**Vision Based Automation**

**Name: Vivek Chandrashekhar Rugale**

**GR No.: 11810369**

**Div.: TY-C**

**Roll No.: 24 (Batch B1)**

**LAB 6: Iris flower classification using NN**

*# Dependencies*

**import** tensorflow **as** tf

**import** pandas **as** pd

**import** numpy **as** np

*# Make results reproducible*

seed **=** 1234

np**.**random**.**seed(seed)

*# Loading the dataset*

dataset **=** pd**.**read\_csv('Iris.csv')

dataset **=** pd**.**get\_dummies(dataset, columns**=**['Species']) *# One Hot Encoding*

values **=** list(dataset**.**columns**.**values)

y **=** dataset[values[**-**3:]]

y **=** np**.**array(y, dtype**=**'float32')

X **=** dataset[values[1:**-**3]]

X **=** np**.**array(X, dtype**=**'float32')

*# Shuffle Data*

indices **=** np**.**random**.**choice(len(X), len(X), replace**=False**)

X\_values **=** X[indices]

y\_values **=** y[indices]

*# Creating a Train and a Test Dataset*

test\_size **=** 10

X\_test **=** X\_values[**-**test\_size:]

X\_train **=** X\_values[:**-**test\_size]

y\_test **=** y\_values[**-**test\_size:]

y\_train **=** y\_values[:**-**test\_size]

*# Session*

sess **=** tf**.**compat**.**v1**.**Session()

*# Interval / Epochs*

interval **=** 50

epoch **=** 500

**import** tensorflow.compat.v1 **as** tf

tf**.**disable\_v2\_behavior()

*# Initialize placeholders*

X\_data **=** tf**.**placeholder(shape**=**[**None**, 4], dtype**=**tf**.**float32)

y\_target **=** tf**.**placeholder(shape**=**[**None**, 3], dtype**=**tf**.**float32)

*# Input neurons : 4*

*# Hidden neurons : 8*

*# Output neurons : 3*

hidden\_layer\_nodes **=** 8

*# Create variables for Neural Network layers*

w1 **=** tf**.**Variable(tf**.**random\_normal(shape**=**[4,hidden\_layer\_nodes])) *# Inputs -> Hidden Layer*

b1 **=** tf**.**Variable(tf**.**random\_normal(shape**=**[hidden\_layer\_nodes])) *# First Bias*

w2 **=** tf**.**Variable(tf**.**random\_normal(shape**=**[hidden\_layer\_nodes,3])) *# Hidden layer -> Outputs*

b2 **=** tf**.**Variable(tf**.**random\_normal(shape**=**[3])) *# Second Bias*

*# Operations*

hidden\_output **=** tf**.**nn**.**relu(tf**.**add(tf**.**matmul(X\_data, w1), b1))

final\_output **=** tf**.**nn**.**softmax(tf**.**add(tf**.**matmul(hidden\_output, w2), b2))

*# Cost Function*

loss **=** tf**.**reduce\_mean(**-**tf**.**reduce\_sum(y\_target **\*** tf**.**log(final\_output), axis**=**0))

*# Optimizer*

optimizer **=** tf**.**train**.**GradientDescentOptimizer(learning\_rate**=**0.001)**.**minimize(loss)

*# Initialize variables*

init **=** tf**.**global\_variables\_initializer()

sess**.**run(init)

*# Training*

ls **=** {}

print('Training the model...')

**for** i **in** range(1, (epoch **+** 1)):

sess**.**run(optimizer, feed\_dict**=**{X\_data: X\_train, y\_target: y\_train})

**if** i **%** interval **==** 0:

l **=** sess**.**run(loss, feed\_dict**=**{X\_data: X\_train, y\_target: y\_train})

print('Epoch', i, '|', 'Loss:', l)

ls[i] **=** l

Training the model...

Epoch 50 | Loss: 14.917066

Epoch 100 | Loss: 11.265358

Epoch 150 | Loss: 9.466841

Epoch 200 | Loss: 8.288619

Epoch 250 | Loss: 7.451693

Epoch 300 | Loss: 6.82758

Epoch 350 | Loss: 6.345309

Epoch 400 | Loss: 5.962297

Epoch 450 | Loss: 5.6509037

Epoch 500 | Loss: 5.3927016

**import** matplotlib.pyplot **as** plt

iterations **=** list(ls**.**keys())

costs **=** list(ls**.**values())

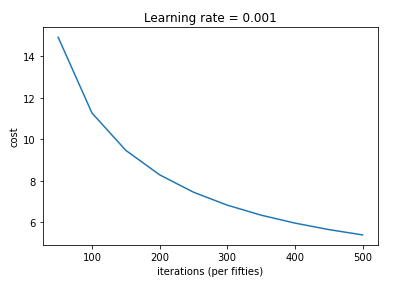
plt**.**plot(iterations,costs)

plt**.**ylabel('cost')

plt**.**xlabel('iterations (per fifties)')

plt**.**title("Learning rate = " **+** str(**.**001))

plt**.**show()



*# Prediction*

**for** i **in** range(len(X\_test)):

print('Actual:', y\_test[i], 'Predicted:', np**.**rint(sess**.**run(final\_output, feed\_dict**=**{X\_data: [X\_test[i]]})))

Actual: [0. 0. 1.] Predicted: [[0. 0. 1.]]

Actual: [1. 0. 0.] Predicted: [[1. 0. 0.]]

Actual: [0. 0. 1.] Predicted: [[0. 0. 1.]]

Actual: [1. 0. 0.] Predicted: [[1. 0. 0.]]

Actual: [1. 0. 0.] Predicted: [[1. 0. 0.]]

Actual: [0. 0. 1.] Predicted: [[0. 0. 1.]]

Actual: [0. 0. 1.] Predicted: [[0. 0. 1.]]

Actual: [0. 1. 0.] Predicted: [[0. 1. 0.]]

Actual: [1. 0. 0.] Predicted: [[1. 0. 0.]]

Actual: [1. 0. 0.] Predicted: [[1. 0. 0.]]