#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

Dataset

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
!unzip -qq
/content/drive/MyDrive/datasets/simplilearn_RL_stock_trading/dataset.z
ip
```

```
!scp /content/dataset/GSPC_Training_Dataset.csv
/content/GSPC_Training_Dataset.csv
!scp /content/dataset/GSPC_Evaluation_Dataset.csv
/content/GSPC_Evaluation_Dataset.csv
```

REINFORCEMENT LIBRARIES

```
!pip install python-opengl xvfb
!pip install pyvirtualdisplay
!apt install xvfb -y
!pip install piglet
!pip3 install box2d-py
!pip3 install gym[Box_2D]
!pip install tensorflow==2.3.1 gym keras-rl2
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
ERROR: Could not find a version that satisfies the requirement python-
opengl (from versions: none)
ERROR: No matching distribution found for python-opengl
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting pyvirtualdisplay
  Downloading PyVirtualDisplay-3.0-py3-none-any.whl (15 kB)
Installing collected packages: pyvirtualdisplay
Successfully installed pyvirtualdisplay-3.0
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer
required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
The following NEW packages will be installed:
  xvfb
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.
Need to get 785 kB of archives.
After this operation, 2,271 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64
```

```
xvfb amd64 2:1.19.6-1ubuntu4.11 [785 kB]
Fetched 785 kB in 0s (2,637 kB/s)
Selecting previously unselected package xvfb.
(Reading database ... 159447 files and directories currently
installed.)
Preparing to unpack .../xvfb_2%3a1.19.6-1ubuntu4.11_amd64.deb ...
Unpacking xvfb (2:1.19.6-1ubuntu4.11) ...
Setting up xvfb (2:1.19.6-1ubuntu4.11) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting piglet
  Downloading piglet-1.0.0-py2.py3-none-any.whl (2.2 kB)
Collecting piglet-templates
  Downloading piglet_templates-1.3.0-py3-none-any.whl (67 kB)
ent already satisfied: attrs in /usr/local/lib/python3.7/dist-packages
(from piglet-templates->piglet) (22.1.0)
Requirement already satisfied: astunparse in
/usr/local/lib/python3.7/dist-packages (from piglet-templates->piglet)
(1.6.3)
Requirement already satisfied: pyparsing in
/usr/local/lib/python3.7/dist-packages (from piglet-templates->piglet)
(3.0.9)
Requirement already satisfied: markupsafe in
/usr/local/lib/python3.7/dist-packages (from piglet-templates->piglet)
(2.0.1)
Requirement already satisfied: six<2.0,>=1.6.1 in
/usr/local/lib/python3.7/dist-packages (from astunparse->piglet-
templates->piglet) (1.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.7/dist-packages (from astunparse->piglet-
templates->piglet) (0.37.1)
Installing collected packages: piglet-templates, piglet
Successfully installed piglet-1.0.0 piglet-templates-1.3.0
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting box2d-py
  Downloading box2d_py-2.3.8-cp37-cp37m-manylinux1_x86_64.whl (448 kB)
ple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gym[Box_2D] in
/usr/local/lib/python3.7/dist-packages (0.25.2)
WARNING: gym 0.25.2 does not provide the extra 'box_2d'
Requirement already satisfied: importlib-metadata>=4.8.0 in
/usr/local/lib/python3.7/dist-packages (from gym[Box_2D]) (4.12.0)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.7/dist-packages (from gym[Box_2D]) (0.0.8)
Requirement already satisfied: numpy>=1.18.0 in
/usr/local/lib/python3.7/dist-packages (from gym[Box_2D]) (1.21.6)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.7/dist-packages (from gym[Box_2D]) (1.5.0)
```

```
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-
metadata>=4.8.0->gym[Box_2D]) (3.8.1)
Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from importlib-
metadata>=4.8.0->gym[Box_2D]) (4.1.1)
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Collecting tensorflow==2.3.1
  Downloading tensorflow-2.3.1-cp37-cp37m-manylinux2010_x86_64.whl
(320.4 MB)
ent already satisfied: gym in /usr/local/lib/python3.7/dist-packages
(0.25.2)
Collecting keras-rl2
  Downloading keras_rl2-1.0.5-py3-none-any.whl (52 kB)
-manylinux1_x86_64.whl (2.9 MB)
ent already satisfied: absl-py>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(1.2.0)
Requirement already satisfied: protobuf>=3.9.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(3.17.3)
Collecting tensorflow-estimator<2.4.0,>=2.3.0
  Downloading tensorflow_estimator-2.3.0-py2.py3-none-any.whl (459 kB)
ent already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(1.1.0)
Requirement already satisfied: tensorboard<3,>=2.3.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(2.8.0)
Requirement already satisfied: wheel>=0.26 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(0.37.1)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(1.15.0)
Requirement already satisfied: grpcio>=1.8.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(1.48.1)
Requirement already satisfied: wrapt>=1.11.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(1.14.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(3.3.0)
Collecting numpy<1.19.0,>=1.16.0
  Downloading numpy-1.18.5-cp37-cp37m-manylinux1_x86_64.whl (20.1 MB)
ent already satisfied: astunparse==1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(1.6.3)
```

```
Requirement already satisfied: google-pasta>=0.1.8 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(0.2.0)
Requirement already satisfied: keras-preprocessing<1.2,>=1.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.3.1)
(1.1.2)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (57.4.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (3.4.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (1.8.1)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (1.0.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (2.23.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (0.4.6)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0
in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (0.6.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (1.35.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (0.2.8)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (4.2.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<3,>=2.3.0->tensorflow==2.3.1)
(1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (4.12.0)
Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<3,>=2.3.0->tensorflow==2.3.1) (4.1.1)
Requirement already satisfied: zipp>=0.5 in
```

```
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<3,>=2.3.0->tensorflow==2.3.1) (3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<3,>=2.3.0->tensorflow==2.3.1)
(0.4.8)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (2022.6.15)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<3,>=2.3.0->tensorflow==2.3.1) (2.10)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard<3,>=2.3.0-
>tensorflow==2.3.1) (3.2.0)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.7/dist-packages (from gym) (1.5.0)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.7/dist-packages (from gym) (0.0.8)
Installing collected packages: numpy, tensorflow-estimator, h5py,
gast, tensorflow, keras-rl2
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.6
    Uninstalling numpy-1.21.6:
      Successfully uninstalled numpy-1.21.6
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.8.0
    Uninstalling tensorflow-estimator-2.8.0:
      Successfully uninstalled tensorflow-estimator-2.8.0
  Attempting uninstall: h5py
    Found existing installation: h5py 3.1.0
    Uninstalling h5py-3.1.0:
      Successfully uninstalled h5py-3.1.0
  Attempting uninstall: gast
    Found existing installation: gast 0.5.3
    Uninstalling gast-0.5.3:
      Successfully uninstalled gast-0.5.3
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow
2.8.2+zzzcolab20220719082949
    Uninstalling tensorflow-2.8.2+zzzcolab20220719082949:
      Successfully uninstalled tensorflow-2.8.2+zzzcolab20220719082949
ERROR: pip's dependency resolver does not currently take into account
```

```
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
xarray-einstats 0.2.2 requires numpy>=1.21, but you have numpy 1.18.5
which is incompatible.
tables 3.7.0 requires numpy>=1.19.0, but you have numpy 1.18.5 which
is incompatible.
plotnine 0.8.0 requires numpy>=1.19.0, but you have numpy 1.18.5 which
is incompatible.
jaxlib 0.3.15+cuda11.cudnn805 requires numpy>=1.19, but you have numpy
1.18.5 which is incompatible.
jax 0.3.17 requires numpy>=1.20, but you have numpy 1.18.5 which is
incompatible.
cmdstanpy 1.0.7 requires numpy>=1.21, but you have numpy 1.18.5 which
is incompatible.
Successfully installed gast-0.3.3 h5py-2.10.0 keras-rl2-1.0.5 numpy-
1.18.5 tensorflow-2.3.1 tensorflow-estimator-2.3.0
```

{"pip_warning":{"packages":["numpy"]}}

Import the libraries

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

Create a DQN agent

Use the instruction below to prepare an agent

```
# 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 model
#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

The environment is given

```
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])

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dropout

# changes
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 = 0.001
        self.window_size = window_size
        self.gamma = 0.95
        #Explore/Exploit
        self.epsilon = 1
        self.epsilon_min = 0.01
        self.epsilon_decay = 0.995
        # self.model = self.build_model()
        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')) #[Input] -> Layer 1
        # #   Dense: Densely connected layer
https://keras.io/layers/core/
        # #   24: Number of neurons
        # #   input_dim: Number of input variables
        # #   activation: Rectified Linear Unit (relu) ranges >= 0
        # model.add(keras.layers.Dense(24, activation='relu')) #Layer
2 -> 3
        # model.add(keras.layers.Dense(self.nA, activation='linear'))
#Layer 3 -> 4
        # # model.add(keras.layers.Dense(self.nA,
activation='linear')) #Layer 4 -> [output]
        # #   Size has to match the output (different actions)
        # #   Linear activation on the last layer
        # model.compile(loss='mean_squared_error', #Loss function:
Mean Squared Error
        #
optimizer=keras.optimizers.Adam(lr=self.alpha)) #Optimaizer: Adam
(Feel free to check other options)

        model = Sequential()
        model.add(Dense(50, input_dim=self.window_size,
kernel_initializer="uniform", kernel_regularizer=l2(0.0002),
name="LAYER____1"))
        model.add(Activation("elu"))
        model.add(BatchNormalization())
        model.add(Dropout(0.25))
        model.add(Dense(35, kernel_initializer="uniform",
kernel_regularizer=l2(0.0002), name="LAYER____2"))
        model.add(Activation("elu"))
        model.add(BatchNormalization())
        model.add(Dropout(0.25))
        model.add(Dense(20, kernel_initializer="uniform",
kernel_regularizer=l2(0.0002), name="LAYER____3"))
        model.add(Activation("elu"))
        model.add(BatchNormalization())
        model.add(Dropout(0.25))
        model.add(Dense(self.nA, activation='linear'))

        opt = Adam(lr=0.0001, beta_1=0.5, decay=0.0002)
        model.compile(loss="mean_squared_error", optimizer=opt)

        return model

    def act(self, state):#act
        if np.random.rand() <= self.epsilon:
            return random.randrange(3) #Explore
        action_vals = self.model.predict(state) #Exploit: Use the NN
to predict the correct action from this state
        return np.argmax(action_vals[0])

    def test_action(self, state): #Exploit
        action_vals = self.model.predict(state)
        return np.argmax(action_vals[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): ## training the neural network
        #Execute the experience replay
        minibatch = random.sample( self.memory, batch_size ) #Randomly
sample from memory

        #Convert to numpy for speed by vectorization
        x = []
        y = []
        np_array = np.array(minibatch)
        st = np.zeros((0, self.nS)) #States
        nst = np.zeros( (0, self.nS) )#Next States
        for i in range(len(np_array)): #Creating the state and next
state np arrays
            st = np.append( st, np_array[i,0], axis=0)
            nst = np.append( nst, np_array[i,3], axis=0)
        st_predict = self.model.predict(st) #Here is the speedup! I
can predict on the ENTIRE batch
        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: Just assign reward much like
{* (not done) - QB[state][action]}
                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 for Keras Fit
        x_reshape = np.array(x).reshape(batch_size,self.nS)
        y_reshape = np.array(y)
        epoch_count = 1 #Epochs is the number or iterations
        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

```
import sys

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
#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_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 ("-----Episode: {} -----".format(e))
                  print ("Total Profit: " + formatPrice(total_profit))


            if len(agent.memory) > batch_size:
                  agent.expReplay(batch_size)
      # # if e % 10 == 0:
      if e % 10 == 0:
           agent.model.save("model_ep" + str(e))
      # agent.model.save("model_ep" + str(e))

#Fill the given information when prompted:
#Enter stock_name = GSPC_Training_Dataset
#window_size = 10
#Episode_count = 100 or it can be 10 or 20 or 30 and so on.
```

```
Usage: python train.py [stock] [window] [episodes]
Enter stock_name, window_size, Episode_countGSPC_Training_Dataset
10
30
Episode 0/30
--------------------------------
-----Episode: 0 -----
Total Profit: $865.92

WARNING:tensorflow:From
/usr/local/lib/python3.7/dist-packages/tensorflow/python/training/
tracking/tracking.py:111: Model.state_updates (from
tensorflow.python.keras.engine.training) is deprecated and will be
removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are
applied automatically.
WARNING:tensorflow:From
/usr/local/lib/python3.7/dist-packages/tensorflow/python/training/
tracking/tracking.py:111: Layer.updates (from
tensorflow.python.keras.engine.base_layer) is deprecated and will be
removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are
applied automatically.

Episode 1/30
--------------------------------
-----Episode: 1 -----
Total Profit: $7141.67
Episode 2/30
--------------------------------
-----Episode: 2 -----
Total Profit: $6871.84
Episode 3/30
--------------------------------
-----Episode: 3 -----
Total Profit: $6425.62
Episode 4/30
--------------------------------
-----Episode: 4 -----
```

```
Total Profit: $7306.68
Episode 5/30
--------------------------------
-----Episode: 5 -----
Total Profit: $7161.61
Episode 6/30
--------------------------------
-----Episode: 6 -----
Total Profit: $6990.42
Episode 7/30
--------------------------------
-----Episode: 7 -----
Total Profit: $7420.58
Episode 8/30
--------------------------------
-----Episode: 8 -----
Total Profit: $5741.95
Episode 9/30
--------------------------------
-----Episode: 9 -----
Total Profit: $7061.20
Episode 10/30
--------------------------------
-----Episode: 10 -----
Total Profit: $5241.04
Episode 11/30
--------------------------------
-----Episode: 11 -----
Total Profit: $7092.10
Episode 12/30
--------------------------------
-----Episode: 12 -----
Total Profit: $6816.25
Episode 13/30
--------------------------------
-----Episode: 13 -----
Total Profit: $6431.38
Episode 14/30
--------------------------------
-----Episode: 14 -----
Total Profit: $6240.14
Episode 15/30
--------------------------------
-----Episode: 15 -----
Total Profit: $5827.87
Episode 16/30
--------------------------------
-----Episode: 16 -----
Total Profit: $5912.93
Episode 17/30
```

```
--------------------------------
-----Episode: 17 -----
Total Profit: $4592.13
Episode 18/30
--------------------------------
-----Episode: 18 -----
Total Profit: $6146.69
Episode 19/30
--------------------------------
-----Episode: 19 -----
Total Profit: $6504.60
Episode 20/30
--------------------------------
-----Episode: 20 -----
Total Profit: $5526.26
Episode 21/30
--------------------------------
-----Episode: 21 -----
Total Profit: $6493.13
Episode 22/30
--------------------------------
-----Episode: 22 -----
Total Profit: $6761.81
Episode 23/30
--------------------------------
-----Episode: 23 -----
Total Profit: $6308.82
Episode 24/30
--------------------------------
-----Episode: 24 -----
Total Profit: $6131.78
Episode 25/30
--------------------------------
-----Episode: 25 -----
Total Profit: $6599.88
Episode 26/30
--------------------------------
-----Episode: 26 -----
Total Profit: $5144.48
Episode 27/30
--------------------------------
-----Episode: 27 -----
Total Profit: $5699.61
Episode 28/30
--------------------------------
-----Episode: 28 -----
Total Profit: $5480.21
Episode 29/30
--------------------------------
-----Episode: 29 -----
```

```
Total Profit: $5844.89
Episode 30/30
--------------------------------
-----Episode: 30 -----
Total Profit: $6726.90
```

```
# save all the model to my google drive
!scp -r /content/model_ep*
/content/drive/MyDrive/datasets/simplilearn_RL_stock_trading/model_fil
e
```

Evaluate the model and agent

```
import sys
from tensorflow.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
#Model_name = respective model name

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    ("-----Episode: {} -----".format(e))
            print (stock_name + " Total Profit: " +
formatPrice(total_profit))

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

# GSPC_Evaluation_Dataset
# /content/model_ep30
```

```
Enter Stock_name, Model_nameGSPC_Evaluation_Dataset
/content/model_ep30
Buy: $1271.87
Buy: $1276.56
Buy: $1273.85
Buy: $1271.50
Buy: $1269.75
Buy: $1285.96
Sell: $1293.24 | Profit: $21.37
Buy: $1295.02
Sell: $1281.92 | Profit: $5.36
Sell: $1280.26 | Profit: $6.41
Buy: $1283.35
Buy: $1291.18
Buy: $1296.63
Sell: $1276.34 | Profit: $4.84
Sell: $1286.12 | Profit: $16.37
Sell: $1307.10 | Profit: $21.14
Buy: $1310.87
Buy: $1319.05
Sell: $1324.57 | Profit: $29.55
Buy: $1320.88
Sell: $1321.87 | Profit: $38.52
Buy: $1329.15
Sell: $1328.01 | Profit: $36.83
```

```
Sell: $1343.01 | Profit: $46.38
Sell: $1315.44 | Profit: $4.57
Sell: $1307.40 | Profit: -$11.65
Buy: $1306.10
Buy: $1319.88
Sell: $1327.22 | Profit: $6.34
Sell: $1306.33 | Profit: -$22.82
Sell: $1330.97 | Profit: $24.87
Sell: $1321.15 | Profit: $1.27
Buy: $1310.13
Buy: $1321.82
Buy: $1320.02
Buy: $1295.11
Buy: $1304.28
Sell: $1296.39 | Profit: -$13.74
Buy: $1281.87
Sell: $1256.88 | Profit: -$64.94
Buy: $1298.38
Sell: $1293.77 | Profit: -$26.25
Sell: $1297.54 | Profit: $2.43
Buy: $1319.44
Sell: $1328.26 | Profit: $23.98
Buy: $1325.83
Sell: $1332.41 | Profit: $50.54
Buy: $1332.87
Sell: $1335.54 | Profit: $37.16
Buy: $1333.51
Buy: $1324.46
Sell: $1314.52 | Profit: -$4.92
Sell: $1319.68 | Profit: -$6.15
Sell: $1305.14 | Profit: -$27.73
Buy: $1312.62
Sell: $1330.36 | Profit: -$3.15
Buy: $1337.38
Sell: $1347.24 | Profit: $22.78
Buy: $1355.66
Buy: $1360.48
Buy: $1363.61
Sell: $1361.22 | Profit: $48.60
Buy: $1356.62
Sell: $1347.32 | Profit: $9.94
Sell: $1340.20 | Profit: -$15.46
Buy: $1346.29
Buy: $1357.16
Sell: $1342.08 | Profit: -$18.40
Sell: $1328.98 | Profit: -$34.63
Sell: $1340.68 | Profit: -$15.94
Buy: $1343.60
Buy: $1333.27
Buy: $1317.37
```

```
Sell: $1320.47 | Profit: -$25.82
Buy: $1331.10
Sell: $1286.17 | Profit: -$70.99
Buy: $1279.56
Sell: $1271.83 | Profit: -$71.77
Buy: $1287.87
Buy: $1267.64
Buy: $1271.50
Buy: $1295.52
Sell: $1287.14 | Profit: -$46.13
Buy: $1296.67
Buy: $1320.64
Sell: $1339.67 | Profit: $22.30
Buy: $1337.88
Sell: $1339.22 | Profit: $8.12
Buy: $1353.22
Sell: $1319.49 | Profit: $39.93
Sell: $1313.64 | Profit: $25.77
Sell: $1317.72 | Profit: $50.08
Sell: $1308.87 | Profit: $37.37
Buy: $1316.14
Sell: $1326.73 | Profit: $31.21
Sell: $1325.84 | Profit: $29.17
Sell: $1343.80 | Profit: $23.16
Sell: $1345.02 | Profit: $7.14
Sell: $1337.43 | Profit: -$15.79
Buy: $1331.94
Sell: $1304.89 | Profit: -$11.25
Sell: $1300.67 | Profit: -$31.27
Buy: $1286.94
Buy: $1260.34
Sell: $1199.38 | Profit: -$87.56
Sell: $1172.53 | Profit: -$87.81
Buy: $1178.81
Buy: $1193.89
Buy: $1140.65
Buy: $1123.82
Sell: $1162.35 | Profit: -$16.46
Sell: $1159.27 | Profit: -$34.62
Sell: $1176.80 | Profit: $36.15
Sell: $1210.08 | Profit: $86.26
Buy: $1204.42
Sell: $1165.24 | Profit: -$39.18
Buy: $1198.62
Buy: $1185.90
Buy: $1172.87
Buy: $1209.11
Buy: $1204.09
Sell: $1202.09 | Profit: $3.47
Sell: $1129.56 | Profit: -$56.34
```

```
Sell: $1136.43 | Profit: -$36.44
Buy: $1175.38
Buy: $1151.06
Buy: $1160.40
Buy: $1131.42
Sell: $1123.95 | Profit: -$85.16
Buy: $1144.03
Buy: $1155.46
Sell: $1207.25 | Profit: $3.16
Sell: $1203.66 | Profit: $28.28
Buy: $1224.58
Sell: $1209.88 | Profit: $58.82
Buy: $1215.39
Sell: $1254.19 | Profit: $93.79
Buy: $1229.05
Buy: $1242.00
Sell: $1284.59 | Profit: $153.17
Buy: $1253.30
Buy: $1237.90
Buy: $1261.15
Sell: $1253.23 | Profit: $109.20
Sell: $1261.12 | Profit: $105.66
Sell: $1229.10 | Profit: $4.52
Sell: $1251.78 | Profit: $36.39
Sell: $1257.81 | Profit: $28.76
Buy: $1216.13
Buy: $1192.98
Sell: $1188.04 | Profit: -$53.96
Sell: $1161.79 | Profit: -$91.51
Sell: $1158.67 | Profit: -$79.23
Buy: $1195.19
Sell: $1246.96 | Profit: -$14.19
Sell: $1244.58 | Profit: $28.45
Sell: $1261.01 | Profit: $68.03
Buy: $1234.35
Sell: $1255.19 | Profit: $60.00
Buy: $1236.47
Sell: $1225.73 | Profit: -$8.62
Sell: $1211.82 | Profit: -$24.65
Buy: $1215.75
Buy: $1219.66
Sell: $1243.72 | Profit: $27.97
Sell: $1254.00 | Profit: $34.34
--------------------------------
GSPC_Evaluation_Dataset Total Profit: $445.39
```

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