Artificial Intelligence

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH

TU DUBLIN

TECHNOLOGICAL
UNIVERSITY DUBLIN

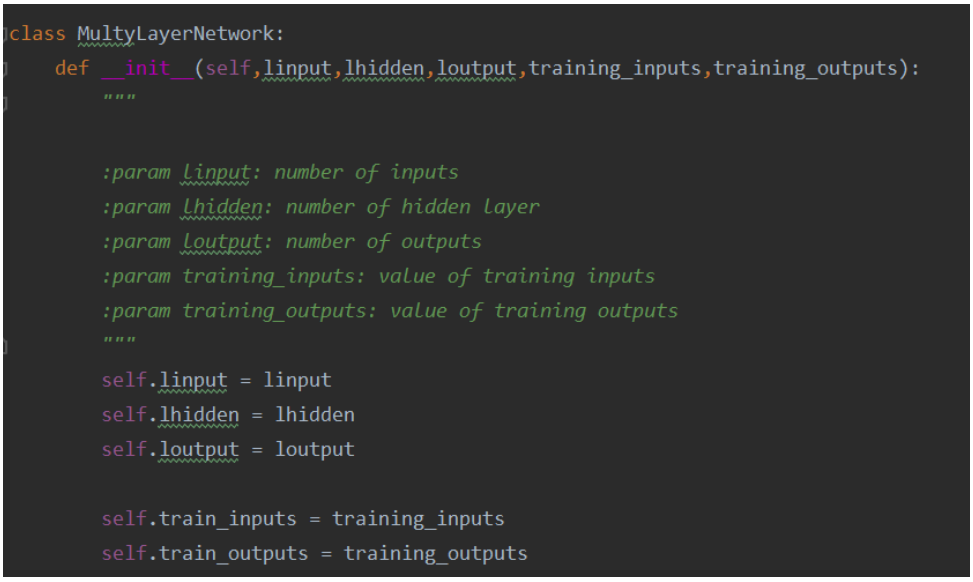
DT228/3 – Artificial Intelligence

Eurico Carajote

Code:

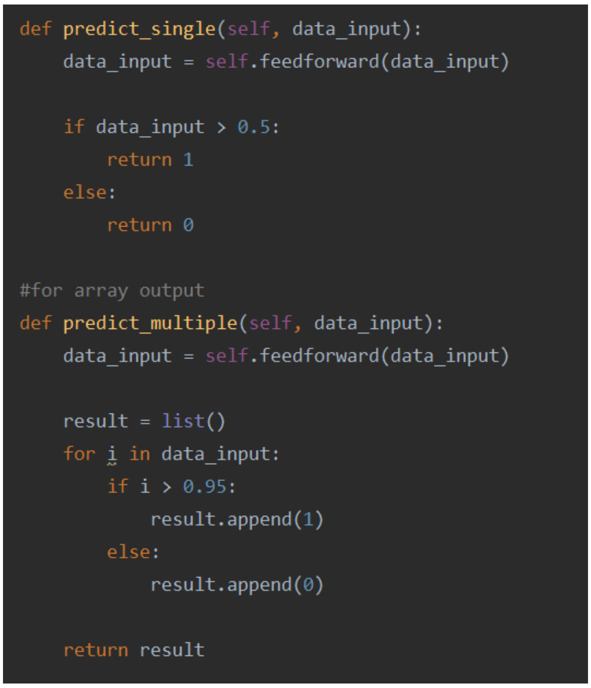
In my code I’ve only made a few changes, I made the program modular so that I can just input my parameters for different questions, I’ve also implemented my predict functions, one for a singular output and one for an array of outputs, in addition to this my cost function is running as a cross entropy function. In the following screen shot there is also my code to write my output and predict statement into a csv file using pandas.

Modularity:



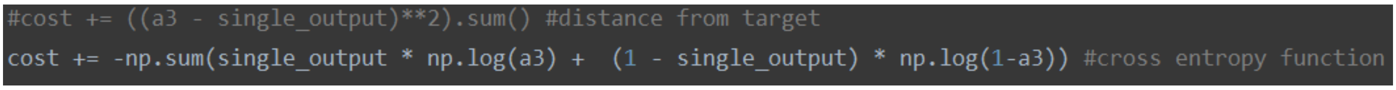
```
class MultyLayerNetwork:
    def __init__(self,linput,lhidden,loutput,training_inputs,training_outputs):
        """

        :param linput: number of inputs
        :param lhidden: number of hidden layer
        :param loutput: number of outputs
        :param training_inputs: value of training inputs
        :param training_outputs: value of training outputs
        """
        self.linput = linput
        self.lhidden = lhidden
        self.loutput = loutput

        self.train_inputs = training_inputs
        self.train_outputs = training_outputs
```

Predict Functions:



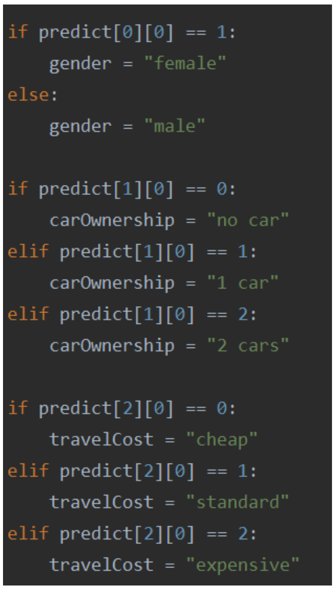
```
def predict_single(self, data_input):
    data_input = self.feedforward(data_input)

    if data_input > 0.5:
        return 1
    else:
        return 0

#for array output
def predict_multiple(self, data_input):
    data_input = self.feedforward(data_input)

    result = list()
    for i in data_input:
        if i > 0.95:
            result.append(1)
        else:
            result.append(0)

    return result
```

cross entropy function:

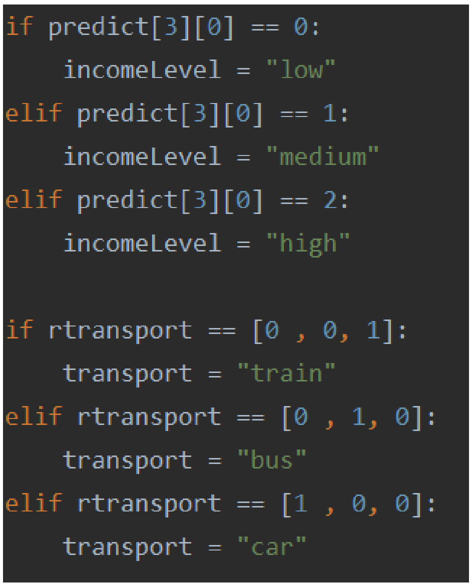


```
#cost += ((a3 - single_output)**2).sum() #distance from target
cost += -np.sum(single_output * np.log(a3) +  (1 - single_output) * np.log(1-a3)) #cross entropy function
```

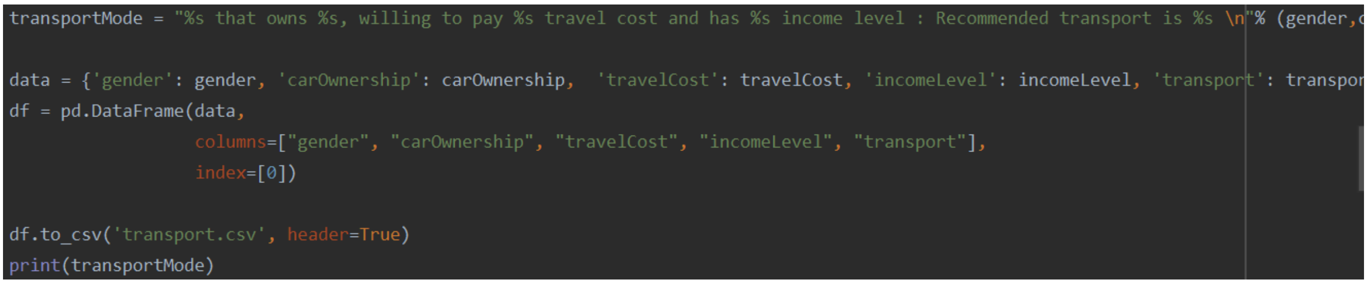
Output to csv:



```
if predict[0][0] == 1:
    gender = "female"
else:
    gender = "male"

if predict[1][0] == 0:
    carOwnership = "no car"
elif predict[1][0] == 1:
    carOwnership = "1 car"
elif predict[1][0] == 2:
    carOwnership = "2 cars"

if predict[2][0] == 0:
    travelCost = "cheap"
elif predict[2][0] == 1:
    travelCost = "standard"
elif predict[2][0] == 2:
    travelCost = "expensive"
```

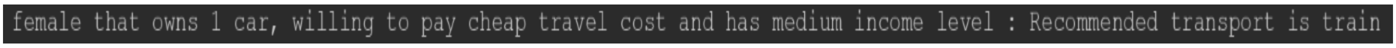


```
if predict[3][0] == 0:
    incomeLevel = "low"
elif predict[3][0] == 1:
    incomeLevel = "medium"
elif predict[3][0] == 2:
    incomeLevel = "high"

if rtransport == [0 , 0, 1]:
    transport = "train"
elif rtransport == [0 , 1, 0]:
    transport = "bus"
elif rtransport == [1 , 0, 0]:
    transport = "car"
```



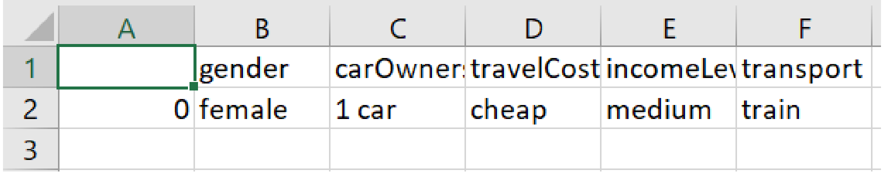
```
transportMode = "%s that owns %s, willing to pay %s travel cost and has %s income level : Recommended transport is %s \n"% (gender,

data = {'gender': gender, 'carOwnership': carOwnership,  'travelCost': travelCost, 'incomeLevel': incomeLevel, 'transport': transpor
df = pd.DataFrame(data,
                  columns=["gender", "carOwnership", "travelCost", "incomeLevel", "transport"],
                  index=[0])

df.to_csv('transport.csv', header=True)
print(transportMode)
```

Output in Jupyter Notebook:



```
female that owns 1 car, willing to pay cheap travel cost and has medium income level : Recommended transport is train
```

Output in csv:

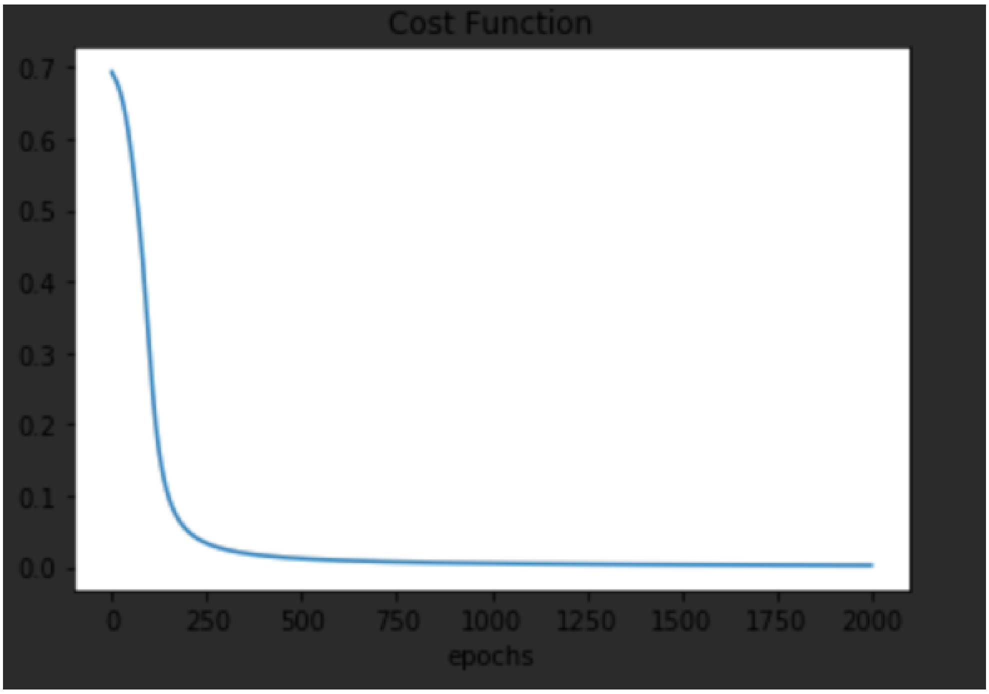


	A	B	C	D	E	F
1		gender	carOwner	travelCost	incomeLe	transport
2	0	female	1 car	cheap	medium	train
3						

Problem 1:

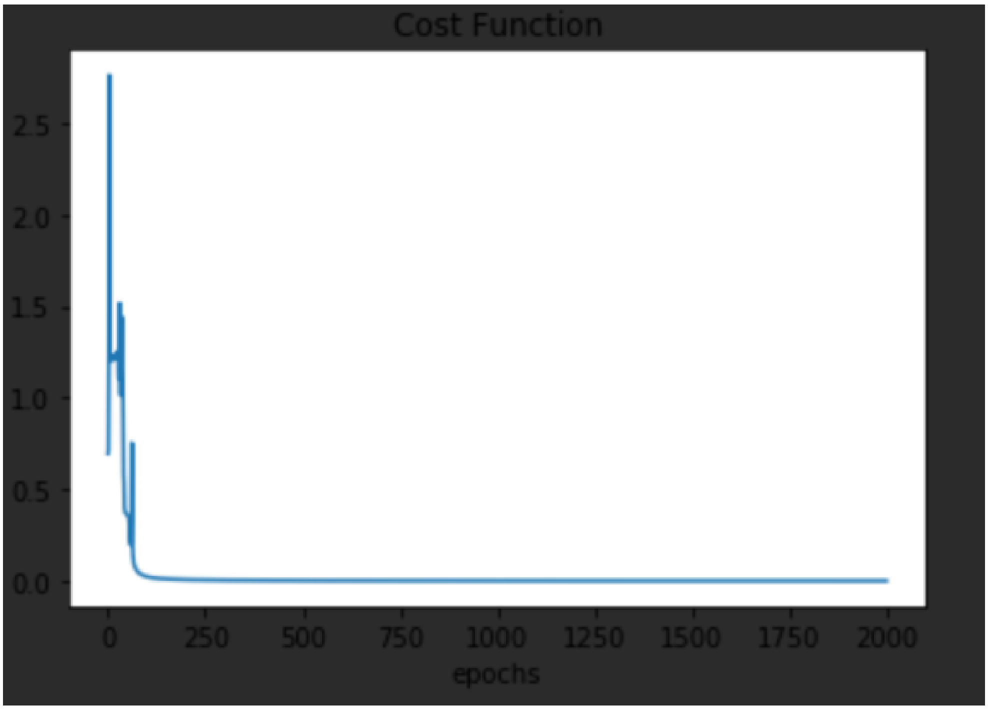
Ephocs: 2000

Eta: 2.0



Ephocs: 2000

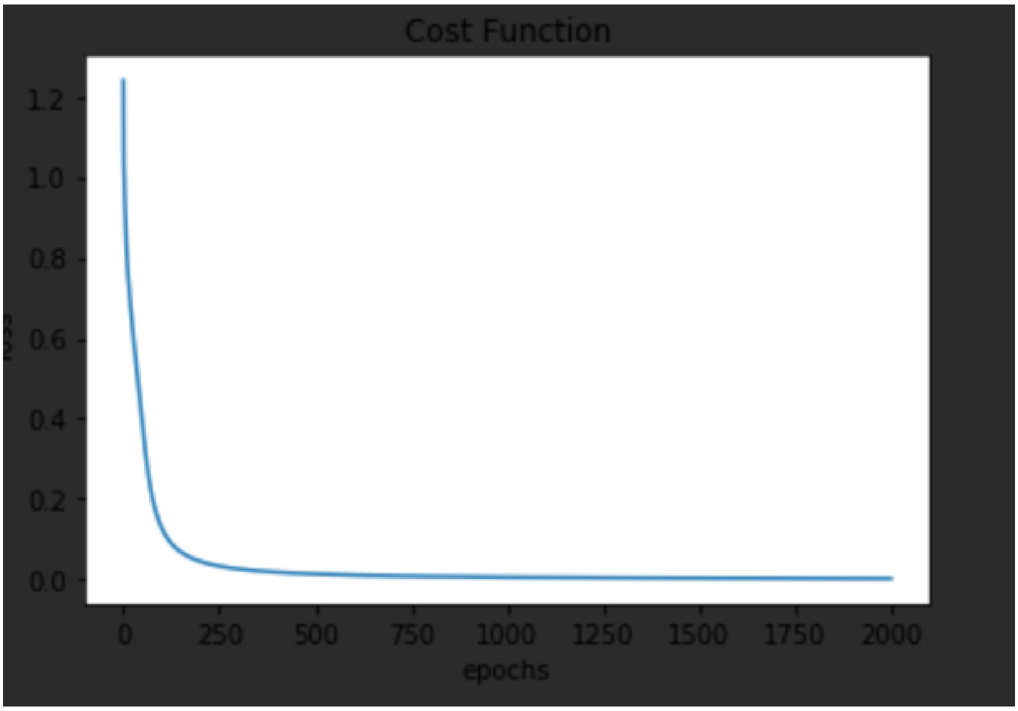
Eta: 11.0



Problem 2:

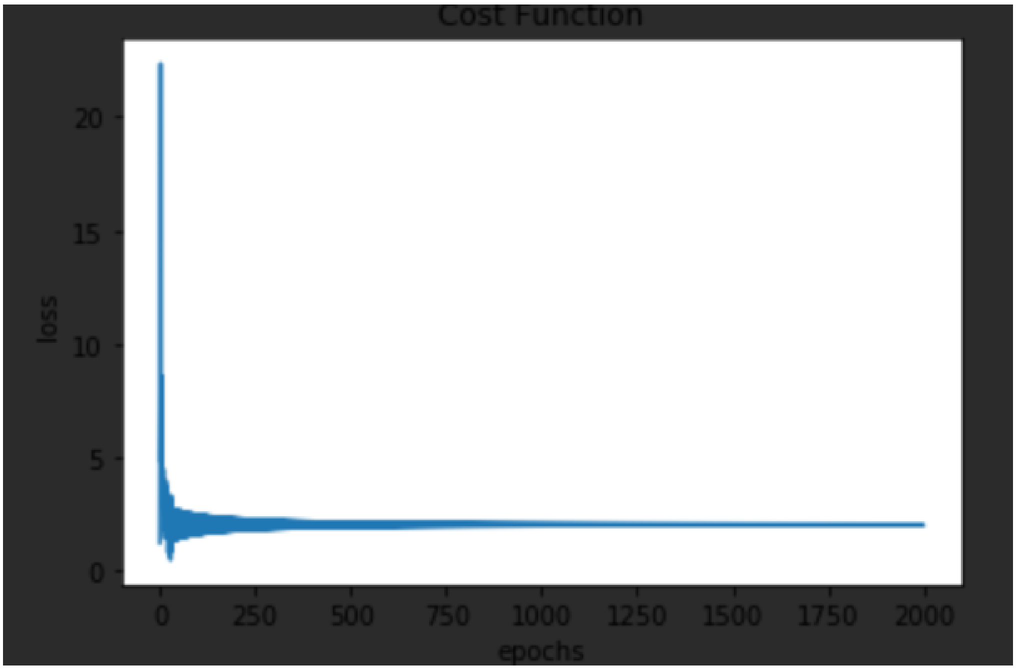
Ephocs: 2000

Eta: 2.0



Ephocs: 2000

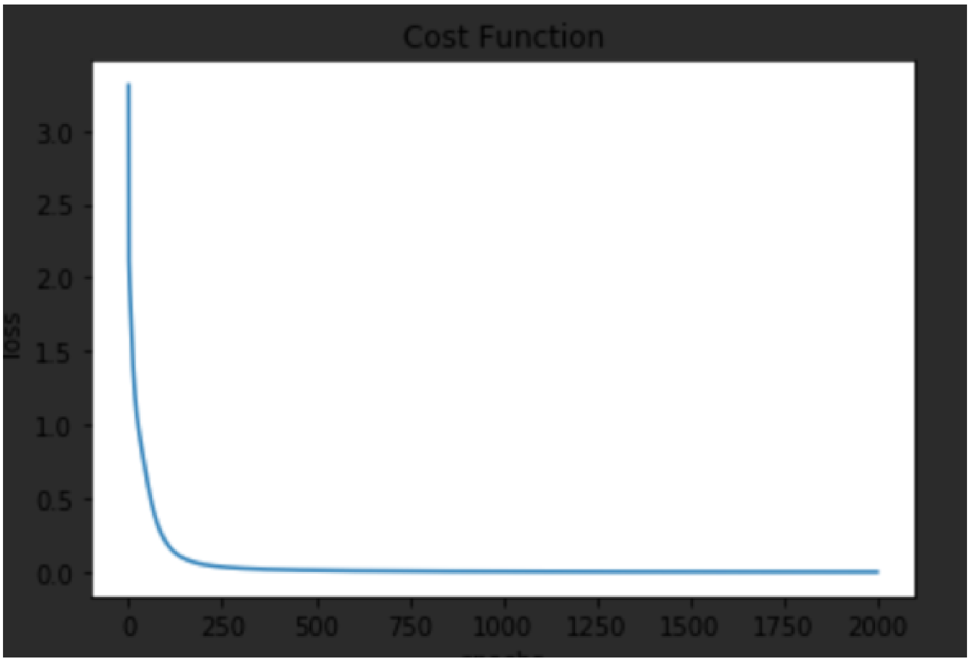
Eta: 50.0



Problem 3:

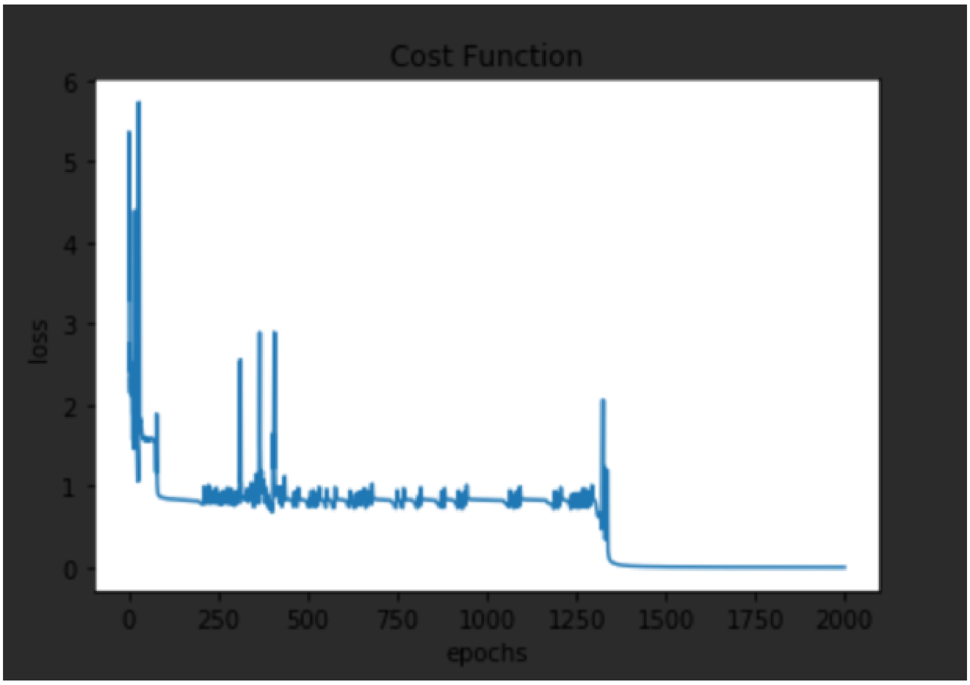
Ephocs: 2000

Eta: 2.0



Ephocs: 2000

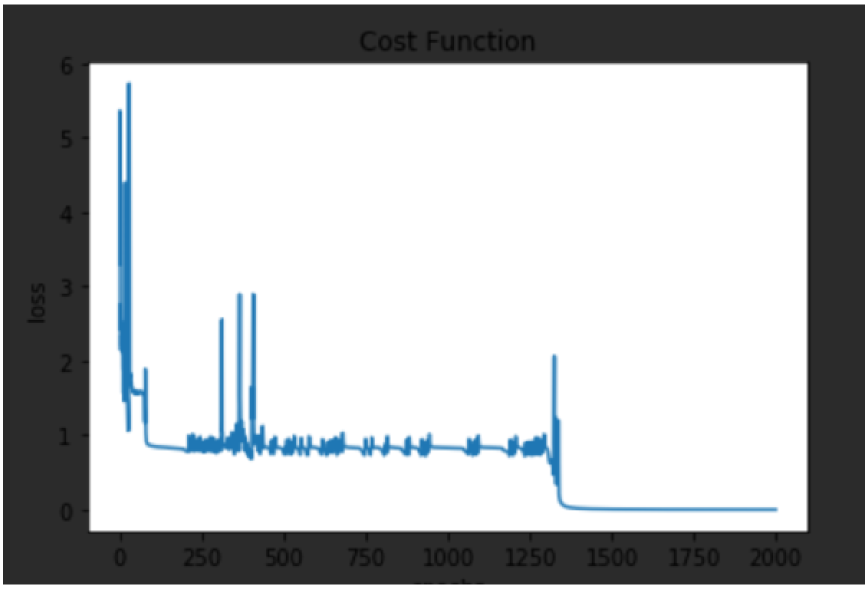
Eta: 11.0



Problem 4:

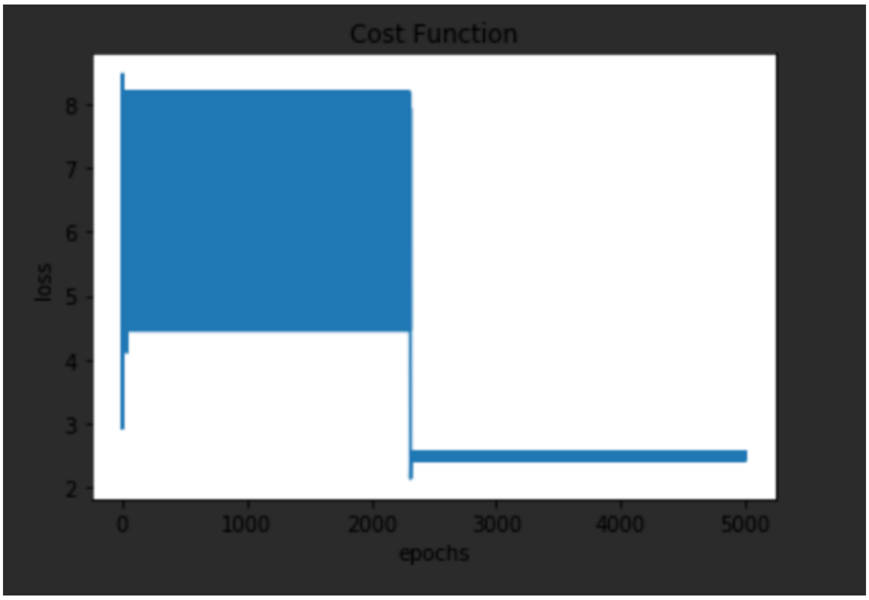
Ephocs: 5000

Eta: 0.05



Ephocs: 5000

Eta: 11.00


Cost Function
loss
epochs