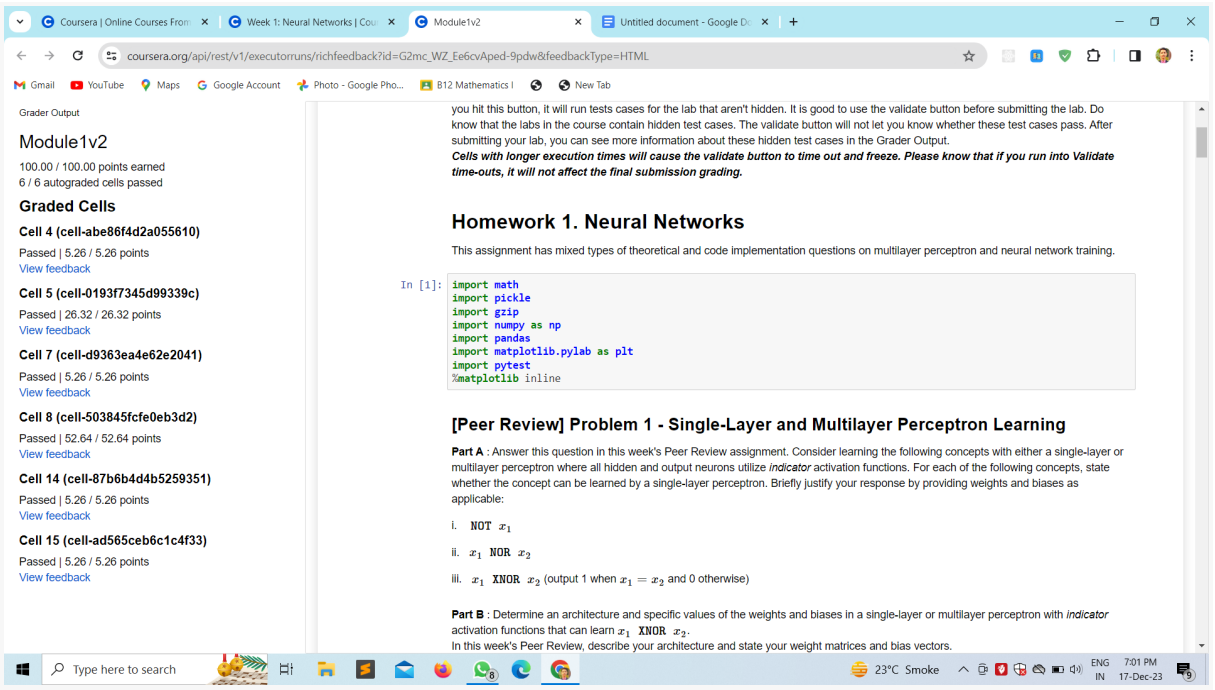
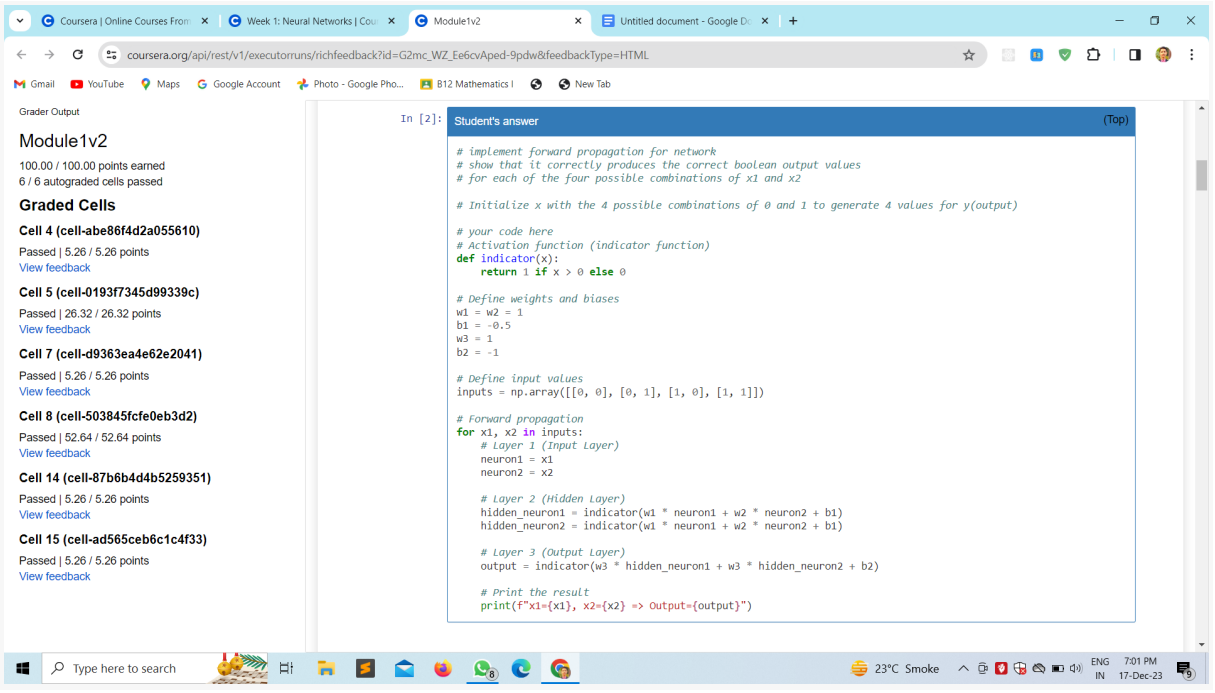
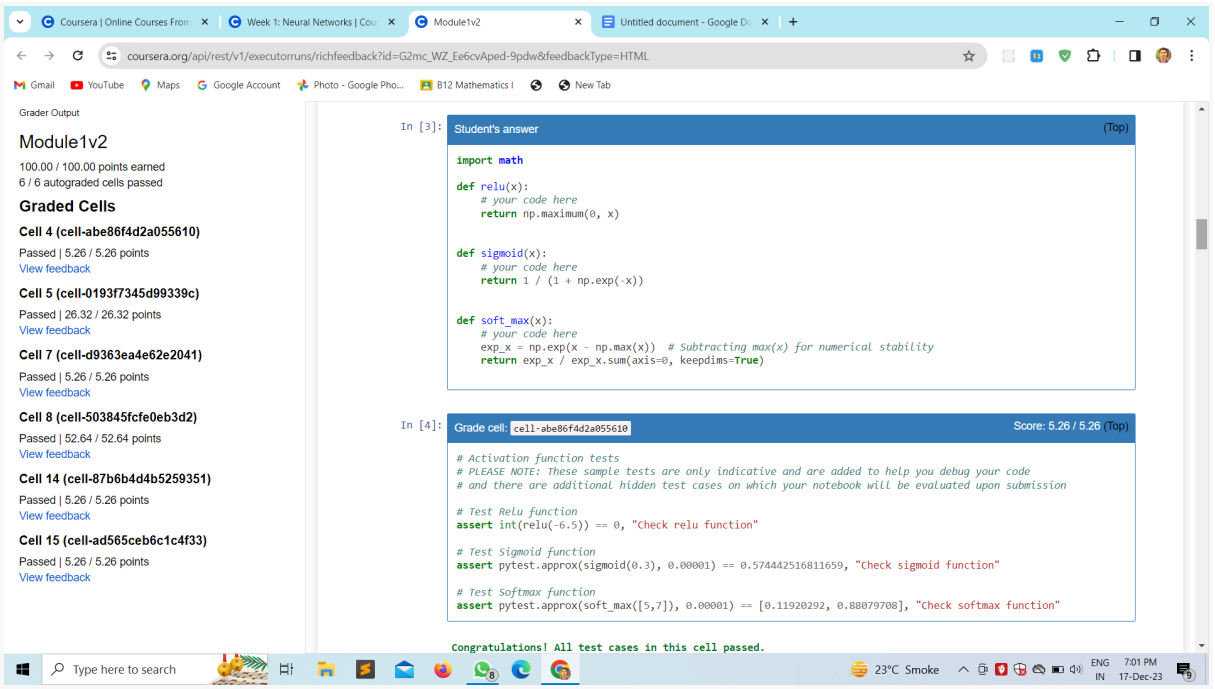

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
you hit this button, it will run tests cases for the lab that aren't hidden. It is good to use the validate button before submitting the lab. Do know that the labs in the course contain hidden test cases. The validate button will not let you know whether these test cases pass. After submitting your lab, you can see more information about these hidden test cases in the Grader Output.
Cells with longer execution times will cause the validate button to time out and freeze. Please know that if you run into Validate time-outs, it will not affect the final submission grading.
Homework 1. Neural Networks
This assignment has mixed types of theoretical and code implementation questions on multilayer perceptron and neural network training.
In [1]:
import math
import pickle
import gzip
import numpy as np
import pandas
import matplotlib.pylab as plt
import pytest
%matplotlib inline
[Peer Review] Problem 1 - Single-Layer and Multilayer Perceptron Learning
Part A : Answer this question in this week's Peer Review assignment. Consider learning the following concepts with either a single-layer or multilayer perceptron where all hidden and output neurons utilize indicator activation functions. For each of the following concepts, state whether the concept can be learned by a single-layer perceptron. Briefly justify your response by providing weights and biases as applicable:
i. NOT x_1
ii. x_1 NOR x_2
iii. x_1 XNOR x_2 (output 1 when $x_1 = x_2$ and 0 otherwise)
Part B : Determine an architecture and specific values of the weights and biases in a single-layer or multilayer perceptron with indicator activation functions that can learn x_1 XNOR x_2.
In this week's Peer Review, describe your architecture and state your weight matrices and bias vectors.
Type here to search
23°C Smoke
ENG
IN
7:01 PM
17-Dec-23


Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
In [2]:
Student's answer
(Top)
# implement forward propagation for network
# show that it correctly produces the correct boolean output values
# for each of the four possible combinations of x1 and x2
# Initialize x with the 4 possible combinations of 0 and 1 to generate 4 values for y(output)
# your code here
# Activation function (indicator function)
def indicator(x):
return 1 if x > 0 else 0
# Define weights and biases
w1 = w2 = 1
b1 = -0.5
w3 = 1
b2 = -1
# Define input values
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
# Forward propagation
for x1, x2 in inputs:
# Layer 1 (Input Layer)
neuron1 = x1
neuron2 = x2
# Layer 2 (Hidden Layer)
hidden_neuron1 = indicator(w1 * neuron1 + w2 * neuron2 + b1)
hidden_neuron2 = indicator(w1 * neuron1 + w2 * neuron2 + b1)
# Layer 3 (Output Layer)
output = indicator(w3 * hidden_neuron1 + w3 * hidden_neuron2 + b2)
# Print the result
print(f"x1={x1}, x2={x2} => Output={output}")
Type here to search
23°C Smoke
ENG
IN
7:01 PM
17-Dec-23



Grader Output

Module1v2

100.00 / 100.00 points earned
6 / 6 autograded cells passed

Graded Cells

Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback

Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback

Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback

Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback

Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback

Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback

In [3]: Student's answer (Top)

```
import math

def relu(x):
    # your code here
    return np.maximum(0, x)


def sigmoid(x):
    # your code here
    return 1 / (1 + np.exp(-x))


def soft_max(x):
    # your code here
    exp_x = np.exp(x - np.max(x))  # Subtracting max(x) for numerical stability
    return exp_x / exp_x.sum(axis=0, keepdims=True)
```

In [4]: Grade cell: cell-abe86f4d2a055610 Score: 5.26 / 5.26 (Top)

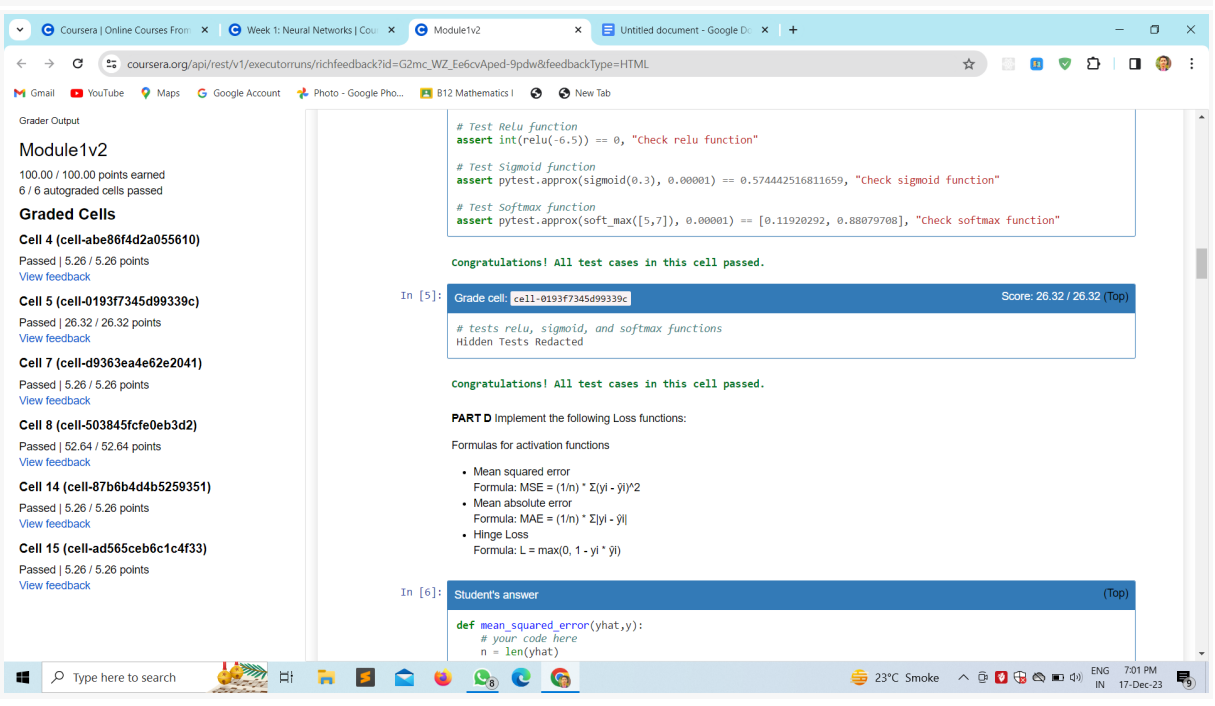
```
# Activation function tests
# PLEASE NOTE: These sample tests are only indicative and are added to help you debug your code
# and there are additional hidden test cases on which your notebook will be evaluated upon submission

# Test Relu function
assert int(relu(-6.5)) == 0, "Check relu function"

# Test Sigmoid function
assert pytest.approx(sigmoid(0.3), 0.00001) == 0.574442516811659, "Check sigmoid function"

# Test Softmax function
assert pytest.approx(soft_max([5,7]), 0.00001) == [0.11920292, 0.88079708], "Check softmax function"
```

Congratulations! All test cases in this cell passed.



Grader Output

Module1v2

100.00 / 100.00 points earned
6 / 6 autograded cells passed

Graded Cells

Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback

Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback

Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback

Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback

Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback

Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback

```
# Test Relu function
assert int(relu(-6.5)) == 0, "Check relu function"

# Test Sigmoid function
assert pytest.approx(sigmoid(0.3), 0.00001) == 0.574442516811659, "Check sigmoid function"

# Test Softmax function
assert pytest.approx(soft_max([5,7]), 0.00001) == [0.11920292, 0.88079708], "Check softmax function"
```

Congratulations! All test cases in this cell passed.

In [5]: Grade cell: cell-0193f7345d99339c Score: 26.32 / 26.32 (Top)

```
# tests relu, sigmoid, and softmax functions
Hidden Tests Redacted
```

Congratulations! All test cases in this cell passed.

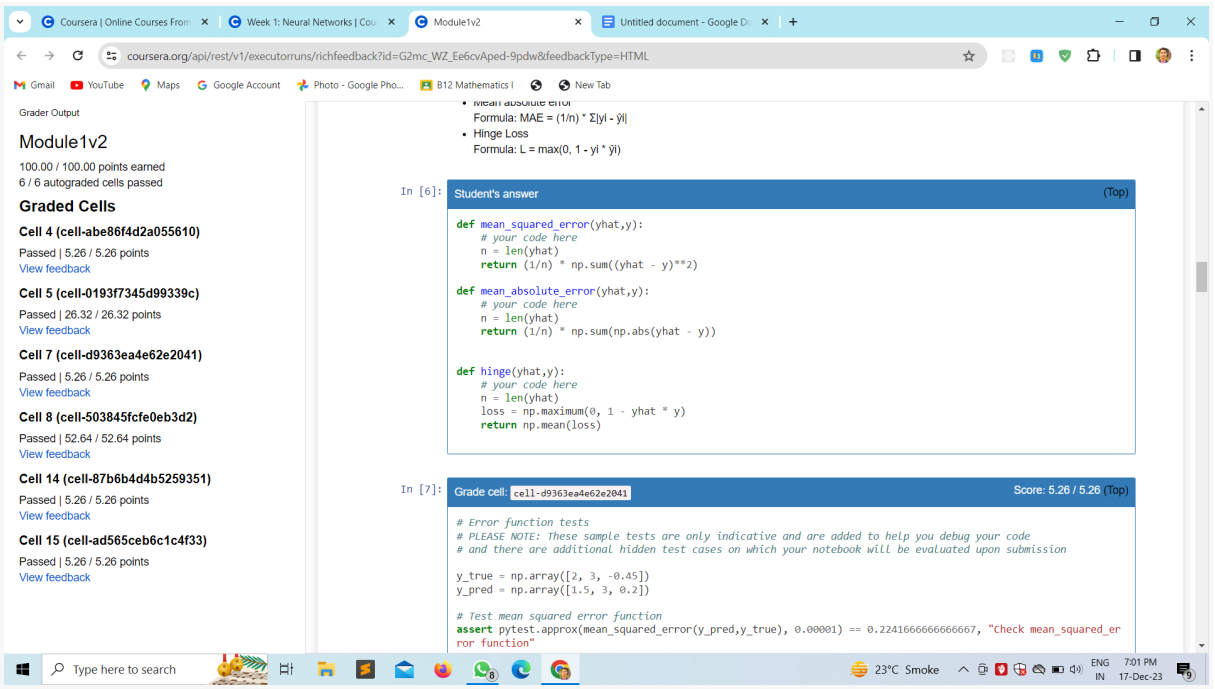
PART D Implement the following Loss functions:

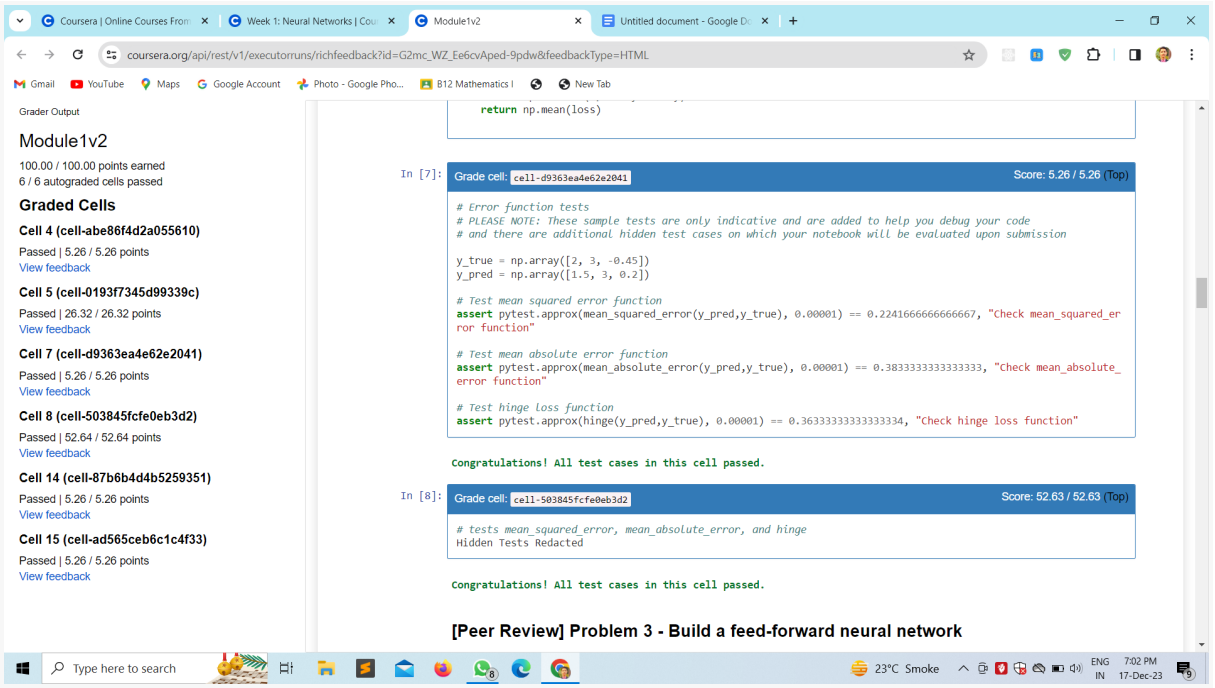
Formulas for activation functions

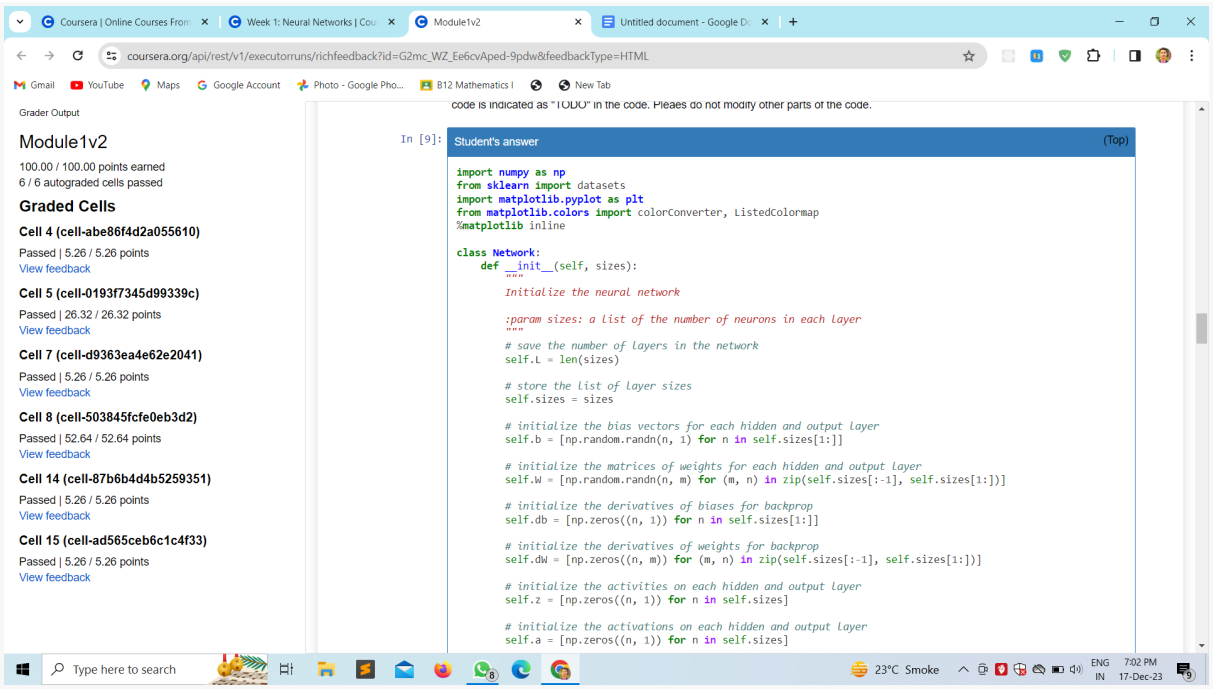
- Mean squared error
  Formula: MSE = (1/n) * Σ(yi - ŷi)^2
- Mean absolute error
  Formula: MAE = (1/n) * Σ|yi - ŷi|
- Hinge Loss
  Formula: L = max(0, 1 - yi * ŷi)

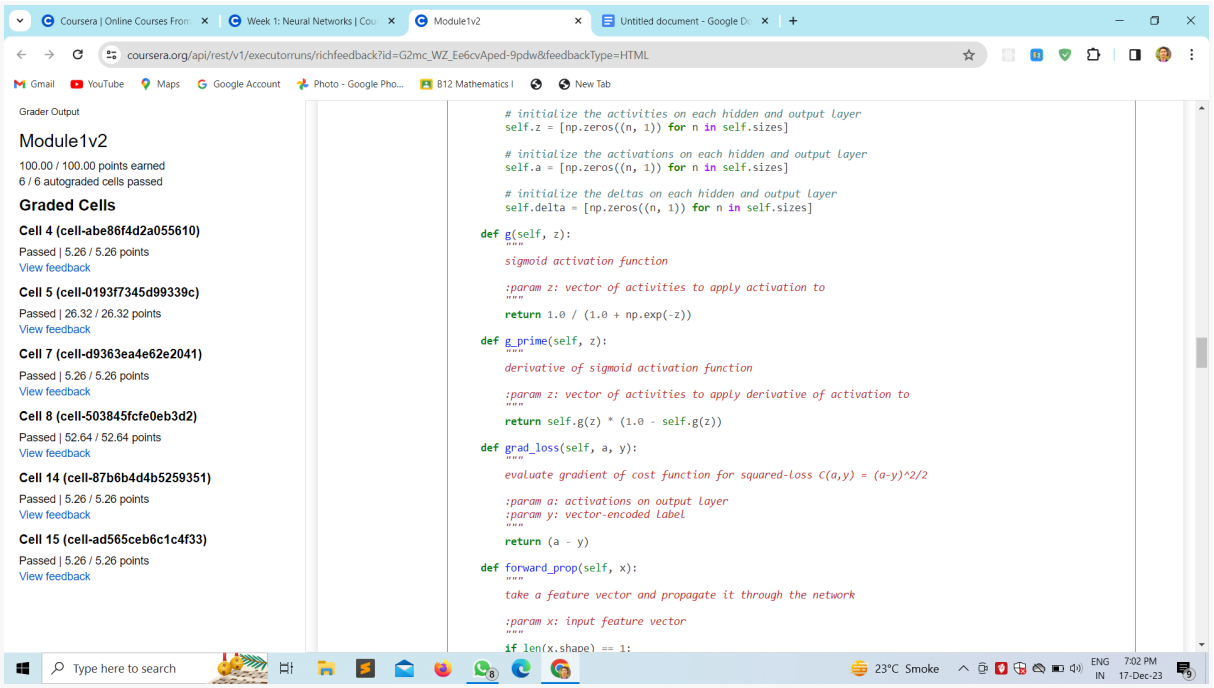
In [6]: Student's answer (Top)

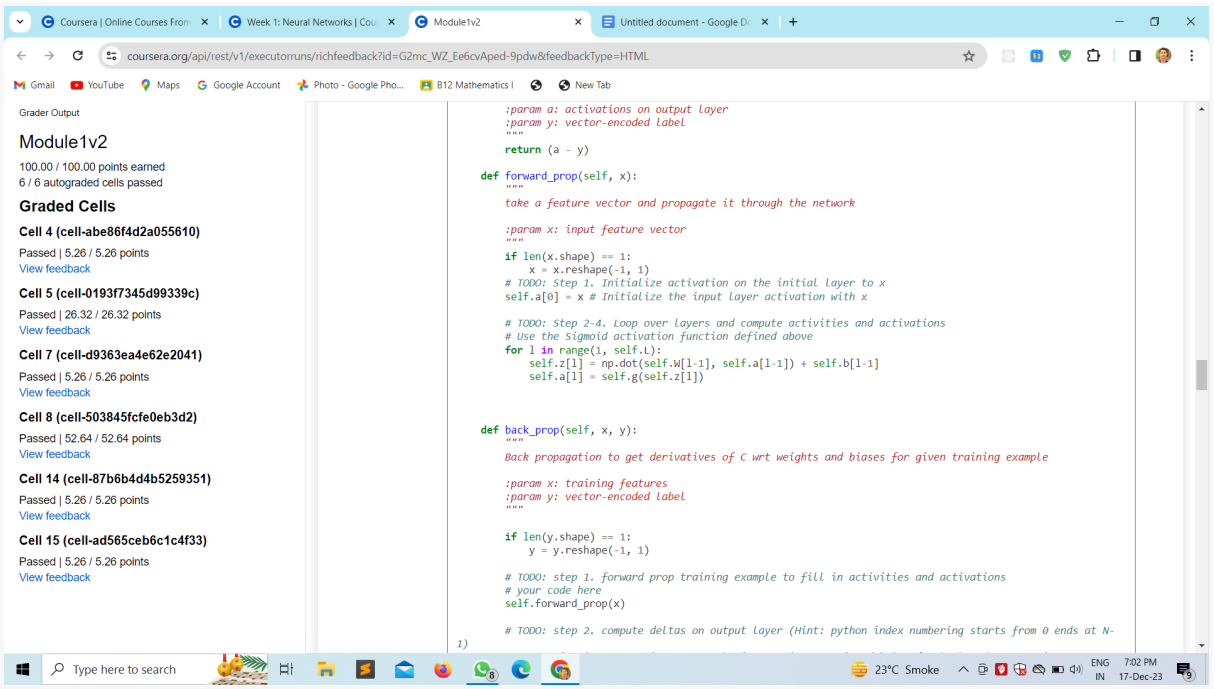
```
def mean_squared_error(yhat,y):
    # your code here
    n = len(yhat)
```

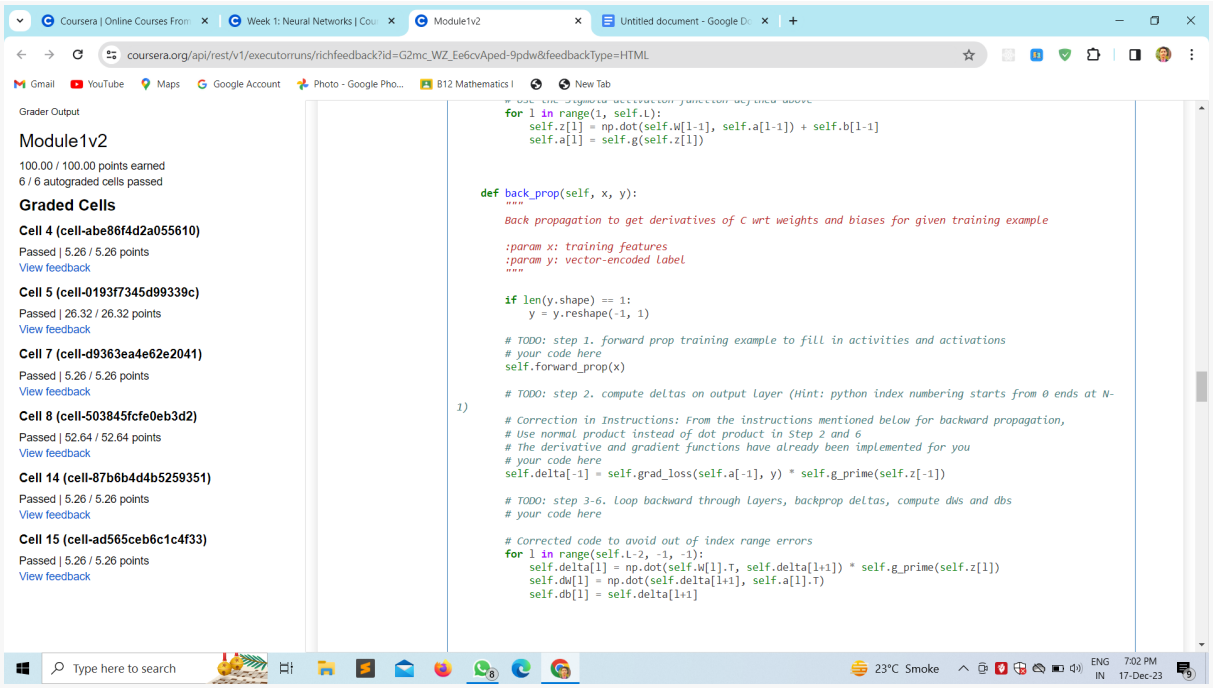

Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
Formula: MAE = (1/n) * Σ|yi - ŷi|
Hinge Loss
Formula: L = max(0, 1 - yi * ŷi)
In [6]:
Student's answer
(Top)
def mean_squared_error(yhat,y):
# your code here
n = len(yhat)
return (1/n) * np.sum((yhat - y)**2)
def mean_absolute_error(yhat,y):
# your code here
n = len(yhat)
return (1/n) * np.sum(np.abs(yhat - y))
def hinge(yhat,y):
# your code here
n = len(yhat)
loss = np.maximum(0, 1 - yhat * y)
return np.mean(loss)
In [7]:
Grade cell: cell-d9363ea4e62e2041
Score: 5.26 / 5.26 (Top)
# Error function tests
# PLEASE NOTE: These sample tests are only indicative and are added to help you debug your code
# and there are additional hidden test cases on which your notebook will be evaluated upon submission
y_true = np.array([2, 3, -0.45])
y_pred = np.array([1.5, 3, 0.2])
# Test mean squared error function
assert pytest.approx(mean_squared_error(y_pred,y_true), 0.00001) == 0.2241666666666667, "Check mean_squared_error function"
Type here to search
23°C Smoke
ENG IN
7:01 PM 17-Dec-23

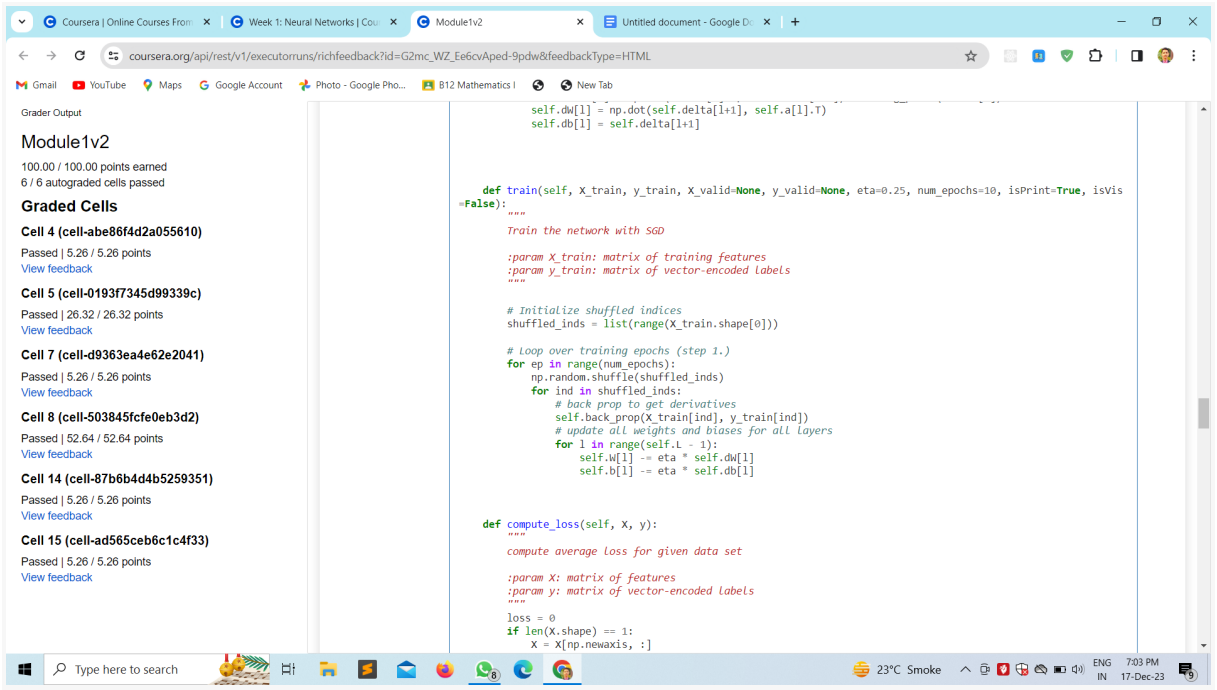

Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
return np.mean(loss)
In [7]:
Grade cell: cell-d9363ea4e62e2041
Score: 5.26 / 5.26 (Top)
# Error function tests
# PLEASE NOTE: These sample tests are only indicative and are added to help you debug your code
# and there are additional hidden test cases on which your notebook will be evaluated upon submission
y_true = np.array([2, 3, -0.45])
y_pred = np.array([1.5, 3, 0.2])
# Test mean squared error function
assert pytest.approx(mean_squared_error(y_pred,y_true), 0.00001) == 0.2241666666666667, "Check mean_squared_error function"
Test mean absolute error function
assert pytest.approx(mean_absolute_error(y_pred,y_true), 0.00001) == 0.3833333333333333, "Check mean_absolute_error function"
Test hinge loss function
assert pytest.approx(hinge(y_pred,y_true), 0.00001) == 0.36333333333333334, "Check hinge loss function"
Congratulations! All test cases in this cell passed.
In [8]:
Grade cell: cell-503845fcfe0eb3d2
Score: 52.63 / 52.63 (Top)
tests mean_squared_error, mean_absolute_error, and hinge
Hidden Tests Redacted
Congratulations! All test cases in this cell passed.
[Peer Review] Problem 3 - Build a feed-forward neural network
Type here to search
23°C Smoke
ENG IN
7:02 PM 17-Dec-23

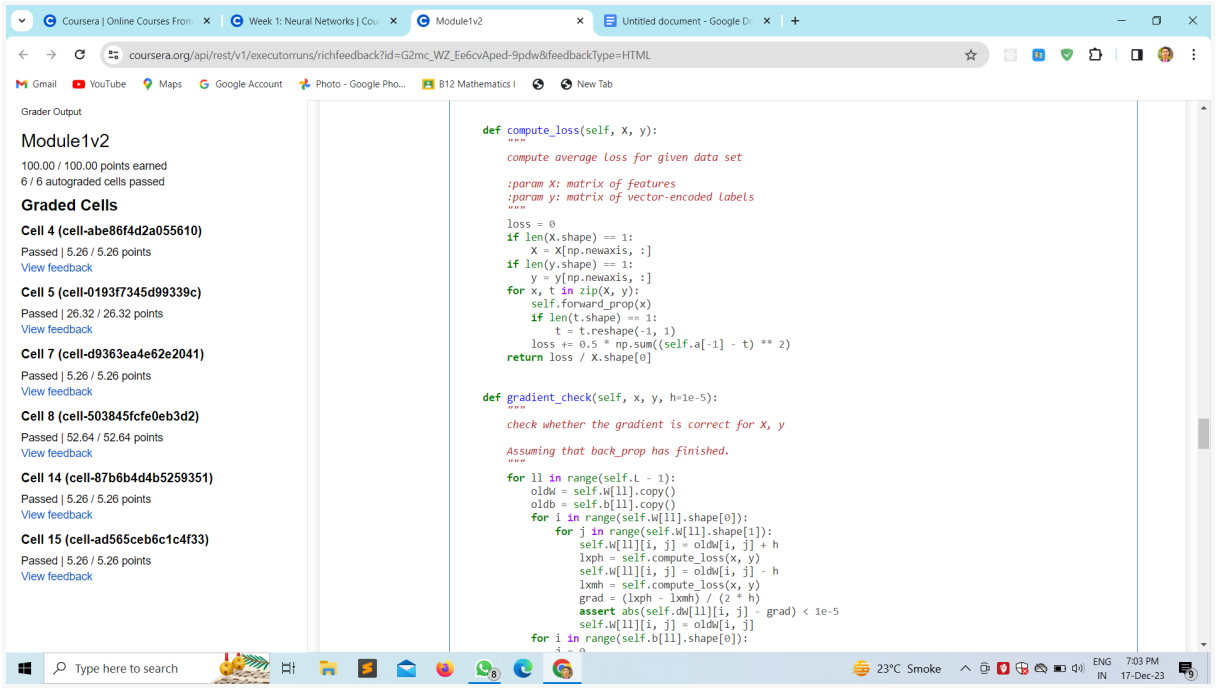

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
code is indicated as "TODO" in the code. Pleaes do not modify other parts of the code.
In [9]:
Student's answer
(Top)
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
from matplotlib.colors import colorConverter, ListedColormap
%matplotlib inline
class Network:
def __init__(self, sizes):
"""
Initialize the neural network
:param sizes: a list of the number of neurons in each layer
"""
# save the number of layers in the network
self.L = len(sizes)
# store the list of layer sizes
self.sizes = sizes
# initialize the bias vectors for each hidden and output layer
self.b = [np.random.randn(n, 1) for n in self.sizes[1:]]
# initialize the matrices of weights for each hidden and output layer
self.W = [np.random.randn(n, m) for (m, n) in zip(self.sizes[:-1], self.sizes[1:])]
# initialize the derivatives of biases for backprop
self.db = [np.zeros((n, 1)) for n in self.sizes[1:]]
# initialize the derivatives of weights for backprop
self.dW = [np.zeros((n, m)) for (m, n) in zip(self.sizes[:-1], self.sizes[1:])]
# initialize the activities on each hidden and output layer
self.z = [np.zeros((n, 1)) for n in self.sizes]
# initialize the activations on each hidden and output layer
self.a = [np.zeros((n, 1)) for n in self.sizes]
Type here to search
23°C Smoke
ENG
IN
7:02 PM
17-Dec-23

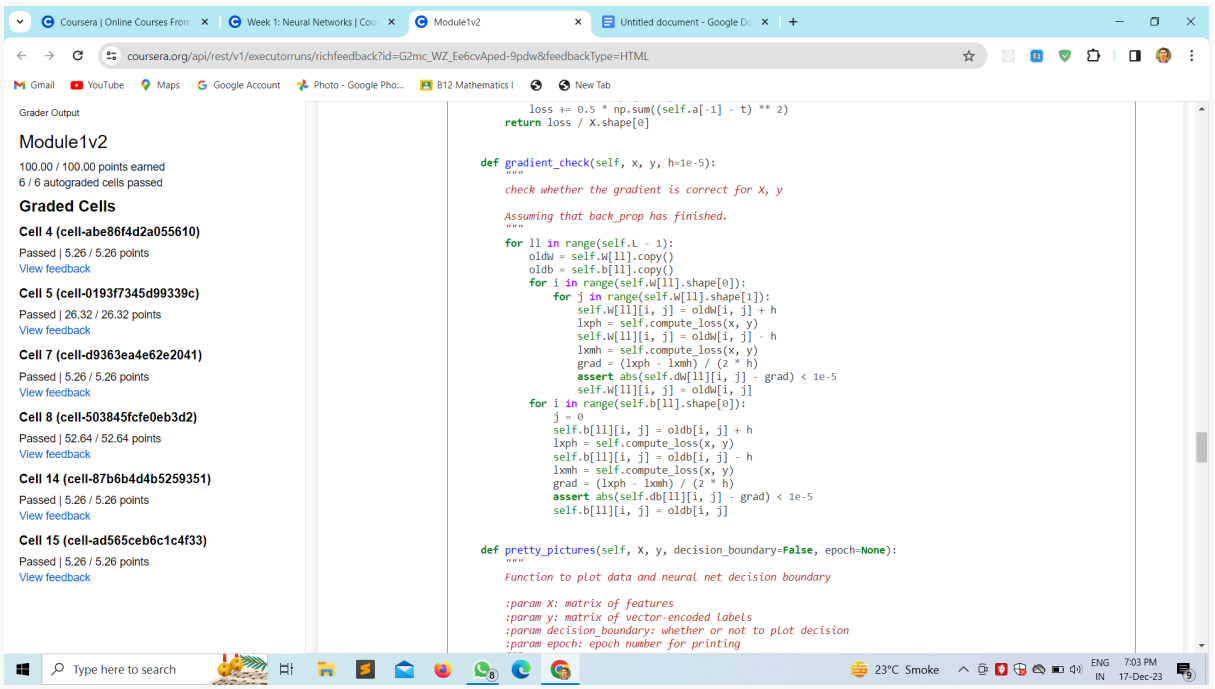

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
# initialize the activities on each hidden and output layer
self.z = [np.zeros((n, 1)) for n in self.sizes]
# initialize the activations on each hidden and output layer
self.a = [np.zeros((n, 1)) for n in self.sizes]
# initialize the deltas on each hidden and output layer
self.delta = [np.zeros((n, 1)) for n in self.sizes]
def g(self, z):
"""
sigmoid activation function
:param z: vector of activities to apply activation to
"""
return 1.0 / (1.0 + np.exp(-z))
def g_prime(self, z):
"""
derivative of sigmoid activation function
:param z: vector of activities to apply derivative of activation to
"""
return self.g(z) * (1.0 - self.g(z))
def grad_loss(self, a, y):
"""
evaluate gradient of cost function for squared-loss C(a,y) = (a-y)^2/2
:param a: activations on output layer
:param y: vector-encoded label
"""
return (a - y)
def forward_prop(self, x):
"""
take a feature vector and propagate it through the network
:param x: input feature vector
"""
if len(x.shape) == 1:
Type here to search
23°C Smoke
ENG
IN
7:02 PM
17-Dec-23

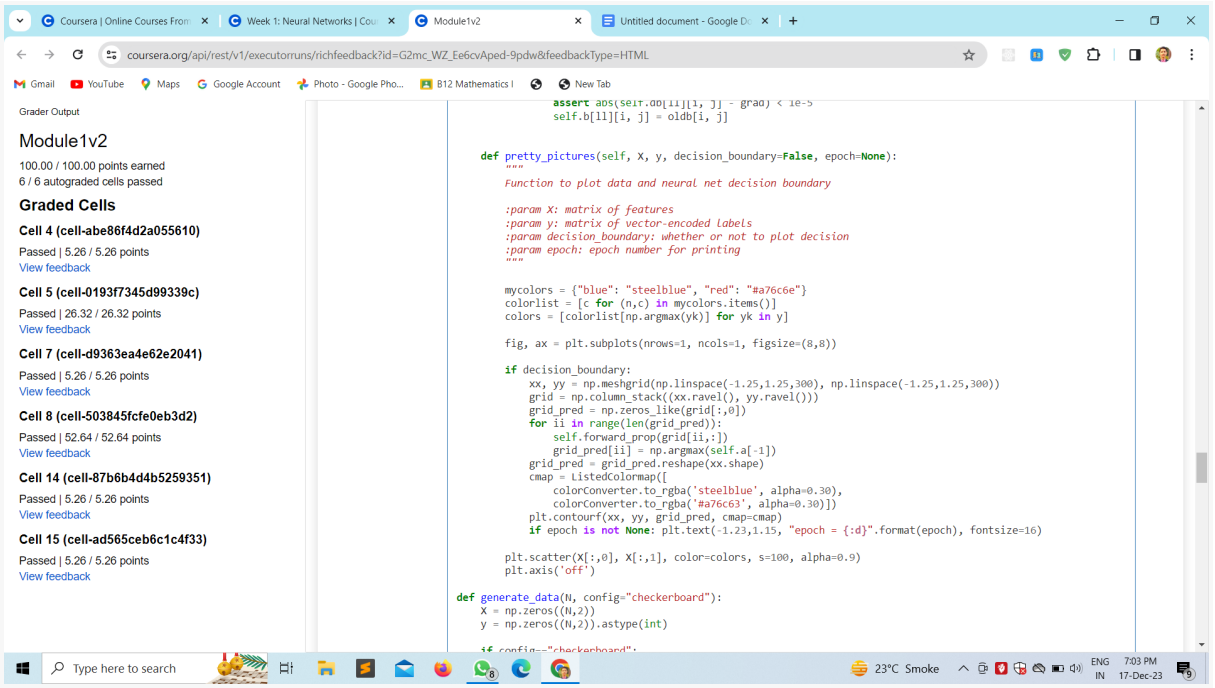

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
:param a: activations on output layer
:param y: vector-encoded label
"""
return (a - y)
def forward_prop(self, x):
"""
take a feature vector and propagate it through the network
:param x: input feature vector
"""
if len(x.shape) == 1:
x = x.reshape(-1, 1)
# TODO: Step 1. Initialize activation on the initial layer to x
self.a[0] = x # Initialize the input layer activation with x
# TODO: Step 2-4. Loop over layers and compute activities and activations
# Use the Sigmoid activation function defined above
for l in range(1, self.L):
self.z[l] = np.dot(self.W[l-1], self.a[l-1]) + self.b[l-1]
self.a[l] = self.g(self.z[l])
def back_prop(self, x, y):
"""
Back propagation to get derivatives of C wrt weights and biases for given training example
:param x: training features
:param y: vector-encoded label
"""
if len(y.shape) == 1:
y = y.reshape(-1, 1)
# TODO: step 1. forward prop training example to fill in activities and activations
# your code here
self.forward_prop(x)
# TODO: step 2. compute deltas on output layer (Hint: python index numbering starts from 0 ends at N-1)
Type here to search
23°C Smoke
ENG IN
7:02 PM 17-Dec-23

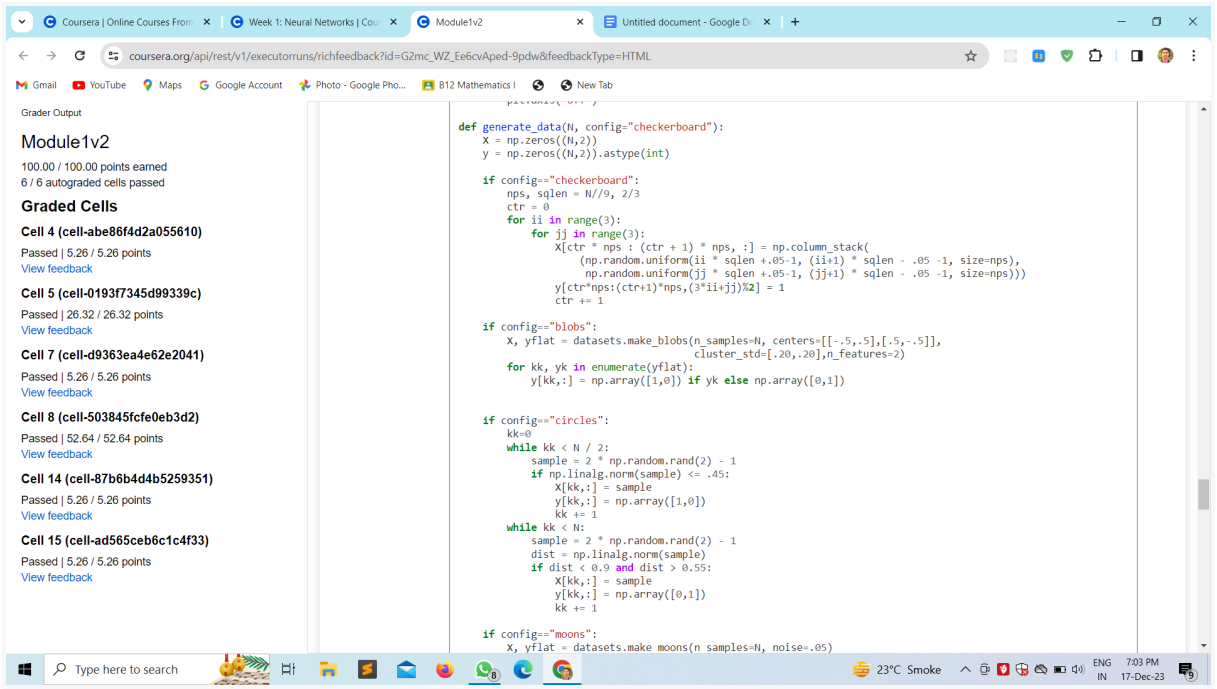

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
for l in range(1, self.L):
self.z[l] = np.dot(self.W[l-1], self.a[l-1]) + self.b[l-1]
self.a[l] = self.g(self.z[l])
def back_prop(self, x, y):
"""
Back propagation to get derivatives of C wrt weights and biases for given training example
:param x: training features
:param y: vector-encoded label
"""
if len(y.shape) == 1:
y = y.reshape(-1, 1)
# TODO: step 1. forward prop training example to fill in activities and activations
# your code here
self.forward_prop(x)
# TODO: step 2. compute deltas on output layer (Hint: python index numbering starts from 0 ends at N-1)
# Correction in Instructions: From the instructions mentioned below for backward propagation,
# Use normal product instead of dot product in Step 2 and 6
# The derivative and gradient functions have already been implemented for you
# your code here
self.delta[-1] = self.grad_loss(self.a[-1], y) * self.g_prime(self.z[-1])
# TODO: step 3-6. Loop backward through layers, backprop deltas, compute dws and dbs
# your code here
# Corrected code to avoid out of index range errors
for l in range(self.L-2, -1, -1):
self.delta[l] = np.dot(self.W[l].T, self.delta[l+1]) * self.g_prime(self.z[l])
self.dW[l] = np.dot(self.delta[l+1], self.a[l].T)
self.db[l] = self.delta[l+1]
Type here to search
23°C Smoke
ENG IN
7:02 PM 17-Dec-23

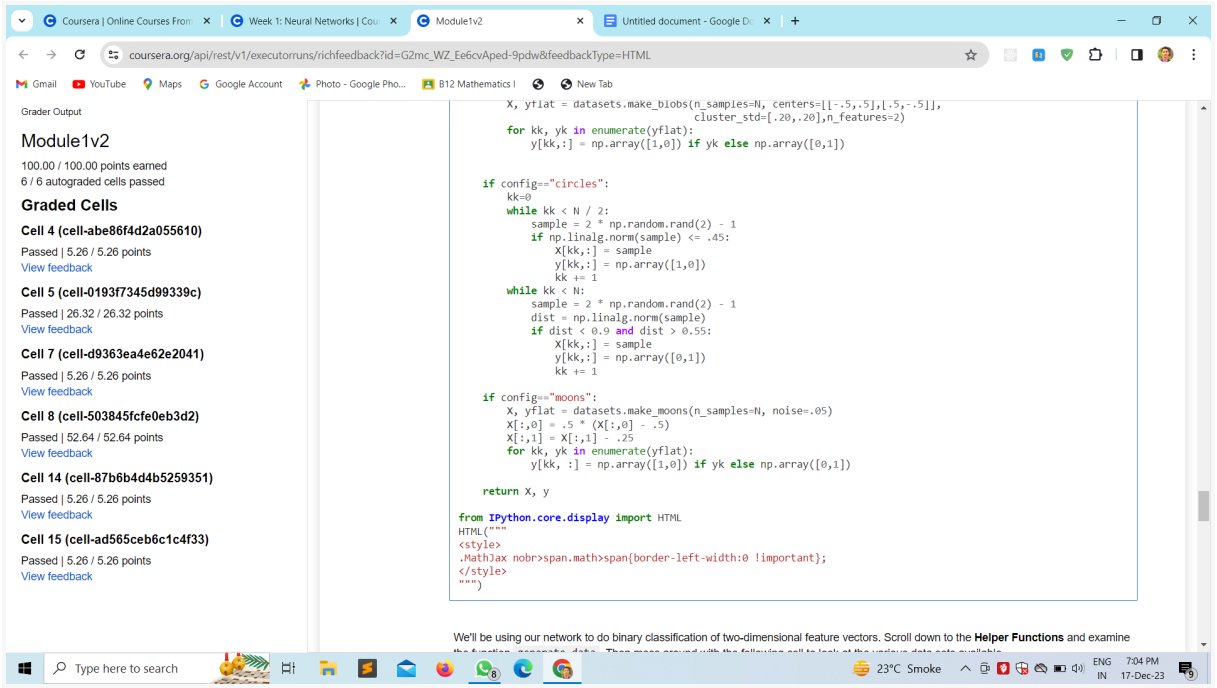

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
self.dw[l] = np.dot(self.delta[l+1], self.a[l].T)
self.db[l] = self.delta[l+1]
def train(self, X_train, y_train, X_valid=None, y_valid=None, eta=0.25, num_epochs=10, isPrint=True, isVis
=False):
"""
Train the network with SGD
:param X_train: matrix of training features
:param y_train: matrix of vector-encoded labels
"""
# Initialize shuffled indices
shuffled_inds = list(range(X_train.shape[0]))
# Loop over training epochs (step 1.)
for ep in range(num_epochs):
np.random.shuffle(shuffled_inds)
for ind in shuffled_inds:
# back prop to get derivatives
self.back_prop(X_train[ind], y_train[ind])
# update all weights and biases for all layers
for l in range(self.L - 1):
self.W[l] -= eta * self.dW[l]
self.b[l] -= eta * self.db[l]
def compute_loss(self, X, y):
"""
compute average loss for given data set
:param X: matrix of features
:param y: matrix of vector-encoded labels
"""
loss = 0
if len(X.shape) == 1:
X = X[np.newaxis, :]
Type here to search
23°C Smoke
ENG
IN
7:03 PM
17-Dec-23

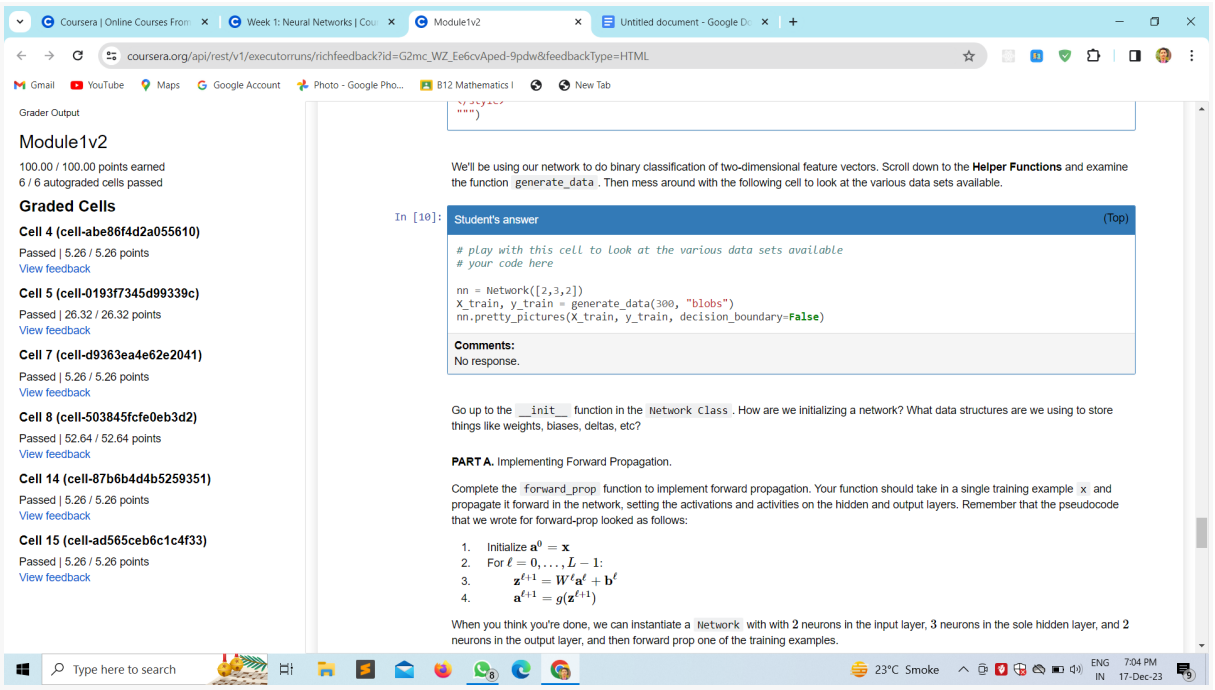

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
def compute_loss(self, X, y):
"""
compute average loss for given data set
:param X: matrix of features
:param y: matrix of vector-encoded labels
"""
loss = 0
if len(X.shape) == 1:
X = X[np.newaxis, :]
if len(y.shape) == 1:
y = y[np.newaxis, :]
for x, t in zip(X, y):
self.forward_prop(x)
if len(t.shape) == 1:
t = t.reshape(-1, 1)
loss += 0.5 * np.sum((self.a[-1] - t) ** 2)
return loss / X.shape[0]
def gradient_check(self, x, y, h=1e-5):
"""
check whether the gradient is correct for X, y
Assuming that back_prop has finished.
"""
for ll in range(self.L - 1):
oldW = self.W[ll].copy()
oldb = self.b[ll].copy()
for i in range(self.W[ll].shape[0]):
for j in range(self.W[ll].shape[1]):
self.W[ll][i, j] = oldW[i, j] + h
lxph = self.compute_loss(x, y)
self.W[ll][i, j] = oldW[i, j] - h
lxmh = self.compute_loss(x, y)
grad = (lxph - lxmh) / (2 * h)
assert abs(self.dW[ll][i, j] - grad) < 1e-5
self.W[ll][i, j] = oldW[i, j]
for i in range(self.b[ll].shape[0]):
Type here to search
23°C Smoke
ENG
IN
7:03 PM
17-Dec-23

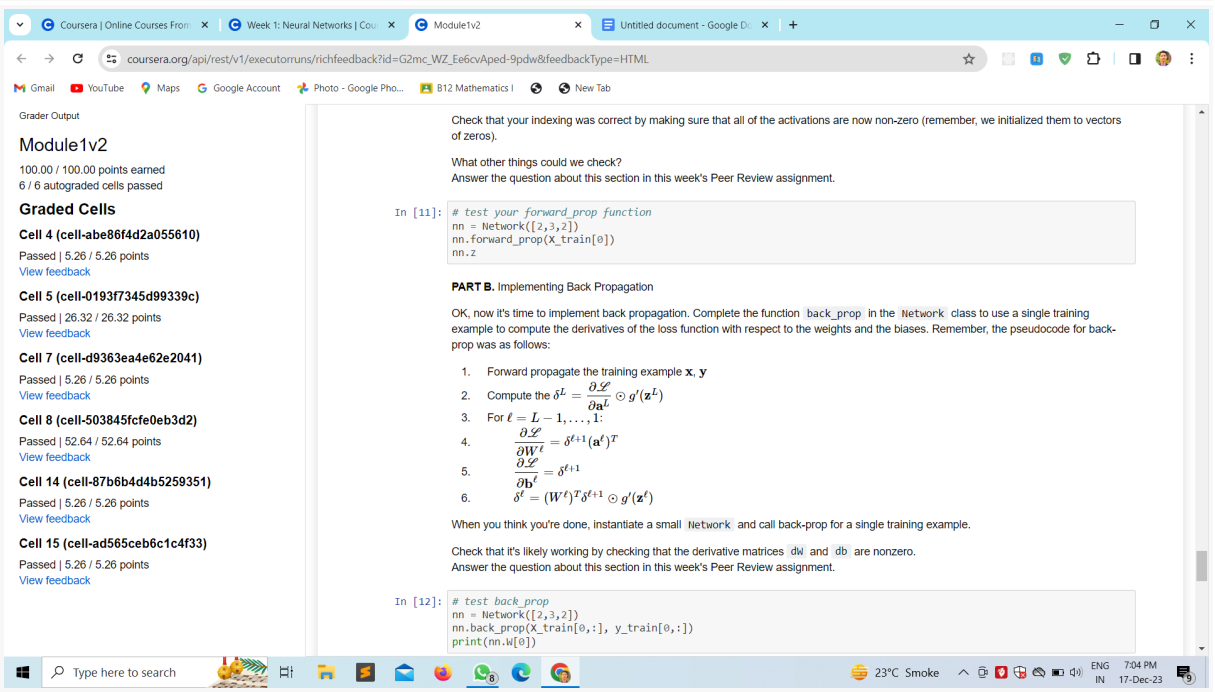

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
loss += 0.5 * np.sum((self.a[-1] - t) ** 2)
return loss / X.shape[0]
def gradient_check(self, x, y, h=1e-5):
"""
check whether the gradient is correct for X, y
Assuming that back_prop has finished.
"""
for ll in range(self.L - 1):
oldW = self.W[ll].copy()
oldb = self.b[ll].copy()
for i in range(self.W[ll].shape[0]):
for j in range(self.W[ll].shape[1]):
self.W[ll][i, j] = oldW[i, j] + h
lxph = self.compute_loss(x, y)
self.W[ll][i, j] = oldW[i, j] - h
lxmh = self.compute_loss(x, y)
grad = (lxph - lxmh) / (2 * h)
assert abs(self.dW[ll][i, j] - grad) < 1e-5
self.W[ll][i, j] = oldW[i, j]
for i in range(self.b[ll].shape[0]):
j = 0
self.b[ll][i, j] = oldb[i, j] + h
lxph = self.compute_loss(x, y)
self.b[ll][i, j] = oldb[i, j] - h
lxmh = self.compute_loss(x, y)
grad = (lxph - lxmh) / (2 * h)
assert abs(self.db[ll][i, j] - grad) < 1e-5
self.b[ll][i, j] = oldb[i, j]
def pretty_pictures(self, X, y, decision_boundary=False, epoch=None):
"""
Function to plot data and neural net decision boundary
:param X: matrix of features
:param y: matrix of vector-encoded labels
:param decision_boundary: whether or not to plot decision
:param epoch: epoch number for printing
Type here to search
23°C Smoke
ENG
IN
7:03 PM
17-Dec-23

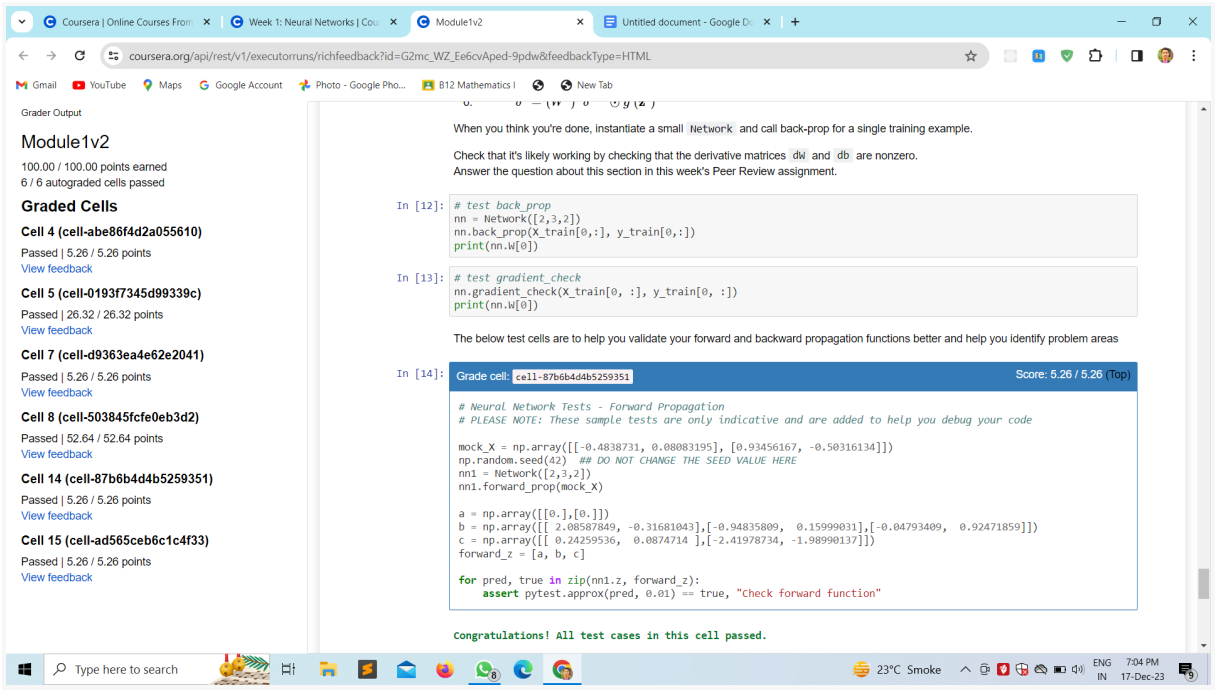

Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
assert abs(self.dW[ll][i, j] - grad) < 1e-5
self.b[ll][i, j] = oldb[i, j]
def pretty_pictures(self, X, y, decision_boundary=False, epoch=None):
"""
Function to plot data and neural net decision boundary
:param X: matrix of features
:param y: matrix of vector-encoded labels
:param decision_boundary: whether or not to plot decision
:param epoch: epoch number for printing
"""
mycolors = {"blue": "steelblue", "red": "#a76c6e"}
colorlist = [c for (n,c) in mycolors.items()]
colors = [colorlist[np.argmax(yk)] for yk in y]
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(8,8))
if decision_boundary:
xx, yy = np.meshgrid(np.linspace(-1.25,1.25,300), np.linspace(-1.25,1.25,300))
grid = np.column_stack((xx.ravel(), yy.ravel()))
grid_pred = np.zeros_like(grid[:,0])
for ii in range(len(grid_pred)):
self.forward_prop(grid[ii,:])
grid_pred[ii] = np.argmax(self.a[-1])
grid_pred = grid_pred.reshape(xx.shape)
cmap = ListedColormap([
colorConverter.to_rgba('steelblue', alpha=0.30),
colorConverter.to_rgba('#a76c63', alpha=0.30)])
plt.contourf(xx, yy, grid_pred, cmap=cmap)
if epoch is not None: plt.text(-1.23,1.15, "epoch = {:d}".format(epoch), fontsize=16)
plt.scatter(X[:,0], X[:,1], color=colors, s=100, alpha=0.9)
plt.axis('off')
def generate_data(N, config="checkerboard"):
X = np.zeros((N,2))
y = np.zeros((N,2)).astype(int)
Type here to search
23°C Smoke
ENG
IN
7:03 PM
17-Dec-23


Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
def generate_data(N, config="checkerboard"):
X = np.zeros((N,2))
y = np.zeros((N,2)).astype(int)
if config=="checkerboard":
nps, sqlen = N//9, 2/3
ctr = 0
for ii in range(3):
for jj in range(3):
X[ctr * nps : (ctr + 1) * nps, :] = np.column_stack(
(np.random.uniform(ii * sqlen +.05-1, (ii+1) * sqlen - .05 -1, size=nps),
np.random.uniform(jj * sqlen +.05-1, (jj+1) * sqlen - .05 -1, size=nps)))
y[ctr*nps:(ctr+1)*nps,(3*ii+jj)%2] = 1
ctr += 1
if config=="blobs":
X, yflat = datasets.make_blobs(n_samples=N, centers=[[-.5,.5],[.5,-.5]],
cluster_std=[.20,.20],n_features=2)
for kk, yk in enumerate(yflat):
y[kk,:] = np.array([1,0]) if yk else np.array([0,1])
if config=="circles":
kk=0
while kk < N / 2:
sample = 2 * np.random.rand(2) - 1
if np.linalg.norm(sample) <= .45:
X[kk,:] = sample
y[kk,:] = np.array([1,0])
kk += 1
while kk < N:
sample = 2 * np.random.rand(2) - 1
dist = np.linalg.norm(sample)
if dist < 0.9 and dist > 0.55:
X[kk,:] = sample
y[kk,:] = np.array([0,1])
kk += 1
if config=="moons":
X, yflat = datasets.make_moons(n_samples=N, noise=.05)
Type here to search
23°C Smoke
ENG
IN
7:03 PM
17-Dec-23


Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
X, yflat = datasets.make_blobs(n_samples=N, centers=[[-.5,.5],[.5,-.5]],
cluster_std=[.20,.20],n_features=2)
for kk, yk in enumerate(yflat):
y[kk,:] = np.array([1,0]) if yk else np.array([0,1])
if config=="circles":
kk=0
while kk < N / 2:
sample = 2 * np.random.rand(2) - 1
if np.linalg.norm(sample) <= .45:
X[kk,:] = sample
y[kk,:] = np.array([1,0])
kk += 1
while kk < N:
sample = 2 * np.random.rand(2) - 1
dist = np.linalg.norm(sample)
if dist < 0.9 and dist > 0.55:
X[kk,:] = sample
y[kk,:] = np.array([0,1])
kk += 1
if config=="moons":
X, yflat = datasets.make_moons(n_samples=N, noise=.05)
X[:,0] = .5 * (X[:,0] - .5)
X[:,1] = X[:,1] - .25
for kk, yk in enumerate(yflat):
y[kk, :] = np.array([1,0]) if yk else np.array([0,1])
return X, y
from IPython.core.display import HTML
HTML("""
<style>
.MathJax nobr>span.math>span{border-left-width:0 !important};
</style>
""")
We'll be using our network to do binary classification of two-dimensional feature vectors. Scroll down to the Helper Functions and examine
Type here to search
23°C Smoke
ENG
IN
7:04 PM
17-Dec-23


Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
"""
We'll be using our network to do binary classification of two-dimensional feature vectors. Scroll down to the Helper Functions and examine the function generate_data. Then mess around with the following cell to look at the various data sets available.
In [10]:
Student's answer
(Top)
# play with this cell to look at the various data sets available
# your code here
nn = Network([2,3,2])
X_train, y_train = generate_data(300, "blobs")
nn.pretty_pictures(X_train, y_train, decision_boundary=False)
Comments:
No response.
Go up to the __init__ function in the Network Class. How are we initializing a network? What data structures are we using to store things like weights, biases, deltas, etc?
PART A. Implementing Forward Propagation.
Complete the forward_prop function to implement forward propagation. Your function should take in a single training example x and propagate it forward in the network, setting the activations and activities on the hidden and output layers. Remember that the pseudocode that we wrote for forward-prop looked as follows:
1. Initialize $\mathbf{a}^0 = \mathbf{x}$
2. For $\ell = 0, \ldots, L-1$:
3. $\mathbf{z}^{\ell+1} = W^\ell \mathbf{a}^\ell + \mathbf{b}^\ell$
4. $\mathbf{a}^{\ell+1} = g(\mathbf{z}^{\ell+1})$
When you think you're done, we can instantiate a Network with with 2 neurons in the input layer, 3 neurons in the sole hidden layer, and 2 neurons in the output layer, and then forward prop one of the training examples.
Type here to search
23°C Smoke
ENG IN
7:04 PM 17-Dec-23


Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
Check that your indexing was correct by making sure that all of the activations are now non-zero (remember, we initialized them to vectors of zeros).
What other things could we check?
Answer the question about this section in this week's Peer Review assignment.
In [11]:
# test your forward_prop function
nn = Network([2,3,2])
nn.forward_prop(X_train[0])
nn.z
PART B. Implementing Back Propagation
OK, now it's time to implement back propagation. Complete the function back_prop in the Network class to use a single training example to compute the derivatives of the loss function with respect to the weights and the biases. Remember, the pseudocode for back-prop was as follows:
1. Forward propagate the training example $\mathbf{x}, \mathbf{y}$
2. Compute the $\delta^L = \frac{\partial \mathscr{L}}{\partial \mathbf{a}^L} \odot g'(\mathbf{z}^L)$
3. For $\ell = L-1, \ldots, 1$:
4. $\frac{\partial \mathscr{L}}{\partial W^\ell} = \delta^{\ell+1} (\mathbf{a}^\ell)^T$
5. $\frac{\partial \mathscr{L}}{\partial \mathbf{b}^\ell} = \delta^{\ell+1}$
6. $\delta^\ell = (W^\ell)^T \delta^{\ell+1} \odot g'(\mathbf{z}^\ell)$
When you think you're done, instantiate a small Network and call back-prop for a single training example.
Check that it's likely working by checking that the derivative matrices dW and db are nonzero.
Answer the question about this section in this week's Peer Review assignment.
In [12]:
# test back_prop
nn = Network([2,3,2])
nn.back_prop(X_train[0,:], y_train[0,:])
print(nn.w[0])
Type here to search
23°C Smoke
ENG IN
7:04 PM 17-Dec-23


Coursera | Online Courses From
Week 1: Neural Networks | Cou
Module1v2
Untitled document - Google Do
coursera.org/api/rest/v1/executorruns/richfeedback?id=G2mc_WZ_Ee6cvAped-9pdw&feedbackType=HTML
Gmail
YouTube
Maps
Google Account
Photo - Google Pho...
B12 Mathematics I
New Tab
Grader Output
Module1v2
100.00 / 100.00 points earned
6 / 6 autograded cells passed
Graded Cells
Cell 4 (cell-abe86f4d2a055610)
Passed | 5.26 / 5.26 points
View feedback
Cell 5 (cell-0193f7345d99339c)
Passed | 26.32 / 26.32 points
View feedback
Cell 7 (cell-d9363ea4e62e2041)
Passed | 5.26 / 5.26 points
View feedback
Cell 8 (cell-503845fcfe0eb3d2)
Passed | 52.64 / 52.64 points
View feedback
Cell 14 (cell-87b6b4d4b5259351)
Passed | 5.26 / 5.26 points
View feedback
Cell 15 (cell-ad565ceb6c1c4f33)
Passed | 5.26 / 5.26 points
View feedback
When you think you're done, instantiate a small Network and call back-prop for a single training example.
Check that it's likely working by checking that the derivative matrices dW and db are nonzero.
Answer the question about this section in this week's Peer Review assignment.
In [12]:
# test back_prop
nn = Network([2,3,2])
nn.back_prop(X_train[0,:], y_train[0,:])
print(nn.W[0])
In [13]:
# test gradient_check
nn.gradient_check(X_train[0, :], y_train[0, :])
print(nn.W[0])
The below test cells are to help you validate your forward and backward propagation functions better and help you identify problem areas
In [14]:
Grade cell: cell-87b6b4d4b5259351
Score: 5.26 / 5.26 (Top)
# Neural Network Tests - Forward Propagation
# PLEASE NOTE: These sample tests are only indicative and are added to help you debug your code
mock_X = np.array([[-0.4838731, 0.08083195], [0.93456167, -0.50316134]])
np.random.seed(42) ## DO NOT CHANGE THE SEED VALUE HERE
nn1 = Network([2,3,2])
nn1.forward_prop(mock_X)
a = np.array([[0.],[0.]])
b = np.array([[2.08587849, -0.31681043],[-0.94835809, 0.15999031],[-0.04793409, 0.92471859]])
c = np.array([[0.24259536, 0.0874714],[-2.41978734, -1.98990137]])
forward_z = [a, b, c]
for pred, true in zip(nn1.z, forward_z):
    assert pytest.approx(pred, 0.01) == true, "Check forward function"
Congratulations! All test cases in this cell passed.
Type here to search
23°C Smoke
ENG
IN
7:04 PM
17-Dec-23