Stock Trading Using Deep Q-Learning

Problem Statement

Prepare an agent by implementing Deep Q-Learning that can perform unsupervised trading in stock trade. The aim of this project is to train an agent that uses Q-learning and neural networks to predict the profit or loss by building a model and implementing it on a dataset that is available for evaluation.

The stock trading index environment provides the agent with a set of actions:

- Buy
- Sell
- Sit

This project has following sections:

- Import libraries
- Create a DQN agent
- Preprocess the data
- Train and build the model
- Evaluate the model and agent

Steps to perform

In the section **create a DQN agent**, create a class called agent where:

- Action size is defined as 3
- Experience replay memory to deque is 1000
- Empty list for stocks that has already been bought
- The agent must possess the following hyperparameters:
  - gamma= 0.95
  - epsilon = 1.0
  - epsilon_final = 0.01
  - epsilon_decay = 0.995

  `Note: It is advised to compare the results using different values in hyperparameters.`

- Neural network has 3 hidden layers
- Action and experience replay are defined

Solution

Import the libraries

In [3]:

```
import warnings
warnings.filterwarnings("ignore")

import keras
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.optimizers import adam_v2
import numpy as np
import random
from collections import deque
```

Create a DQN agent

Use the instruction below to prepare an agent

In [ ]:

```
# Action space include 3 actions: Buy, Sell, and Sit
#Setting up the experience replay memory to deque with 1000 elements inside it
#Empty list with inventory is created that contains the stocks that were already bought
#Setting up gamma to 0.95, that helps to maximize the current reward over the long-term
#Epsilon parameter determines whether to use a random action or to use the model for the action.
#In the beginning random actions are encouraged, hence epsilon is set up to 1.0 when the model is not trained.
#And over time the epsilon is reduced to 0.01 in order to decrease the random actions and use the trained mode
#We're then set the speed of decreasing epsililon in the epsilon_decay parameter

#Defining our neural network:
#Define the neural network function called _model and it just takes the keyword self
#Define the model with Sequential()
#Define states i.e. the previous n days and stock prices of the days
#Defining 3 hidden layers in this network
#Changing the activation function to relu because mean-squared error is used for the loss
```

Preprocess the stock market data

In [4]:

```
import math

# prints formatted price
def formatPrice(n):
        return ("-$" if n < 0 else "$") + "{0:.2f}".format(abs(n))

# returns the vector containing stock data from a fixed file
def getStockDataVec(key):
        vec = []
        lines = open("" + key + ".csv", "r").read().splitlines()

        for line in lines[1:]:
                vec.append(float(line.split(",")[4]))

        return vec

# returns the sigmoid
def sigmoid(x):
        return 1 / (1 + math.exp(-x))

# returns an an n-day state representation ending at time t
def getState(data, t, n):
        d = t - n + 1
        block = data[d:t + 1] if d >= 0 else -d * [data[0]] + data[0:t + 1] # pad with t0
        res = []
        for i in range(n - 1):
                res.append(sigmoid(block[i + 1] - block[i]))

        return np.array([res])
```

In [5]:

```
class Agent():
    def __init__(self,window_size,is_eval=False,model_name=""):
        self.nS = window_size
        self.nA = 3
        self.memory = deque([],maxlen=1000)
        self.alpha = .001
        self.window_size = window_size
        self.gamma = 0.95
        #Explore/Explot
        self.epsilon = 1
        self.epsilon_min = 0.01
        self.epsilon_decay = 0.995
        self.loss = []

        self.is_eval = is_eval
        self.model = load_model (model_name) if self.is_eval else self.build_model()

    def build_model(self):
        model = keras.Sequential()
        model.add(keras.layers.Dense(24,input_dim=self.window_size,activation='relu'))
        model.add(keras.layers.Dense(24,activation='relu'))
        model.add(keras.layers.Dense(self.nA,activation='linear'))
        #model.compile(loss='mean_squared_error',optimizer=adam_v2(learning_rate=self.alpha))
        model.compile(loss="mse", optimizer=adam_v2.Adam(learning_rate=self.alpha))
        return model

    def act(self,state):
        if np.random.rand() <= self.epsilon:
            return random.randrange(3) # Explore
        action_vals = self.model.predict(state)
        return np.argmax(action_vals[0])

    def test_action(): #Exploit
        action_vals = self.model.predict(state)
        return np.argmax(action_valls[0])

    def store(self,state,action,reward,nstate,done):
        # Store the experience in memory
        self.memory.append((state,action,reward,nstate,done))

    def expReplay(self,batch_size):
        # Execute the experience replay
        minibatch = random.sample(self.memory,batch_size)

        x=[]
        y=[]
        np_array = np.array(minibatch)
        st = np.zeros((0,self.nS)) # State
        nst  = np.zeros((0,self.nS)) # Next State
        for i in range (len(np_array)):
            st = np.append(st,np_array[i,0],axis=0)
            nst = np.append(st,np_array[i,3],axis=0)

        st_predict= self.model.predict(st) # speedup, can do on the Entire batch as well
        nst_predict= self.model.predict(nst)

        index = 0
        for state,action,reward,nstate,done in minibatch:
            x.append(state)
            # Predict from state
            nst_action_predict_model = nst_predict[index]
            if done == True:   # Terminal
                target = reward
            else: # Non Terminal
                target = reward + self.gamma * np.amax(nst_action_predict_model)

            target_f = st_predict[index]
            target_f[action] = target
            y.append(target_f)
            index +=1

        #Reshape the keras fit
        x_reshape = np.array(x).reshape(batch_size,self.nS)
        y_reshape = np.array(y)
        epoch_count = 1
        hist = self.model.fit(x_reshape,y_reshape,epochs=epoch_count,verbose=0)
        # Graph losses
        for i in range(epoch_count):
            self.loss.append( hist.history['loss'][i])
        # decay epsilon
        if self.epsilon > self.epsilon_min:
            self.epsilon *= self.epsilon_decay
```

Train and build the model

In [5]:

```
import sys
from collections import deque

if len(sys.argv) != 4:
        print ("Usage: python train.py [stock] [window] [episodes]")
        exit()


stock_name = input("Enter stock_name, window_size, Episode_count")
#Fill the given information when prompted:
#Enter stock_name = GSPC_Training_Dataset /content/GSPC_Evaluation_Dataset
#window_size = 10
#Episode_count = 100 or it can be 10 or 20 or 30 and so on.

window_size = input()
episode_count = input()
stock_name = str(stock_name)
window_size = int(window_size)
episode_count = int(episode_count)

agent = Agent(window_size)
data = getStockDataVec(stock_name)
l = len(data) - 1
batch_size = 32

for e in range(episode_count + 1):
        print ("Episode " + str(e) + "/" + str(episode_count))
        state = getState(data, 0, window_size + 1)

        total_profit = 0
        agent.inventory = []

        for t in range(l):
                action = agent.act(state)

                # sit
                next_state = getState(data, t + 1, window_size + 1)
                reward = 0

                if action == 1: # buy
                        agent.inventory.append(data[t])
                        #print ("Buy: " + formatPrice(data[t]))

                elif action == 2 and len(agent.inventory) > 0: # sell
                        bought_price = agent.inventory.pop(0)
                        reward = max(data[t] - bought_price, 0)
                        total_profit += data[t] - bought_price
                        #print ("Sell: " + formatPrice(data[t]) + " | Profit: " + formatPrice(data[t] - bought_

                done = True if t == l - 1 else False
                agent.memory.append((state, action, reward, next_state, done))
                state = next_state

                if done:
                        print ("--------------------------------")
                        print ("Total Profit: " + formatPrice(total_profit))


                if len(agent.memory) > batch_size:
                        agent.expReplay(batch_size)

        if e % 10 == 0:
                agent.model.save("model_ep" + str(e))
```

```
Usage: python train.py [stock] [window] [episodes]
Enter stock_name, window_size, Episode_count/content/GSPC_Evaluation_Dataset
10
100
Episode 0/100
--------------------------------
Total Profit: -$1130.21
Episode 1/100
--------------------------------
Total Profit: $17.19
Episode 2/100
--------------------------------
Total Profit: -$1.18
Episode 3/100
--------------------------------
Total Profit: $26.03
Episode 4/100
--------------------------------
Total Profit: -$60.15
Episode 5/100
--------------------------------
Total Profit: $74.57
Episode 6/100
--------------------------------
Total Profit: $1.41
Episode 7/100
--------------------------------
Total Profit: $26.34
Episode 8/100
--------------------------------
Total Profit: $38.36
Episode 9/100
--------------------------------
Total Profit: -$87.07
Episode 10/100
--------------------------------
Total Profit: -$223.95
Episode 11/100
--------------------------------
Total Profit: -$19.76
Episode 12/100
--------------------------------
Total Profit: -$83.73
Episode 13/100
--------------------------------
Total Profit: $30.36
Episode 14/100
--------------------------------
Total Profit: $113.85
Episode 15/100
--------------------------------
Total Profit: $100.37
Episode 16/100
--------------------------------
Total Profit: -$73.23
Episode 17/100
--------------------------------
Total Profit: -$15.04
Episode 18/100
--------------------------------
Total Profit: -$4.98
Episode 19/100
--------------------------------
Total Profit: -$4.56
Episode 20/100
--------------------------------
Total Profit: $56.12
Episode 21/100
--------------------------------
Total Profit: -$50.91
Episode 22/100
--------------------------------
Total Profit: $101.48
Episode 23/100
--------------------------------
Total Profit: -$42.45
Episode 24/100
--------------------------------
Total Profit: $78.78
Episode 25/100
--------------------------------
Total Profit: $86.96
Episode 26/100
--------------------------------
Total Profit: $21.74
Episode 27/100
--------------------------------
Total Profit: $117.25
Episode 28/100
--------------------------------
Total Profit: $58.21
Episode 29/100
--------------------------------
Total Profit: -$140.05
Episode 30/100
--------------------------------
Total Profit: -$20.85
Episode 31/100
--------------------------------
Total Profit: -$191.23
Episode 32/100
--------------------------------
Total Profit: $102.20
Episode 33/100
--------------------------------
Total Profit: $573.42
Episode 34/100
--------------------------------
Total Profit: -$147.66
Episode 35/100
--------------------------------
Total Profit: -$192.26
Episode 36/100
--------------------------------
Total Profit: $58.11
Episode 37/100
--------------------------------
Total Profit: $126.53
Episode 38/100
--------------------------------
Total Profit: -$95.53
Episode 39/100
--------------------------------
Total Profit: $121.31
Episode 40/100
--------------------------------
Total Profit: $30.82
Episode 41/100
--------------------------------
Total Profit: $142.45
Episode 42/100
--------------------------------
Total Profit: $198.18
Episode 43/100
--------------------------------
Total Profit: $75.77
Episode 44/100
--------------------------------
Total Profit: -$67.23
Episode 45/100
--------------------------------
Total Profit: $21.17
Episode 46/100
--------------------------------
Total Profit: $120.22
Episode 47/100
--------------------------------
Total Profit: $65.75
Episode 48/100
--------------------------------
Total Profit: $148.76
Episode 49/100
--------------------------------
Total Profit: $134.71
Episode 50/100
--------------------------------
Total Profit: -$53.95
Episode 51/100
--------------------------------
Total Profit: $201.16
Episode 52/100
--------------------------------
Total Profit: $2.72
Episode 53/100
--------------------------------
Total Profit: $8.22
Episode 54/100
--------------------------------
Total Profit: $77.63
Episode 55/100
--------------------------------
Total Profit: $28.64
Episode 56/100
--------------------------------
Total Profit: $150.36
Episode 57/100
--------------------------------
Total Profit: $86.14
Episode 58/100
--------------------------------
Total Profit: $13.10
Episode 59/100
--------------------------------
Total Profit: $76.66
Episode 60/100
--------------------------------
Total Profit: $153.73
Episode 61/100
--------------------------------
Total Profit: $42.67
Episode 62/100
--------------------------------
Total Profit: -$19.12
Episode 63/100
--------------------------------
Total Profit: -$71.03
Episode 64/100
--------------------------------
Total Profit: $57.01
Episode 65/100
--------------------------------
Total Profit: -$67.74
Episode 66/100
--------------------------------
Total Profit: -$35.79
Episode 67/100
--------------------------------
Total Profit: -$24.04
Episode 68/100
--------------------------------
Total Profit: $55.49
Episode 69/100
--------------------------------
Total Profit: $37.29
Episode 70/100
--------------------------------
Total Profit: -$68.03
Episode 71/100
--------------------------------
Total Profit: -$221.20
Episode 72/100
--------------------------------
Total Profit: $3.61
Episode 73/100
--------------------------------
Total Profit: $108.94
Episode 74/100
--------------------------------
Total Profit: $178.88
Episode 75/100
--------------------------------
Total Profit: $116.79
Episode 76/100
--------------------------------
Total Profit: $22.36
Episode 77/100
--------------------------------
Total Profit: $35.26
Episode 78/100
--------------------------------
Total Profit: -$25.46
Episode 79/100
--------------------------------
Total Profit: $1.22
Episode 80/100
--------------------------------
Total Profit: $205.81
Episode 81/100
--------------------------------
Total Profit: $20.93
Episode 82/100
--------------------------------
Total Profit: $192.64
Episode 83/100
--------------------------------
Total Profit: -$36.63
Episode 84/100
--------------------------------
Total Profit: $23.98
Episode 85/100
--------------------------------
Total Profit: -$147.11
Episode 86/100
--------------------------------
Total Profit: -$258.32
Episode 87/100
--------------------------------
Total Profit: -$254.56
Episode 88/100
--------------------------------
Total Profit: $51.12
Episode 89/100
--------------------------------
Total Profit: $361.17
Episode 90/100
--------------------------------
Total Profit: $130.86
Episode 91/100
--------------------------------
Total Profit: $30.36
Episode 92/100
--------------------------------
Total Profit: -$54.79
Episode 93/100
--------------------------------
Total Profit: -$3.14
Episode 94/100
--------------------------------
Total Profit: -$47.94
Episode 95/100
--------------------------------
Total Profit: -$198.18
Episode 96/100
--------------------------------
Total Profit: -$346.77
Episode 97/100
--------------------------------
Total Profit: $52.02
Episode 98/100
--------------------------------
Total Profit: -$274.26
Episode 99/100
--------------------------------
Total Profit: $49.93
Episode 100/100
--------------------------------
Total Profit: -$82.84
```

Evaluate the model and agent

In [8]:

```
import sys
from keras.models import load_model


if len(sys.argv) != 3:
        print ("Usage: python evaluate.py [stock] [model]")
        exit()


stock_name = input("Enter Stock_name, Model_name")
model_name = input()
#Note:
#Fill the given information when prompted:
#Enter stock_name = GSPC_Evaluation_Dataset /content/GSPC_Evaluation_Dataset
#Model_name = respective model name /content/model_ep90

model = load_model(model_name)
window_size = model.layers[0].input.shape.as_list()[1]

agent = Agent(window_size, True, model_name)
data = getStockDataVec(stock_name)
l = len(data) - 1
batch_size = 32

state = getState(data, 0, window_size + 1)
total_profit = 0
agent.inventory = []

for t in range(l):
        action = agent.act(state)

        # sit
        next_state = getState(data, t + 1, window_size + 1)
        reward = 0

        if action == 1: # buy
                agent.inventory.append(data[t])
                print ("Buy: " + formatPrice(data[t]))

        elif action == 2 and len(agent.inventory) > 0: # sell
                bought_price = agent.inventory.pop(0)
                reward = max(data[t] - bought_price, 0)
                total_profit += data[t] - bought_price
                print ("Sell: " + formatPrice(data[t]) + " | Profit: " + formatPrice(data[t] - bought_price))

        done = True if t == l - 1 else False
        agent.memory.append((state, action, reward, next_state, done))
        state = next_state

        if done:
                print ("--------------------------------")
                print (stock_name + " Total Profit: " + formatPrice(total_profit))
```

```
Enter Stock_name, Model_name/content/GSPC_Evaluation_Dataset
/content/model_ep90
Buy: $1276.56
Buy: $1269.75
Sell: $1274.48 | Profit: -$2.08
Sell: $1285.96 | Profit: $16.21
Buy: $1293.24
Buy: $1295.02
Buy: $1281.92
Buy: $1280.26
Buy: $1283.35
Sell: $1290.84 | Profit: -$2.40
Sell: $1291.18 | Profit: -$3.84
Buy: $1296.63
Sell: $1299.54 | Profit: $17.62
Buy: $1276.34
Buy: $1307.59
Buy: $1304.03
Sell: $1310.87 | Profit: $30.61
Sell: $1324.57 | Profit: $41.22
Sell: $1332.32 | Profit: $35.69
Sell: $1328.01 | Profit: $51.67
Sell: $1340.43 | Profit: $32.84
Sell: $1343.01 | Profit: $38.98
Buy: $1307.40
Buy: $1306.10
Sell: $1327.22 | Profit: $19.82
Sell: $1330.97 | Profit: $24.87
Buy: $1256.88
Sell: $1273.72 | Profit: $16.84
Buy: $1309.66
Buy: $1310.19
Buy: $1319.44
Sell: $1328.26 | Profit: $18.60
Sell: $1332.41 | Profit: $22.22
Sell: $1332.87 | Profit: $13.43
Buy: $1332.63
Sell: $1328.17 | Profit: -$4.46
Buy: $1337.38
Sell: $1335.25 | Profit: -$2.13
Buy: $1360.48
Buy: $1363.61
Buy: $1361.22
Sell: $1356.62 | Profit: -$3.86
Buy: $1335.10
Sell: $1346.29 | Profit: -$17.32
Sell: $1357.16 | Profit: -$4.06
Sell: $1337.77 | Profit: $2.67
Buy: $1340.68
Buy: $1333.27
Buy: $1316.28
Sell: $1320.47 | Profit: -$20.21
Buy: $1345.20
Sell: $1314.55 | Profit: -$18.72
Buy: $1312.94
Buy: $1284.94
Sell: $1279.56 | Profit: -$36.72
Buy: $1289.00
Sell: $1270.98 | Profit: -$74.22
Buy: $1271.83
Buy: $1287.87
Sell: $1267.64 | Profit: -$45.30
Buy: $1271.50
Sell: $1278.36 | Profit: -$6.58
Buy: $1295.52
Sell: $1287.14 | Profit: -$1.86
Sell: $1283.50 | Profit: $11.67
Buy: $1280.10
Sell: $1339.22 | Profit: $51.35
Sell: $1353.22 | Profit: $81.72
Buy: $1343.80
Buy: $1319.49
Sell: $1313.64 | Profit: $18.12
Sell: $1317.72 | Profit: $37.62
Sell: $1316.14 | Profit: -$27.66
Sell: $1325.84 | Profit: $6.35
Buy: $1345.02
Buy: $1304.89
Sell: $1300.67 | Profit: -$44.35
Sell: $1286.94 | Profit: -$17.95
Buy: $1260.34
Buy: $1200.07
Sell: $1199.38 | Profit: -$60.96
Buy: $1119.46
Buy: $1172.53
Buy: $1120.76
Sell: $1172.64 | Profit: -$27.43
Sell: $1204.49 | Profit: $85.03
Sell: $1192.76 | Profit: $20.23
Sell: $1193.89 | Profit: $73.13
Buy: $1140.65
Buy: $1162.35
Buy: $1177.60
Buy: $1159.27
Buy: $1176.80
Sell: $1210.08 | Profit: $69.43
Buy: $1218.89
Buy: $1204.42
Sell: $1173.97 | Profit: $11.62
Sell: $1165.24 | Profit: -$12.36
Sell: $1185.90 | Profit: $26.63
Sell: $1162.27 | Profit: -$14.53
Sell: $1172.87 | Profit: -$46.02
Buy: $1209.11
Sell: $1216.01 | Profit: $11.59
Sell: $1204.09 | Profit: -$5.02
Buy: $1166.76
Buy: $1136.43
Buy: $1162.95
Buy: $1151.06
Sell: $1160.40 | Profit: -$6.36
Buy: $1131.42
Sell: $1099.23 | Profit: -$37.20
Buy: $1144.03
Buy: $1155.46
Buy: $1195.54
Buy: $1207.25
Buy: $1203.66
Sell: $1224.58 | Profit: $61.63
Buy: $1215.39
Buy: $1254.19
Buy: $1242.00
Buy: $1218.28
Sell: $1237.90 | Profit: $86.84
Sell: $1261.15 | Profit: $129.73
Buy: $1261.12
Buy: $1275.92
Sell: $1229.10 | Profit: $85.07
Sell: $1236.91 | Profit: $81.45
Sell: $1216.13 | Profit: $20.59
Sell: $1158.67 | Profit: -$48.58
Sell: $1192.55 | Profit: -$11.11
Buy: $1195.19
Sell: $1246.96 | Profit: $31.57
Sell: $1257.08 | Profit: $2.89
Sell: $1258.47 | Profit: $16.47
Buy: $1261.01
Buy: $1234.35
Buy: $1225.73
Sell: $1211.82 | Profit: -$6.46
Sell: $1215.75 | Profit: -$45.37
Buy: $1219.66
Buy: $1205.35
Sell: $1254.00 | Profit: -$21.92
Sell: $1249.64 | Profit: $54.45
Buy: $1263.02
--------------------------------
/content/GSPC_Evaluation_Dataset Total Profit: $781.43
```

Note: Run the training section for considerable episodes so that while evaluating the model it can generate significant profit.