

CS5542 Big Data Apps and Analytics

In Class Programming –10

29th October 2020

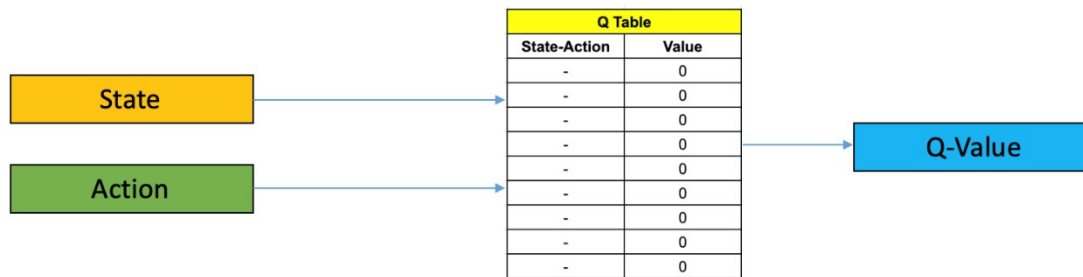
Due Date: 11/3/2020 (Tuesday by 11:59pm)

Submit ICP Feedback in Class. : [Lnk to Feed back Form](#)

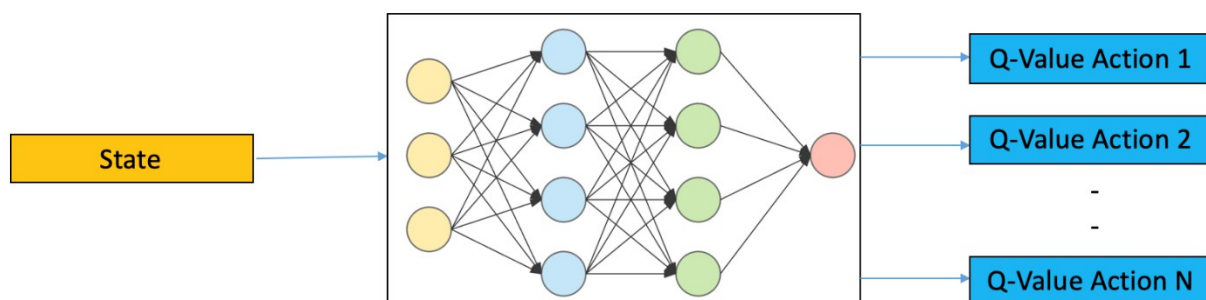
Deep Q-Learning:

Implementing Deep Q-Learning in Python using Keras & OpenAI Gym:

In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output. The comparison between Q-learning & deep Q-learning is illustrated below:



Q Learning



Deep Q Learning

CartPole is one of the simplest environments in the OpenAI gym (a game simulator). The idea of CartPole is that there is a pole standing up on top of a cart. The goal is to balance this pole by moving the cart from side to side to keep the pole balanced upright.

Design a Deep Q learning Network (DQN), using Keras & OpenAI Gym , for cartpole game and visualize your results.

ICP Requirements:

- 1) Designing a DQN for cartpole game in python using Keras & OpenAI Gym (70 points)
- 2) Visualization of DQN cartpole game (10 points)
- 3) overall code quality (10 points)
- 4) Pdf Report quality, video explanation (10 points)

Submission Guidelines:

Same as previous ICPs.

ICP Report:

What I learned in the ICP:

I learned how to use Deep Learning to play and solve games. I learned some about the Gym library and how that is use for game play and deep learning. I learned how to display the cart pole environment.

Description of what task I was performing:

Design a deep learning solution to play the cart pole game and render it on the screen.

Challenges I faced:

One of the hardest parts was trying to figure out how to display the cart pole.

Screen Shots

GitHub:

The screenshot shows a web browser with three tabs: 'UMKC-APL-BigDataAnalytics/icp', 'Lab10.ipynb - Colaboratory', and 'CSC5542/Lab10 at master · BillYerkes'. The address bar shows the URL 'github.com/BillYerkes/CSC5542/tree/master/Lab10'. The GitHub interface includes a search bar, navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore', and a repository header for 'BillYerkes / CSC5542'. Below the header is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The main content area shows the 'master' branch selected, with the path 'CSC5542 / Lab10 /'. A file list is displayed, including a commit by 'BillYerkes' to 'Create Test.txt', a '..' entry, a folder 'Images' with a 'Create Test.txt' file, and a file 'Lab10.ipynb' with an 'Add files via upload' button.

UMKC-APL-BigDataAnalytics/icp x Lab10.ipynb - Colaboratory x CSC5542/Lab10 at master · BillYerkes x +

github.com/BillYerkes/CSC5542/tree/master/Lab10

Search or jump to... Pull requests Issues Marketplace Explore

BillYerkes / CSC5542

<> Code ! Issues 🔗 Pull requests 🎬 Actions 📁 Projects 📖 Wiki 🛡 Security 📈 Insights ⚙ Settings

master CSC5542 / Lab10 /

BillYerkes Create Test.txt

..

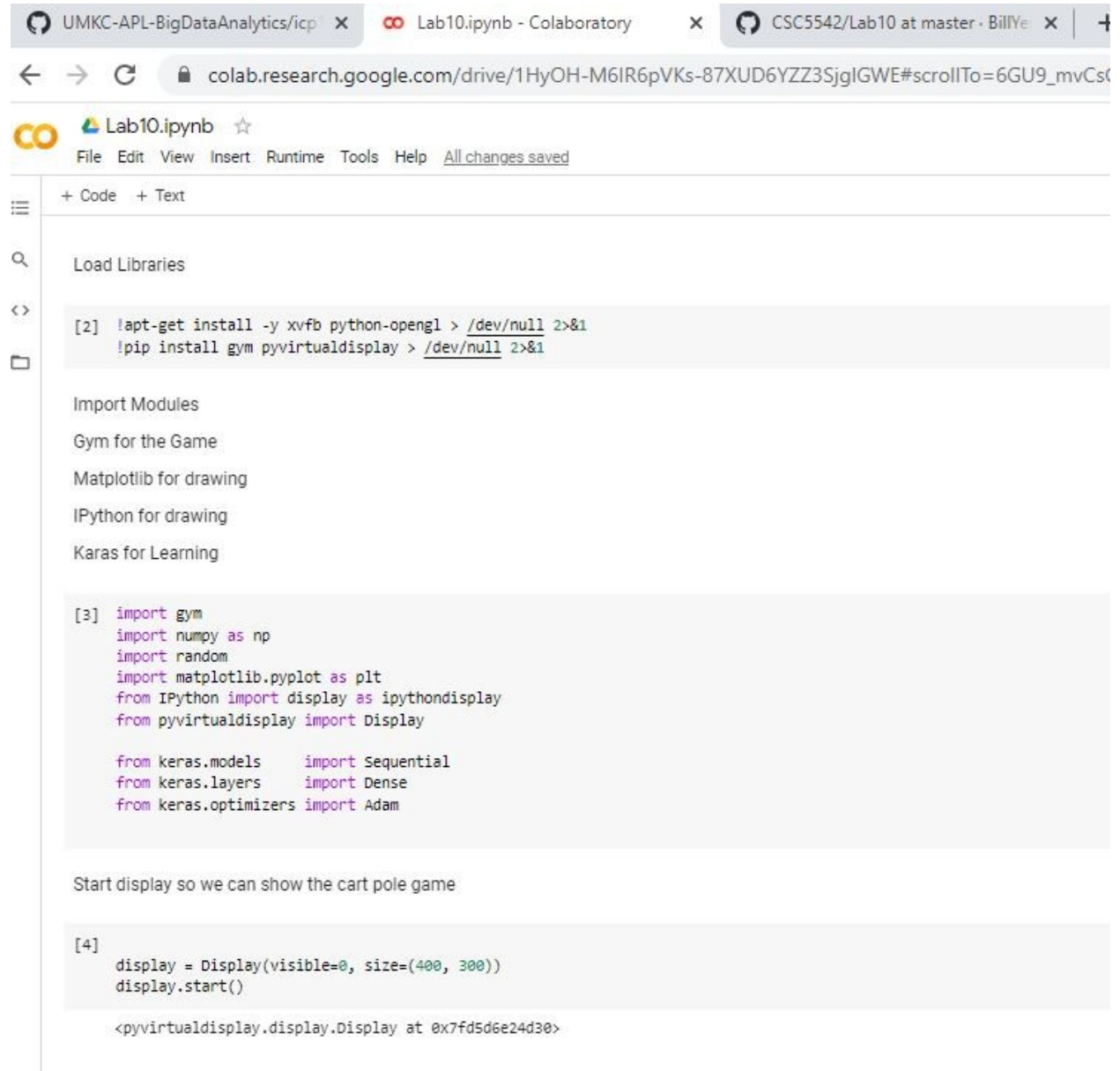
Images Create Test.txt

Lab10.ipynb Add files via upload

Load Libraries and Import Modules

Loading libraries to be able to display cart pole game.

Importing needed modules to create, play, track, and display the game.



The screenshot shows a Google Colaboratory notebook titled "Lab10.ipynb". The browser tabs at the top include "UMKC-APL-BigDataAnalytics/icp", "Lab10.ipynb - Colaboratory", and "CSC5542/Lab10 at master · Billye". The address bar shows the URL "colab.research.google.com/drive/1HyOH-M6IR6pVKs-87XUD6YZZ3SjglGWE#scrollTo=6GU9_mvCs". The notebook interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", and a status bar indicating "All changes saved".

The notebook content is organized into sections:

- Load Libraries**: A code cell [2] containing the following commands:

```
[2] !apt-get install -y xvfb python-opengl > /dev/null 2>&1
    !pip install gym pyvirtualdisplay > /dev/null 2>&1
```
- Import Modules**: A section listing the modules to be imported:
 - Gym for the Game
 - Matplotlib for drawing
 - IPython for drawing
 - Keras for Learning
- Code Cell [3]**: A code cell containing the following import statements:

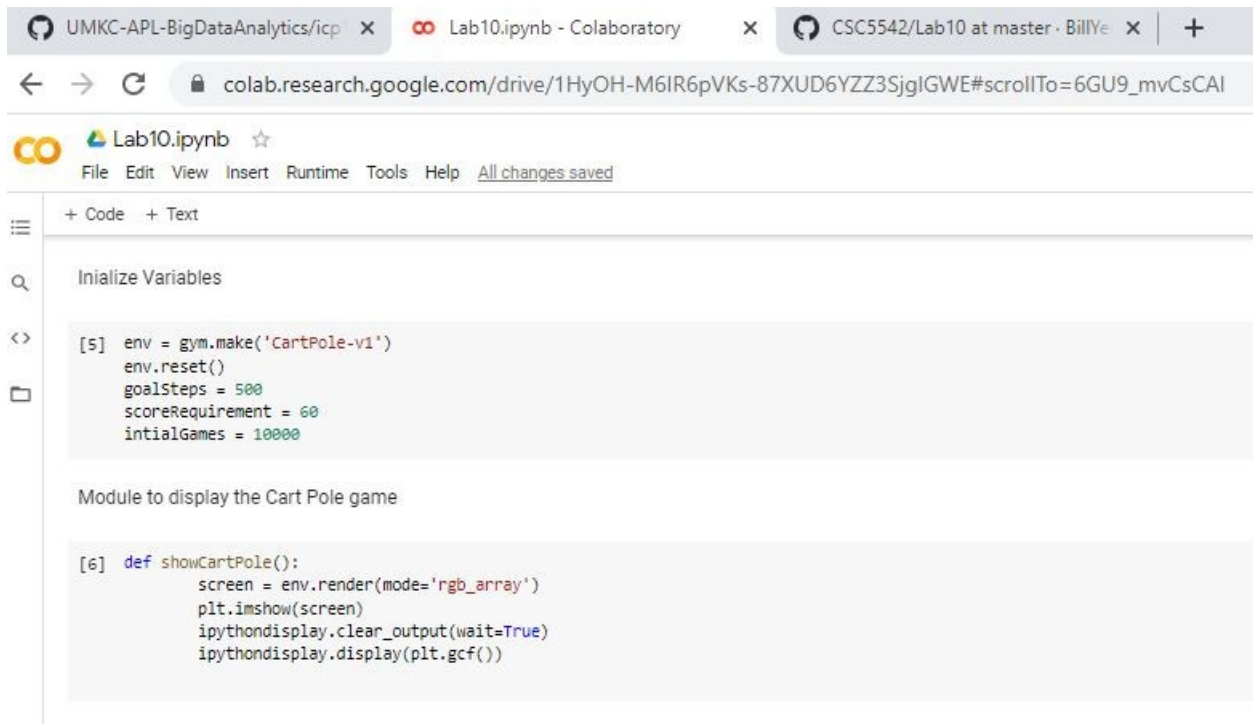
```
[3] import gym
    import numpy as np
    import random
    import matplotlib.pyplot as plt
    from IPython import display as ipythondisplay
    from pyvirtualdisplay import Display

    from keras.models import Sequential
    from keras.layers import Dense
    from keras.optimizers import Adam
```
- Start display so we can show the cart pole game**: A section with a code cell [4] containing the following commands:

```
[4] display = Display(visible=0, size=(400, 300))
    display.start()

<pyvirtualdisplay.display.Display at 0x7fd5d6e24d30>
```

Setting up global variables for the game and creating module to display the game.



The screenshot shows a Google Colaboratory notebook interface. The browser tabs at the top include 'UMKC-APL-BigDataAnalytics/icp', 'Lab10.ipynb - Colaboratory', and 'CSC5542/Lab10 at master · Billye'. The address bar shows the URL 'colab.research.google.com/drive/1HyOH-M6IR6pVKs-87XUD6YZZ3SjglGWE#scrollTo=6GU9_mvCsCAI'. The notebook title is 'Lab10.ipynb' with a star icon. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a link 'All changes saved'. The left sidebar has icons for file explorer, search, and code execution. The main code area contains two code blocks. The first block, titled 'Initalize Variables', contains code to create a Gym environment, reset it, and set parameters. The second block, titled 'Module to display the Cart Pole game', contains a function definition for displaying the game state.

```
[5] env = gym.make('CartPole-v1')
    env.reset()
    goalSteps = 500
    scoreRequirement = 60
    intialGames = 10000

Module to display the Cart Pole game

[6] def showCartPole():
    screen = env.render(mode='rgb_array')
    plt.imshow(screen)
    ipythondisplay.clear_output(wait=True)
    ipythondisplay.display(plt.gcf())
```

Module to play random game:

UMKC-APL-BigDataAnalytics/icp x Lab10.ipynb - Colaboratory x

colab.research.google.com/drive/1HyOH-M6IR6pVKs-87XUD6Y

Lab10.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def playRandomGame():
    #reset environment
    env.reset()

    #set up display
    prevScreen = env.render(mode='rgb_array')
    plt.imshow(prevScreen)

    #Play game
    for step in range(goalsteps):

        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)

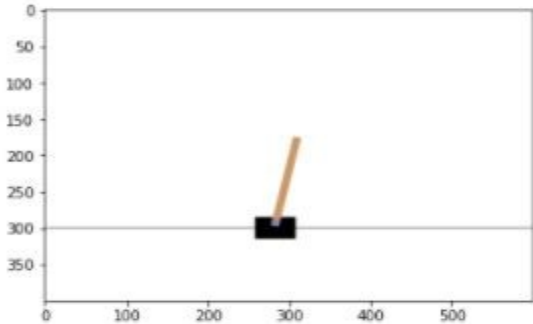
        #Show the cart pole
        showCartPole()

        #Show stats
        print("step:      {}".format(step))
        print("action:     {}".format(action))
        print("observation: {}".format(observation))
        print("reward:      {}".format(reward))
        print("done:        {}".format(done))
        print("info:        {}".format(info))

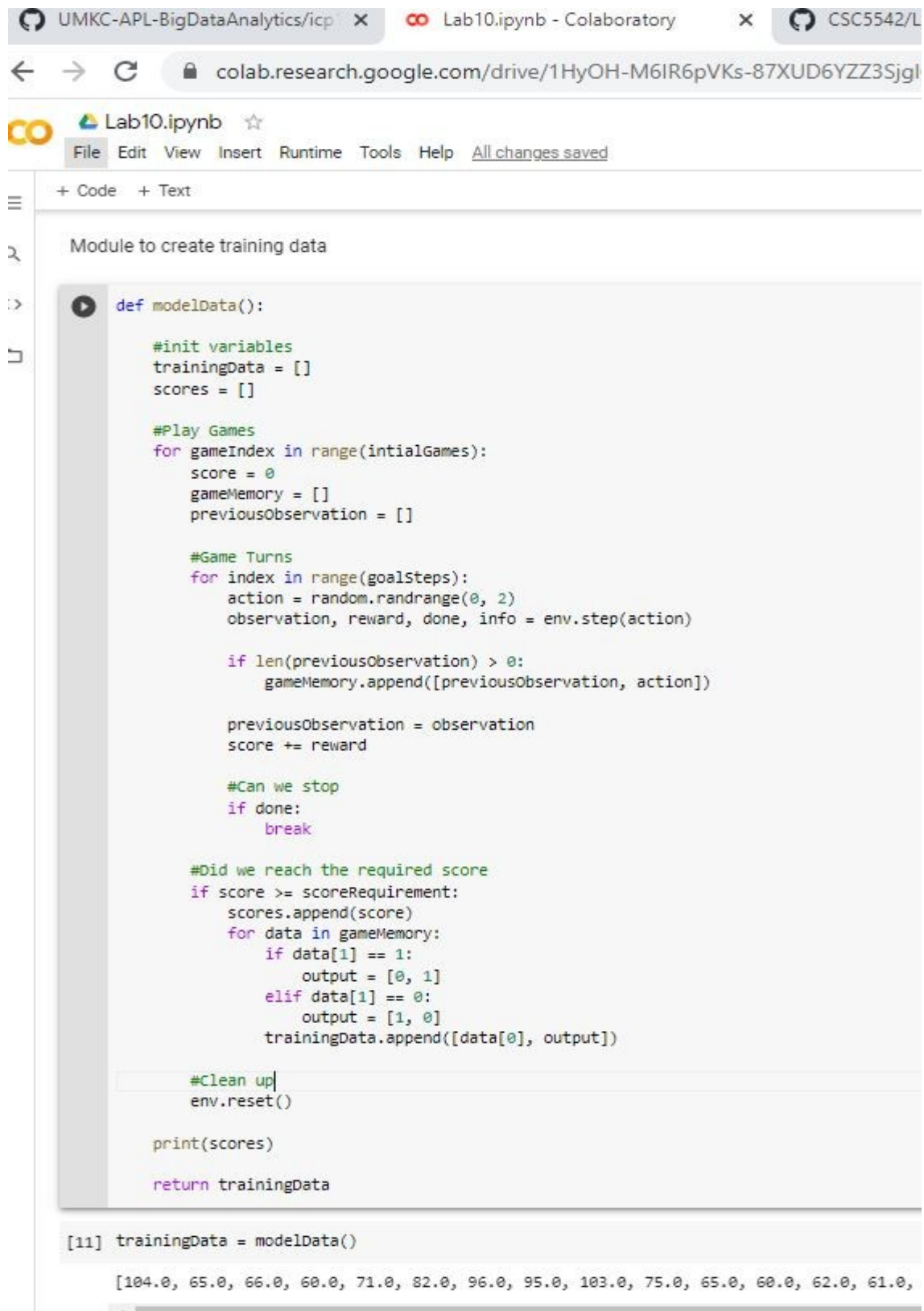
        #If we are finised exit routine
        if done:
            break

    #Clean up
    env.reset()
    ipythondisplay.clear_output(wait=True)
    env.close()
```

[9] playRandomGame()



Create the training data:



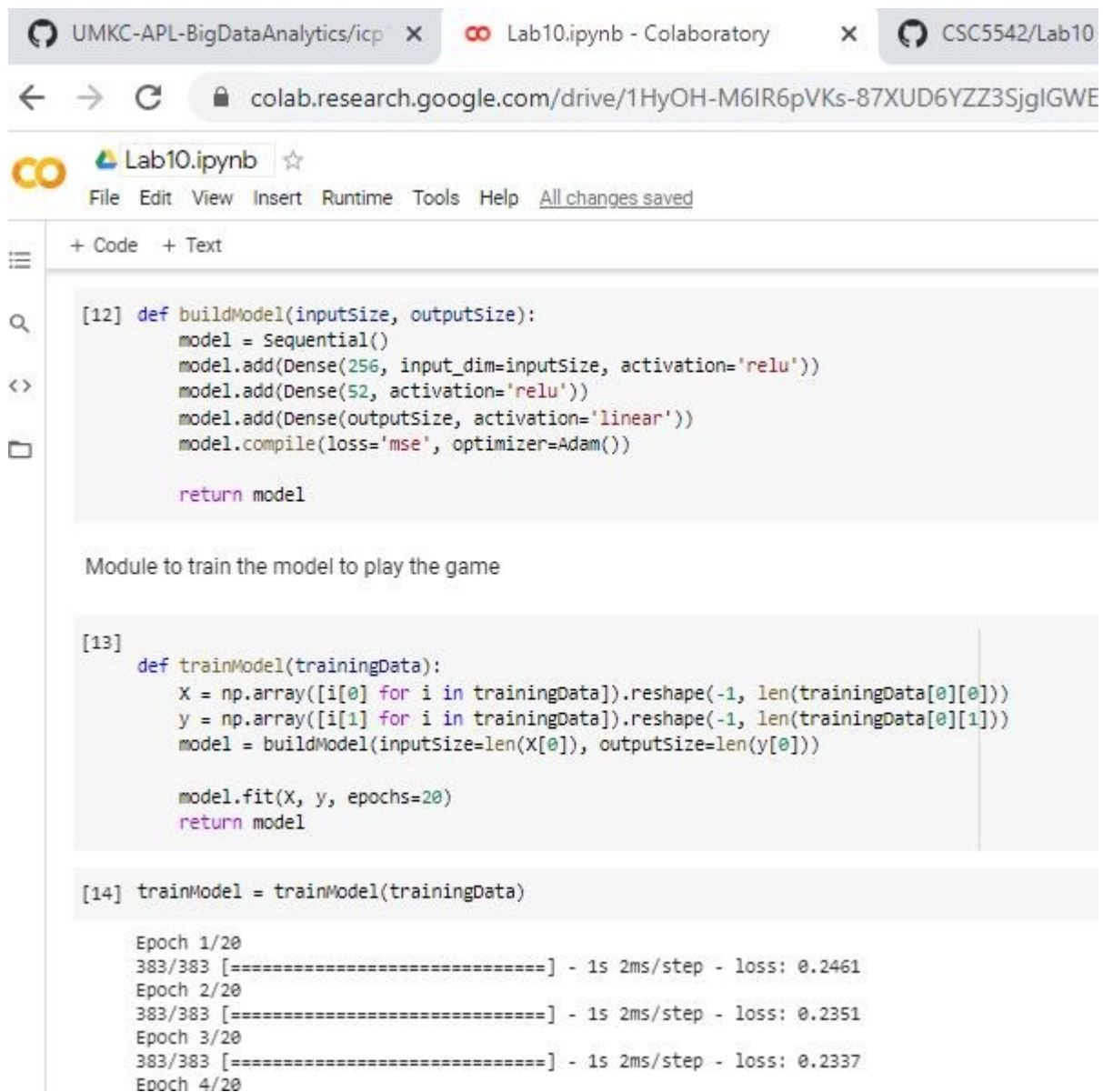
The screenshot shows a Google Colaboratory notebook interface. At the top, there are browser tabs for 'UMKC-APL-BigDataAnalytics/icp', 'Lab10.ipynb - Colaboratory', and 'CSC5542/L'. The address bar shows the URL 'colab.research.google.com/drive/1HyOH-M6IR6pVKs-87XUD6YZZ3Sjgl'. The notebook title is 'Lab10.ipynb'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and 'All changes saved'. The notebook content is divided into a code editor and an output area.

The code editor contains a Python function named `modelData()` with the following logic:

- Initialization:** `trainingData = []` and `scores = []`.
- Play Games:** A loop `for gameIndex in range(intialGames):` (note the typo in the code).
- Game Setup:** `score = 0`, `gameMemory = []`, and `previousObservation = []`.
- Game Turns:** A loop `for index in range(goalSteps):`.
- Action:** `action = random.randrange(0, 2)`.
- Step:** `observation, reward, done, info = env.step(action)`.
- Memory:** If `len(previousObservation) > 0`, append `[previousObservation, action]` to `gameMemory`.
- Update:** `previousObservation = observation` and `score += reward`.
- Stop Condition:** If `done`, break the loop.
- Score Requirement:** If `score >= scoreRequirement`, append `score` to `scores`.
- Data Collection:** For each `data` in `gameMemory`, if `data[1] == 1`, `output = [0, 1]`; if `data[1] == 0`, `output = [1, 0]`. Append `[data[0], output]` to `trainingData`.
- Clean up:** `env.reset()`.
- Output:** `print(scores)`.
- Return:** `return trainingData`.

The output area shows the execution of `trainingData = modelData()` at cell [11], resulting in the following list of scores: `[104.0, 65.0, 66.0, 60.0, 71.0, 82.0, 96.0, 95.0, 103.0, 75.0, 65.0, 60.0, 62.0, 61.0,`

Build and Train the model on how to play the game:



The screenshot shows a Google Colaboratory notebook with the following content:

Browser tabs: UMKC-APL-BigDataAnalytics/icp, Lab10.ipynb - Colaboratory, CSC5542/Lab10

Address bar: colab.research.google.com/drive/1HyOH-M6IR6pVKs-87XUD6YZZ3SjglGWE

Notebook title: Lab10.ipynb

Menu: File Edit View Insert Runtime Tools Help All changes saved

Code editor tabs: + Code + Text

```
[12] def buildModel(inputSize, outputSize):
    model = Sequential()
    model.add(Dense(256, input_dim=inputSize, activation='relu'))
    model.add(Dense(52, activation='relu'))
    model.add(Dense(outputSize, activation='linear'))
    model.compile(loss='mse', optimizer=Adam())

    return model
```

Module to train the model to play the game

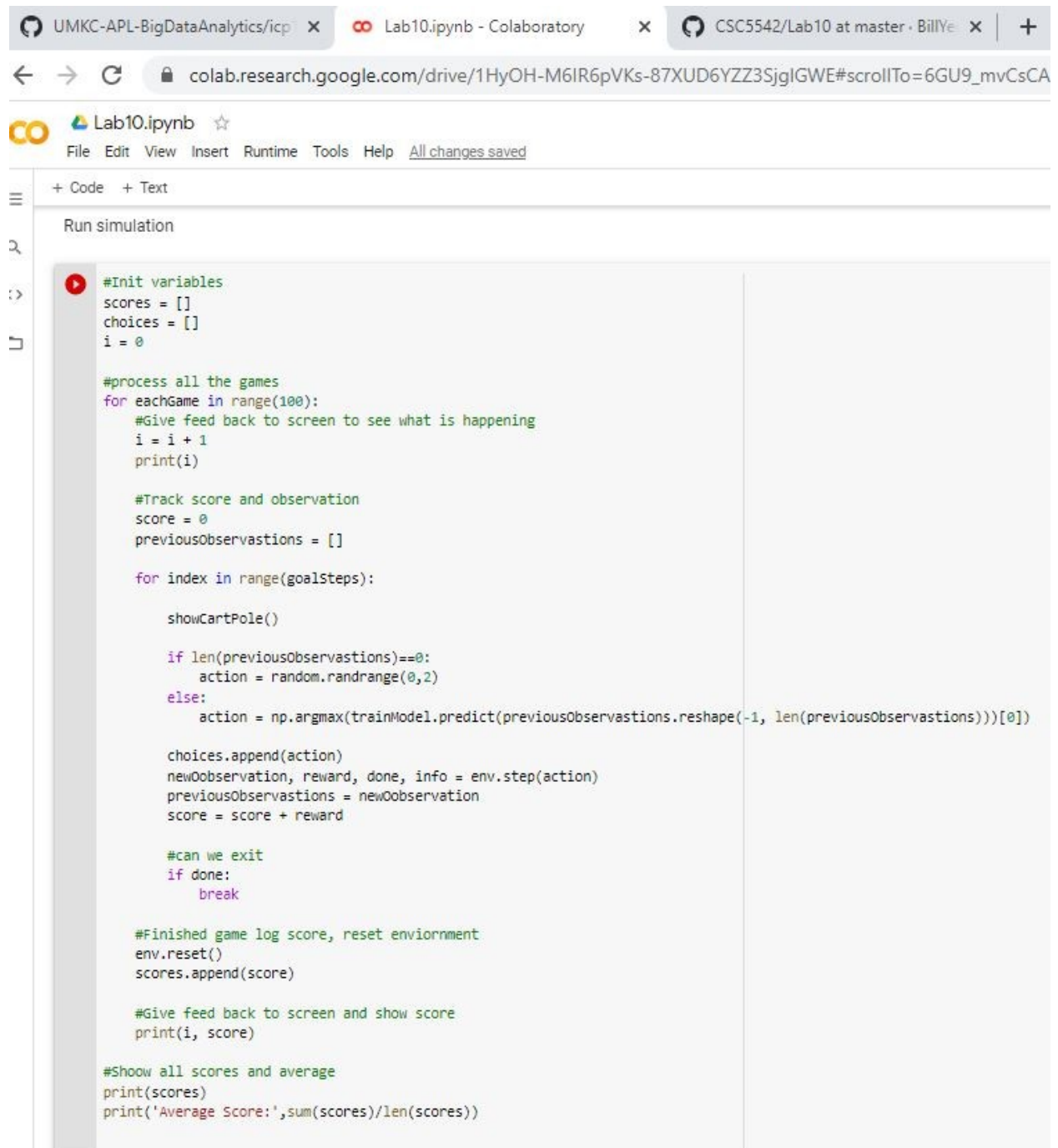
```
[13] def trainModel(trainingData):
    X = np.array([i[0] for i in trainingData]).reshape(-1, len(trainingData[0][0]))
    y = np.array([i[1] for i in trainingData]).reshape(-1, len(trainingData[0][1]))
    model = buildModel(inputSize=len(X[0]), outputSize=len(y[0]))

    model.fit(X, y, epochs=20)
    return model
```

```
[14] trainModel = trainModel(trainingData)
```

Epoch 1/20
383/383 [=====] - 1s 2ms/step - loss: 0.2461
Epoch 2/20
383/383 [=====] - 1s 2ms/step - loss: 0.2351
Epoch 3/20
383/383 [=====] - 1s 2ms/step - loss: 0.2337
Epoch 4/20

Run the simulation:



UMKC-APL-BigDataAnalytics/icp x Lab10.ipynb - Colaboratory x CSC5542/Lab10 at master · BillyE x +

colab.research.google.com/drive/1HyOH-M6IR6pVKs-87XUD6YZZ3SjgIGWE#scrollTo=6GU9_mvCsCA

Lab10.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Run simulation

```
#Init variables
scores = []
choices = []
i = 0

#process all the games
for eachGame in range(100):
    #Give feed back to screen to see what is happening
    i = i + 1
    print(i)

    #Track score and observation
    score = 0
    previousObservastions = []

    for index in range(goalSteps):

        showCartPole()

        if len(previousObservastions)==0:
            action = random.randrange(0,2)
        else:
            action = np.argmax(trainModel.predict(previousObservastions.reshape(-1, len(previousObservastions))))[0])

        choices.append(action)
        newObservation, reward, done, info = env.step(action)
        previousObservastions = newObservation
        score = score + reward

    #can we exit
    if done:
        break

    #Finished game log score, reset enviorment
    env.reset()
    scores.append(score)

    #Give feed back to screen and show score
    print(i, score)

#Shoow all scores and average
print(scores)
print('Average Score:',sum(scores)/len(scores))
```

[Video Link](#)

Any in site about the data or the ICP in general

Would like to do more with this type of programming.