11.double-duel-q-learning-agent

September 29, 2021

[1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

[2]:
```
df = pd.read_csv('../dataset/GOOG-year.csv')
df.head()
```

[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-11-02	778.200012	781.650024	763.450012	768.700012	768.700012	1872400
1	2016-11-03	767.250000	769.950012	759.030029	762.130005	762.130005	1943200
2	2016-11-04	750.659973	770.359985	750.560974	762.020020	762.020020	2134800
3	2016-11-07	774.500000	785.190002	772.549988	782.520020	782.520020	1585100
4	2016-11-08	783.400024	795.632996	780.190002	790.510010	790.510010	1350800

[3]:
```
from collections import deque
import random

class Model:
    def __init__(self, input_size, output_size, layer_size, learning_rate):
        self.X = tf.placeholder(tf.float32, (None, input_size))
        self.Y = tf.placeholder(tf.float32, (None, output_size))
        feed = tf.layers.dense(self.X, layer_size, activation = tf.nn.relu)
        tensor_action, tensor_validation = tf.split(feed,2,1)
        feed_action = tf.layers.dense(tensor_action, output_size)
        feed_validation = tf.layers.dense(tensor_validation, 1)
        self.logits = feed_validation + tf.subtract(feed_action,tf.
↪reduce_mean(feed_action,axis=1,keep_dims=True))
```

```
        self.cost = tf.reduce_sum(tf.square(self.Y - self.logits))
        self.optimizer = tf.train.AdamOptimizer(learning_rate = learning_rate).
↪minimize(self.cost)

class Agent:

    LEARNING_RATE = 0.003
    BATCH_SIZE = 32
    LAYER_SIZE = 500
    OUTPUT_SIZE = 3
    EPSILON = 0.5
    DECAY_RATE = 0.005
    MIN_EPSILON = 0.1
    GAMMA = 0.99
    MEMORIES = deque()
    COPY = 1000
    T_COPY = 0
    MEMORY_SIZE = 300

    def __init__(self, state_size, window_size, trend, skip):
        self.state_size = state_size
        self.window_size = window_size
        self.half_window = window_size // 2
        self.trend = trend
        self.skip = skip
        tf.reset_default_graph()
        self.model = Model(self.state_size, self.OUTPUT_SIZE, self.LAYER_SIZE,␣
↪self.LEARNING_RATE)
        self.model_negative = Model(self.state_size, self.OUTPUT_SIZE, self.
↪LAYER_SIZE, self.LEARNING_RATE)
        self.sess = tf.InteractiveSession()
        self.sess.run(tf.global_variables_initializer())
        self.trainable = tf.trainable_variables()

    def _assign(self):
        for i in range(len(self.trainable)//2):
            assign_op = self.trainable[i+len(self.trainable)//2].assign(self.
↪trainable[i])
            self.sess.run(assign_op)

    def _memorize(self, state, action, reward, new_state, done):
        self.MEMORIES.append((state, action, reward, new_state, done))
        if len(self.MEMORIES) > self.MEMORY_SIZE:
            self.MEMORIES.popleft()

    def _select_action(self, state):
        if np.random.rand() < self.EPSILON:
```

```
            action = np.random.randint(self.OUTPUT_SIZE)
        else:
            action = self.get_predicted_action([state])
        return action

    def _construct_memories(self, replay):
        states = np.array([a[0] for a in replay])
        new_states = np.array([a[3] for a in replay])
        Q = self.predict(states)
        Q_new = self.predict(new_states)
        Q_new_negative = self.sess.run(self.model_negative.logits,␣
↪feed_dict={self.model_negative.X:new_states})
        replay_size = len(replay)
        X = np.empty((replay_size, self.state_size))
        Y = np.empty((replay_size, self.OUTPUT_SIZE))
        for i in range(replay_size):
            state_r, action_r, reward_r, new_state_r, done_r = replay[i]
            target = Q[i]
            target[action_r] = reward_r
            if not done_r:
                target[action_r] += self.GAMMA * Q_new_negative[i, np.
↪argmax(Q_new[i])]
            X[i] = state_r
            Y[i] = target
        return X, Y

    def predict(self, inputs):
        return self.sess.run(self.model.logits, feed_dict={self.model.X:inputs})

    def get_predicted_action(self, sequence):
        prediction = self.predict(np.array(sequence))[0]
        return np.argmax(prediction)

    def get_state(self, t):
        window_size = self.window_size + 1
        d = t - window_size + 1
        block = self.trend[d : t + 1] if d >= 0 else -d * [self.trend[0]] +␣
↪self.trend[0 : t + 1]
        res = []
        for i in range(window_size - 1):
            res.append(block[i + 1] - block[i])
        return np.array(res)

    def buy(self, initial_money):
        starting_money = initial_money
        states_sell = []
        states_buy = []
```

```
        inventory = []
        state = self.get_state(0)
        for t in range(0, len(self.trend) - 1, self.skip):
            action = self._select_action(state)
            next_state = self.get_state(t + 1)

            if action == 1 and initial_money >= self.trend[t]:
                inventory.append(self.trend[t])
                initial_money -= self.trend[t]
                states_buy.append(t)
                print('day %d: buy 1 unit at price %f, total balance %f'% (t,␣
↪self.trend[t], initial_money))

            elif action == 2 and len(inventory):
                bought_price = inventory.pop(0)
                initial_money += self.trend[t]
                states_sell.append(t)
                try:
                    invest = ((close[t] - bought_price) / bought_price) * 100
                except:
                    invest = 0
                print(
                    'day %d, sell 1 unit at price %f, investment %f %%, total␣
↪balance %f,'
                    % (t, close[t], invest, initial_money)
                )

            state = next_state
        invest = ((initial_money - starting_money) / starting_money) * 100
        total_gains = initial_money - starting_money
        return states_buy, states_sell, total_gains, invest


    def train(self, iterations, checkpoint, initial_money):
        for i in range(iterations):
            total_profit = 0
            inventory = []
            state = self.get_state(0)
            starting_money = initial_money
            for t in range(0, len(self.trend) - 1, self.skip):
                if (self.T_COPY + 1) % self.COPY == 0:
                    self._assign()

                action = self._select_action(state)
                next_state = self.get_state(t + 1)

                if action == 1 and starting_money >= self.trend[t]:
```

```
                    inventory.append(self.trend[t])
                    starting_money -= self.trend[t]

                elif action == 2 and len(inventory) > 0:
                    bought_price = inventory.pop(0)
                    total_profit += self.trend[t] - bought_price
                    starting_money += self.trend[t]

                invest = ((starting_money - initial_money) / initial_money)

                self._memorize(state, action, invest, next_state,␣
↪starting_money < initial_money)
                batch_size = min(len(self.MEMORIES), self.BATCH_SIZE)
                state = next_state
                replay = random.sample(self.MEMORIES, batch_size)
                X, Y = self._construct_memories(replay)

                cost, _ = self.sess.run([self.model.cost, self.model.optimizer],
                                        feed_dict={self.model.X: X, self.model.
↪Y:Y})
                self.T_COPY += 1
                self.EPSILON = self.MIN_EPSILON + (1.0 - self.MIN_EPSILON) * np.
↪exp(-self.DECAY_RATE * i)
            if (i+1) % checkpoint == 0:
                print('epoch: %d, total rewards: %f.3, cost: %f, total money:␣
↪%f'%(i + 1, total_profit, cost,
                                                                              ␣
↪  starting_money))
```

[4]:
```
close = df.Close.values.tolist()
initial_money = 10000
window_size = 30
skip = 1
batch_size = 32
agent = Agent(state_size = window_size,
              window_size = window_size,
              trend = close,
              skip = skip)
agent.train(iterations = 200, checkpoint = 10, initial_money = initial_money)
```

```
WARNING:tensorflow:From <ipython-input-3-42f2d1e26a9d>:12: calling reduce_mean
(from tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be
removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
epoch: 10, total rewards: 1486.684997.3, cost: 0.694152, total money:
10514.124999
```

```
epoch: 20, total rewards: 313.279660.3, cost: 0.878157, total money: 8354.909665
epoch: 30, total rewards: 752.595089.3, cost: 0.320037, total money:
10752.595089
epoch: 40, total rewards: 1159.299987.3, cost: 0.318166, total money:
10186.739989
epoch: 50, total rewards: 993.220279.3, cost: 0.391151, total money: 4149.310245
epoch: 60, total rewards: 1616.499880.3, cost: 0.307440, total money:
9630.939883
epoch: 70, total rewards: 941.484560.3, cost: 0.332979, total money: 6969.054506
epoch: 80, total rewards: 904.899903.3, cost: 0.718111, total money: 1132.559876
epoch: 90, total rewards: 346.619873.3, cost: 0.482044, total money: 542.599852
epoch: 100, total rewards: 141.554626.3, cost: 0.238426, total money:
6115.974608
epoch: 110, total rewards: -159.529845.3, cost: 0.202412, total money:
8852.270143
epoch: 120, total rewards: -37.579779.3, cost: 0.433529, total money:
8945.780206
epoch: 130, total rewards: 1049.544800.3, cost: 0.408910, total money:
8099.664795
epoch: 140, total rewards: 59.114809.3, cost: 0.028664, total money: 7098.904848
epoch: 150, total rewards: 96.424866.3, cost: 0.070552, total money: 9079.784851
epoch: 160, total rewards: 74.179754.3, cost: 0.044092, total money:
10074.179754
epoch: 170, total rewards: 80.999883.3, cost: 0.018813, total money: 8047.249883
epoch: 180, total rewards: 62.700011.3, cost: 0.083292, total money:
10062.700011
epoch: 190, total rewards: 70.424991.3, cost: 0.013884, total money: 9053.315006
epoch: 200, total rewards: 10.620115.3, cost: 0.030838, total money:
10010.620115
```

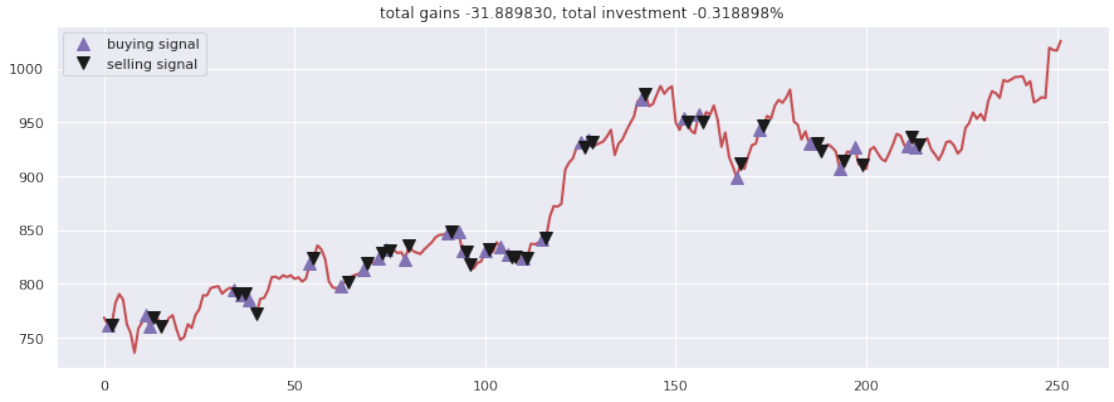
[5]:
```
states_buy, states_sell, total_gains, invest = agent.buy(initial_money =␣
↪initial_money)
```

```
day 1: buy 1 unit at price 762.130005, total balance 9237.869995
day 2, sell 1 unit at price 762.020020, investment -0.014431 %, total balance
9999.890015,
day 11: buy 1 unit at price 771.229980, total balance 9228.660035
day 12: buy 1 unit at price 760.539978, total balance 8468.120057
day 13, sell 1 unit at price 769.200012, investment -0.263212 %, total balance
9237.320069,
day 15, sell 1 unit at price 760.989990, investment 0.059170 %, total balance
9998.310059,
day 34: buy 1 unit at price 794.559998, total balance 9203.750061
day 35, sell 1 unit at price 791.260010, investment -0.415323 %, total balance
9995.010071,
day 36: buy 1 unit at price 789.909973, total balance 9205.100098
day 37, sell 1 unit at price 791.549988, investment 0.207620 %, total balance
9996.650086,
```

day 38: buy 1 unit at price 785.049988, total balance 9211.600098
day 40, sell 1 unit at price 771.820007, investment -1.685241 %, total balance 9983.420105,
day 54: buy 1 unit at price 819.309998, total balance 9164.110107
day 55, sell 1 unit at price 823.869995, investment 0.556566 %, total balance 9987.980102,
day 62: buy 1 unit at price 798.530029, total balance 9189.450073
day 64, sell 1 unit at price 801.340027, investment 0.351896 %, total balance 9990.790100,
day 68: buy 1 unit at price 813.669983, total balance 9177.120117
day 69, sell 1 unit at price 819.239990, investment 0.684554 %, total balance 9996.360107,
day 72: buy 1 unit at price 824.159973, total balance 9172.200134
day 73, sell 1 unit at price 828.070007, investment 0.474427 %, total balance 10000.270141,
day 74: buy 1 unit at price 831.659973, total balance 9168.610168
day 75, sell 1 unit at price 830.760010, investment -0.108213 %, total balance 9999.370178,
day 79: buy 1 unit at price 823.210022, total balance 9176.160156
day 80, sell 1 unit at price 835.239990, investment 1.461349 %, total balance 10011.400146,
day 90: buy 1 unit at price 847.200012, total balance 9164.200134
day 91, sell 1 unit at price 848.780029, investment 0.186499 %, total balance 10012.980163,
day 93: buy 1 unit at price 848.400024, total balance 9164.580139
day 94: buy 1 unit at price 830.460022, total balance 8334.120117
day 95, sell 1 unit at price 829.590027, investment -2.217114 %, total balance 9163.710144,
day 96, sell 1 unit at price 817.580017, investment -1.550948 %, total balance 9981.290161,
day 100: buy 1 unit at price 831.409973, total balance 9149.880188
day 101, sell 1 unit at price 831.500000, investment 0.010828 %, total balance 9981.380188,
day 104: buy 1 unit at price 834.570007, total balance 9146.810181
day 106: buy 1 unit at price 827.880005, total balance 8318.930176
day 107, sell 1 unit at price 824.669983, investment -1.186242 %, total balance 9143.600159,
day 108, sell 1 unit at price 824.729980, investment -0.380493 %, total balance 9968.330139,
day 110: buy 1 unit at price 824.320007, total balance 9144.010132
day 111, sell 1 unit at price 823.559998, investment -0.092198 %, total balance 9967.570130,
day 115: buy 1 unit at price 841.650024, total balance 9125.920106
day 116, sell 1 unit at price 843.190002, investment 0.182971 %, total balance 9969.110108,
day 125: buy 1 unit at price 931.659973, total balance 9037.450135
day 126, sell 1 unit at price 927.130005, investment -0.486225 %, total balance 9964.580140,

```
day 127: buy 1 unit at price 934.299988, total balance 9030.280152
day 128, sell 1 unit at price 932.169983, investment -0.227979 %, total balance
9962.450135,
day 141: buy 1 unit at price 971.469971, total balance 8990.980164
day 142, sell 1 unit at price 975.880005, investment 0.453955 %, total balance
9966.860169,
day 152: buy 1 unit at price 953.400024, total balance 9013.460145
day 153, sell 1 unit at price 950.760010, investment -0.276905 %, total balance
9964.220155,
day 156: buy 1 unit at price 957.369995, total balance 9006.850160
day 157, sell 1 unit at price 950.630005, investment -0.704011 %, total balance
9957.480165,
day 166: buy 1 unit at price 898.700012, total balance 9058.780153
day 167, sell 1 unit at price 911.710022, investment 1.447648 %, total balance
9970.490175,
day 172: buy 1 unit at price 943.830017, total balance 9026.660158
day 173, sell 1 unit at price 947.159973, investment 0.352813 %, total balance
9973.820131,
day 185: buy 1 unit at price 930.500000, total balance 9043.320131
day 186: buy 1 unit at price 930.830017, total balance 8112.490114
day 187, sell 1 unit at price 930.390015, investment -0.011820 %, total balance
9042.880129,
day 188, sell 1 unit at price 923.650024, investment -0.771354 %, total balance
9966.530153,
day 193: buy 1 unit at price 907.239990, total balance 9059.290163
day 194, sell 1 unit at price 914.390015, investment 0.788107 %, total balance
9973.680178,
day 197: buy 1 unit at price 926.960022, total balance 9046.720156
day 199, sell 1 unit at price 910.669983, investment -1.757362 %, total balance
9957.390139,
day 211: buy 1 unit at price 927.809998, total balance 9029.580141
day 212, sell 1 unit at price 935.950012, investment 0.877336 %, total balance
9965.530153,
day 213: buy 1 unit at price 926.500000, total balance 9039.030153
day 214, sell 1 unit at price 929.080017, investment 0.278469 %, total balance
9968.110170,
```

[6]:
```
fig = plt.figure(figsize = (15,5))
plt.plot(close, color='r', lw=2.)
plt.plot(close, '^', markersize=10, color='m', label = 'buying signal',␣
 ↪markevery = states_buy)
plt.plot(close, 'v', markersize=10, color='k', label = 'selling signal',␣
 ↪markevery = states_sell)
plt.title('total gains %f, total investment %f%%'%(total_gains, invest))
plt.legend()
plt.show()
```


total gains -31.889830, total investment -0.318898%
buying signal
selling signal
1000
950
900
850
800
750
0
50
100
150
200
250

[ ]: