zeros((1, self.input_Layer))
# (1x1) bias matrix from hidden to output layer
self.C2 = np.zeros((1, self.output_Layer))
```

Task 7: Implement the gradient descent using all of the dataset. The training should be done instance by instance and repeat for a number of iterations. Weights should be initialized randomly or fixed to small values.

Code:

```
    def forward(self, X, y):
        # Hidden Layer
        self.z1 = np.dot(X,self.W1)
        self.z1 += self.C1
        self.a1 = self.reLU(self.z1)

        # Output Layer
        self.z2 = np.dot(self.a1, self.W2)
        self.z2 += self.C2
        self.a2 = self.sigmoid(self.z2)
        self.predict = np.copy(self.a2)

        # Calculate loss function
        self.loss = (-
1/self.predict.shape[0]) * (np.sum((y*np.log(self.predict)) + ((1-
y)*(np.log(1-self.predict)))))

        self.history_cost.append(self.loss)

        return self.predict

    def backward(self, X, y, predict, lr):
        # Calculate in Output Layer
        # error in output layer
        error = predict - y


        # Calculate gradient of weight in Output Layer
        dw2 = error * self.sigmoidPrime(predict)

        # Calculate in Hidden Layer
        # error in hidden layer
        error_hidden = np.dot(dw2, self.W2.T)
        dw1 = error_hidden * self.reLUprime(self.a1)

        # Update Gradient Descent
        self.W2 -= np.dot(self.a1.T, dw2) * lr
        self.C2 -= np.sum(dw2, axis=0, keepdims=True) * lr
        self.W1 -= np.dot(X.T, dw1) * lr
        self.C1 -= np.sum(dw1, axis=0, keepdims=True) * lr
    def train(self, X, y, epochs, lr=0.1):
        self.saveInitialModel()
        for _ in range(epochs):
            print ("Training Epochs: {0}".format(_))
            predict = self.forward(X,y)
            self.backward(X=X, y=y, predict=predict, lr=lr)
        self.saveModel()
```

Task 8: Show how the error changes by iterations.

Testing on Learning Rate = 0.1,

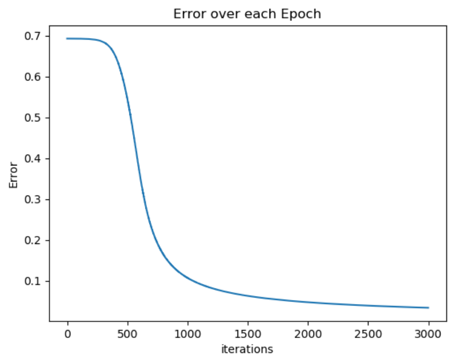
Initial Value

$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.01134429750092792 & -0.005431144786305983 \\ -0.02946391245206806 & 0.009130128718639715 \end{bmatrix}$$

$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.0165294513881923 \\ 0.00740859560094127 \end{bmatrix}$$

$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 \end{bmatrix}$$

$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = [0.0]$$



Testing on Learning Rate = 0.5,

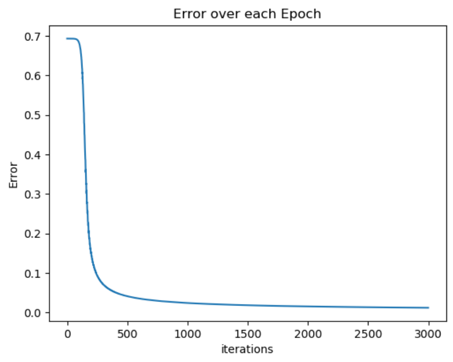
Initial Value

$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} -0.006562361510258101 & 0.009170926540818098 \\ 0.0015321627029268947 & -0.0028099430146093585 \end{bmatrix}$$

$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.002587491199421111 \\ 0.00203392242200524 \end{bmatrix}$$

$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 \end{bmatrix}$$

$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = [0.0]$$



Testing on Learning Rate = 0.8,

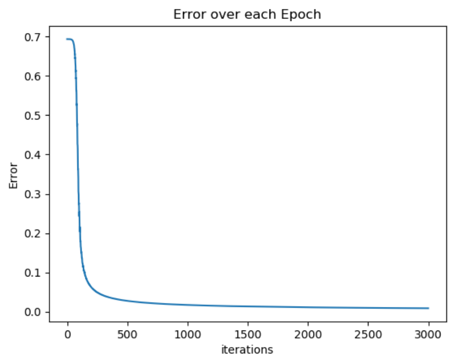
Initial Value

$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} -0.011891386793450223 & 0.010329700885396349 \\ 0.0012844290305561669 & -0.0028750012432950133 \end{bmatrix}$$

$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.0120923619968342 \\ 0.00249207907859658 \end{bmatrix}$$

$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 \end{bmatrix}$$

$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.0 \end{bmatrix}$$



Task 9: Demonstrate that the network has finally learned the XOR function.

Final learned the XOR function on Learning Rate = 0.1,

Initial Value

$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.01134429750092792 & -0.005431144786305983 \\ -0.02946391245206806 & 0.009130128718639715 \end{bmatrix}$$
$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.0165294513881923 \\ 0.00740859560094127 \end{bmatrix}$$
$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = [0.0 \quad 0.0]$$
$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = [0.0]$$

Final Value

$$Predict\ Output\ = \hat{y} = \begin{bmatrix} 0.0123681709793299 \\ 0.994024680368648 \\ 0.994021313800612 \\ 0.01236552779224560 \end{bmatrix}$$
$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.01134429750092792 & -0.005431144786305983 \\ -0.02946391245206806 & 0.009130128718639715 \end{bmatrix}$$
$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.0165294513881923 \\ 0.00740859560094127 \end{bmatrix}$$
$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = [1.6090487936192986 \times 10^{-5} \quad -0.00029069607024283575]$$
$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = [-3.09391257919574]$$

Final learned the XOR function on Learning Rate = 0.5,

Initial Value

$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} -0.006562361510258101 & 0.009170926540818098 \\ 0.0015321627029268947 & -0.0028099430146093585 \end{bmatrix}$$
$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.00258749119942111 \\ 0.00203392242200524 \end{bmatrix}$$
$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = [0.0 \quad 0.0]$$
$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = [0.0]$$

Final Value

$$Predict\ Output\ = \hat{y} = \begin{bmatrix} 0.01609302 \\ 0.99203458 \\ 0.99203233 \\ 0.01609302 \end{bmatrix}$$
$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} -2.529688736974373 & 2.5286490313879977 \\ 2.5297154708671274 & -2.5287611214898345 \end{bmatrix}$$
$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 3.53369654075438 \\ 3.53513251772598 \end{bmatrix}$$
$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = [-0.00013158576476996 \quad -0.000172993384828942]$$
$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = [-4.11333747028484]$$

Final learned the XOR function on Learning Rate = 0.8,

Initial Value

$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.01134429750092792 & -0.005431144786305983 \\ -0.02946391245206806 & 0.009130128718639715 \end{bmatrix}$$

$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.0165294513881923 \\ 0.00740859560094127 \end{bmatrix}$$

$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 \end{bmatrix}$$

$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = \begin{bmatrix} 0.0 \end{bmatrix}$$

Final Value

$$Predict\ Output\ = \hat{y} = \begin{bmatrix} 0.0123681709793299 \\ 0.994024680368648 \\ 0.994021313800612 \\ 0.01236552779224560 \end{bmatrix}$$

$$W^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} \end{bmatrix} = \begin{bmatrix} -2.622021686080415 & 2.6141898435722952 \\ 2.6211896758459354 & -2.6142433284483273 \end{bmatrix}$$

$$W^{[2]} = \begin{bmatrix} W_{1}^{[2]} \\ W_{2}^{[2]} \end{bmatrix} = \begin{bmatrix} 3.62174895036096 \\ 3.63197475118153 \end{bmatrix}$$

$$C^{[1]} = \begin{bmatrix} C_{1}^{[1]} & C_{2}^{[1]} \end{bmatrix} = \begin{bmatrix} -0.000674758663686778 & 0.00010129102755683624 \end{bmatrix}$$

$$C^{[2]} = \begin{bmatrix} C_{1}^{[2]} \end{bmatrix} = \begin{bmatrix} -4.38072230736342 \end{bmatrix}$$

Appendix

Assigment 2 Code:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array(([0, 0], [0, 1], [1, 0], [1, 1]), dtype=float) # input data
y = np.array(([0], [1], [1], [0]), dtype=float) # output

class NeuralNetwork(object):
    def __init__(self, inputLayer=2,hiddenLayer=2, outputLayer = 1):
        #parameters
        self.input_Layer = inputLayer
        self.output_Layer = outputLayer
        self.hidden_Layer = hiddenLayer

        # Initial weights set in small values
        # (2x2) weight matrix from input to hidden layer
        self.W1 = np.array(np.random.randn(self.input_Layer, self.hidden_Layer)) * 0.01
        # (2x1) weight matrix from hidden to output layer
        self.W2 = np.array(np.random.randn(self.hidden_Layer, self.output_Layer)) * 0.01

        # Initial bias
        # (1X2) bias matrix from input to hidden layer
        self.C1 = np.zeros((1, self.input_Layer))
        # (1x1) bias matrix from hidden to output layer
        self.C2 = np.zeros((1, self.output_Layer))

        self.history_cost = []

    def reLU(self,X):
        # reLU function
        return np.maximum(0, X)

    def reLUprime(self,x):
        # Derivative of reLU function
        return np.where(x > 0, 1.0, 0.0)

    def sigmoid(self, s):
        # activation function
        return 1/(1+np.exp(-s))

    def sigmoidPrime(self, s):
        #derivative of sigmoid
        return s * (1 - s)

    def forward(self, X, y):
        # Hidden Layer
        self.z1 = np.dot(X,self.W1)
        self.z1 += self.C1
        self.a1 = self.reLU(self.z1)

        # Output Layer
        self.z2 = np.dot(self.a1, self.W2)
        self.z2 += self.C2
        self.a2 = self.sigmoid(self.z2)
        self.predict = np.copy(self.a2)

        # Calculate loss function
        self.loss = (-1/self.predict.shape[0]) * (np.sum((y*np.log(self.predict)) + ((1-y)*(np.log(1-
self.predict)))))

        self.history_cost.append(self.loss)

        return self.predict

    def backward(self, X, y, predict, lr):
        # Calculate in Output Layer
        # error in output layer
        error = predict - y


        # Calculate gradient of weight in Output Layer
        dw2 = error * self.sigmoidPrime(predict)

        # Calculate in Hidden Layer
        # error in hidden layer
        error_hidden = np.dot(dw2, self.W2.T)
        dw1 = error_hidden * self.reLUprime(self.a1)

        # Update Gradient Descent
        self.W2 -= np.dot(self.a1.T, dw2) * lr
        self.C2 -= np.sum(dw2, axis=0, keepdims=True) * lr
        self.W1 -= np.dot(X.T, dw1) * lr
        self.C1 -= np.sum(dw1, axis=0, keepdims=True) * lr

    def train(self, X, y, epochs, lr=0.1):
        self.saveInitialModel()
        for _ in range(epochs):
            print ("Training Epochs: {0}".format(_))
            predict = self.forward(X,y)
            self.backward(X=X, y=y, predict=predict, lr=lr)
        self.saveModel()

    def saveModel(self):
        np.savetxt("w1.csv", self.W1, fmt="%s")
        np.savetxt("w2.csv", self.W2, fmt="%s")
        np.savetxt("C1.csv", self.C1, fmt="%s")
        np.savetxt("C2.csv", self.C2, fmt="%s")
        np.savetxt("Predict_Output.csv", self.predict, fmt="%s")

    def saveInitialModel(self):
        np.savetxt("Initial_w1.csv", self.W1, fmt="%s")
        np.savetxt("Initial_w2.csv", self.W2, fmt="%s")
        np.savetxt("Initial_C1.csv", self.C1, fmt="%s")
        np.savetxt("Initial_C2.csv", self.C2, fmt="%s")


NN = NeuralNetwork()
NN.train(x,y,3000,lr=0.1)
print ("Weight 1: \n{0}".format(NN.W1))
print ("Weight 2: \n{0}".format(NN.W2))
print ("Predict Output: \n{0}".format(NN.predict))

plt.figure("Error")
plt.plot(NN.history_cost)
plt.ylabel('Error')
plt.xlabel('iterations')
plt.title('Error over each Epoch')
plt.show()
```

References:

- Lecture 6: Deep Forward Network and Backpropagation, CPE 663 Special Topic: Deep Learning, 1/2019, Asst. Prof. Dr. Santitham Prom-on, Department of Computer Engineering, Faculty of Engineering King Mongkut's University of Technology Thonburi
- https://towardsdatascience.com/lets-code-a-neural-network-in-plain-numpy-ae7e74410795
- https://towardsdatascience.com/the-keys-of-deep-learning-in-100-lines-of-code-907398c76504
- https://towardsdatascience.com/coding-a-2-layer-neural-network-from-scratch-in-python-4dd022d19fd2
- https://github.com/yunjey/cs231n/blob/master/assignment1/cs231n/classifiers/neural_net.py
- https://repl.it/@shamdasani/Enlight-Neural-Network
- https://tomaszgolan.github.io/introduction_to_machine_learning/markdown/introduction_to_machine_learning_04_nn/introduction_to_machine_learning_04_nn/
- https://datascienceintuition.wordpress.com/2018/01/16/logistic-regression-as-neural-networks/
- https://youtu.be/Dws9Zveu9ug
- https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/
- https://medium.com/@pdquant/all-the-backpropagation-derivatives-d5275f727f60
- http://neuralnetworksanddeeplearning.com/chap3.html
- https://towardsdatascience.com/implementing-the-xor-gate-using-backpropagation-in-neural-networks-c1f255b4f20d
- https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/