Stock Trade Simulation

Nick Guan, Yuchen Luo, Grace Stewart, Timothy Yang

Introduction

Background:

Financial institutions and hedge funds run cutting-edge algorithmic trading software utilizing deep learning

Motivation:

- We want to see how that all works and make our own similar algorithms
- Identify / capitalize on what works well, and take note of the limitations

General Approach

1. Scrape data from Reddit, Twitter and media outlets regarding the number of times a company ticker has been mentioned
2. Generate a sentiment score per company per day ~ a negative or positive sentiment
3. Implement a deep learning model (q-learning) to run trading simulations for the companies using stock data and the sentiment scores

Twitter

Approach

Utilizing the Python libraries Tweepy and VaderSentiment:

- Get all related Tweets by using the search_full_archive function in Twitter
- Create list of company tickers containing all tickers from Nasdaq, Amex, and NYSE
- Generate sentiment score per tweet per company for top 3, then generate average score per company per day
- Add this data to a dataframe where each row represents each day per company

Experiments

- Level of API Access - .search() vs .search_full_archive()
- Search Term
- Dates

Results::

- Inaccurate per graphs

News Sentiment

- BeautifulSoup
  - Extracted all article links from CNBC for every day from 2019-2021
  - Extracted text from each article link, separated per passage
  - Found all references to target companies (GME, TSLA, S&P500) if any
- Vader
  - For each passage that referenced target company, generate sentiment score
  - Keep track of sentiment across all target companies every day
  - Only modify sentiment values of companies mentioned in passage

Company	Date	Sentiment	# Mentions
GME	1/2/19	0	0
TSLA	1/2/19	3.161	15
S&P500	1/2/19	0	0
GME	1/3/19	0	0
TSLA	1/3/19	-0.5994	1
S&P500	1/3/19	0	0
GME	1/4/19	0.2684	3
TSLA	1/4/19	4.4913	42
S&P500	1/4/19	0	0

News Sentiment

TSLA
$61.22 -> $62.02

TLSA
$61.20 -> $63.54

This makes sense.

Lots of positive mentions? Buy.
Lots of negative mentions? Sell.

Time of day, price movement, sentiment score all matter.

Company	Date	Sentiment	# Mentions
GME	1/2/19	0	0
TSLA	1/2/19	3.161	15
S&P500	1/2/19	0	0
GME	1/3/19	0	0
TSLA	1/3/19	-0.5994	1
S&P500	1/3/19	0	0
GME	1/4/19	0.2684	3
TSLA	1/4/19	4.4913	42
S&P500	1/4/19	0	0

Reddit – approach

Utilizing the Python libraries psaw for reddit and flair for sentiment score

- Obtain the top mentioned company tickers
- Retrieve all related subReddit posts in a specific date by using the search_submission in psaw library
- Generate sentiment score per day by flair, for a specific time range
- Output a dataframe that contains the date and corresponding sentiment score

Reddit – result

Date	DD	M	K	FOR	G	TSLA	SP500
2019-01-01 00:00:00	-0.0105	0.9878	0	-0.1237	0	0.9211	-0.0526
2019-01-02 00:00:00	-0.2582	0.2108	0.3302	-0.2035	0.8237	-0.2050	-0.1386
2019-01-03 00:00:00	-0.0311	-0.6825	-0.1870	-0.1816	0.3176	-0.4517	0.5352
2019-01-04 00:00:00	-0.0750	-0.0878	-0.1828	-0.1112	0	-0.0177	0

Model Inputs

We used

- Adj Close Price
- Volume
- Sentiment Score
- Low Price
- Relative Strength Index

Date	Open	High	Low	Close	Adj Close	Volume	RSI	Sentiment
2019-01-09	79.105331	79.788506	78.165977	79.162262	75.593346	5250445	0.000000	-0.301708
2019-01-10	78.265602	79.333054	77.582436	79.247658	75.674896	7244586	100.000000	-0.374780
2019-01-11	78.621422	79.304588	77.867088	78.678352	75.131256	5030951	12.225705	0.079539
2019-01-14	77.610901	79.389984	77.269318	78.820679	75.267151	6295785	29.003316	-0.329321
2019-01-15	78.052116	79.005707	77.397408	78.294067	74.764290	4400642	16.463773	-0.479455

Deep Q-Learning

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

- Agent that can buy/sell
- Wants to maximize future rewards (expected return)
- Want to model a day trader (risky behavior)

Ex: agent parameters

Learning rate	Gamma	Epsilon	Epsilon Min	Epsilon Dec
1e-4	.90	1.0	0.15	2e-5

The model network

```
class LinearDeepQNetwork(nn.Module):
  def __init__(self, lr, n_actions, input_dims):
    super(LinearDeepQNetwork, self).__init__()
    self.fc1 = nn.Linear(*input_dims, 64)
    self.fc2 = nn.Linear(64, 32)
    self.fc3 = nn.Linear(32, 16)
    self.fc4 = nn.Linear(16, n_actions)
    self.optimizer = optim.Adam(self.parameters(), lr=lr)
    self.loss = nn.MSELoss()

    self.device = T.device('cuda:0' if T.cuda.is_available() else 'cpu')
    self.to(self.device)

  def forward(self, data):
    layer1 = F.relu(self.fc1(data), inplace=True)
    layer2 = F.relu(self.fc2(layer1), inplace=True)
    layer3 = F.relu(self.fc3(layer2), inplace=True)
    layer4 = self.fc4(layer3)
    return layer4
```

Model Architecture:

- Linear Deep Q Network
- Hidden layers
- Adam optimizer
- Input -> Output of size 2 (buy/sell)

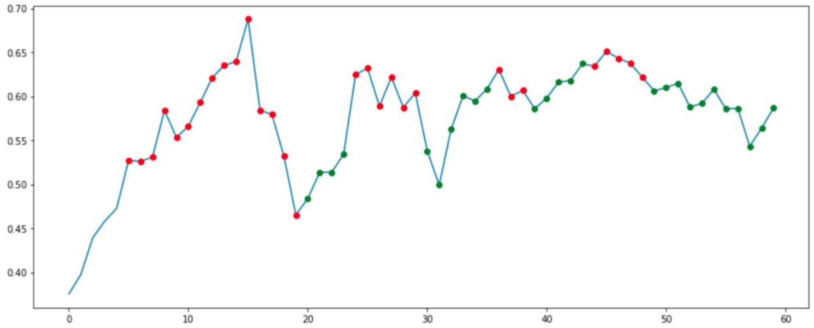
```
class LinearDeepQNetwork(nn.Module):
  def __init__(self, lr, n_actions, input_dims):
    super(LinearDeepQNetwork, self).__init__()
    self.fc1 = nn.Linear(*input_dims, 64)
    self.fc2 = nn.Linear(64, 32)
    self.fc3 = nn.Linear(32, 16)
    self.fc4 = nn.Linear(16, n_actions)
    self.optimizer = optim.Adam(self.parameters(), lr=lr)
    self.loss = nn.MSELoss()
    self.elu = nn.ELU()
    self.device = T.device('cuda:0' if T.cuda.is_available() else 'cpu')
    self.to(self.device)

  def forward(self, data):
    layer1 = self.elu(self.fc1(data))
    layer2 = self.elu(self.fc2(layer1))
    layer3 = self.elu(self.fc3(layer2))
    layer4 = self.fc4(layer3)
    return layer4
```

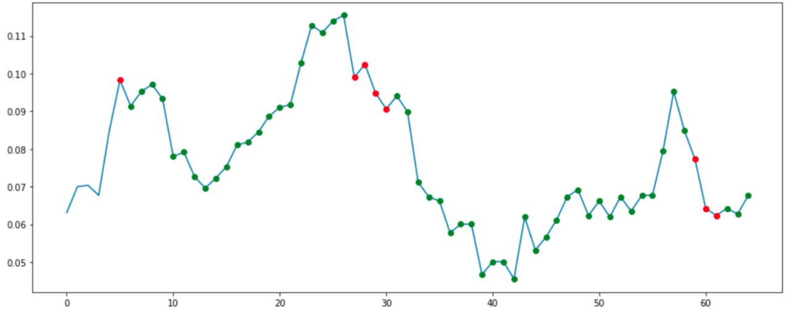
Difficulties:

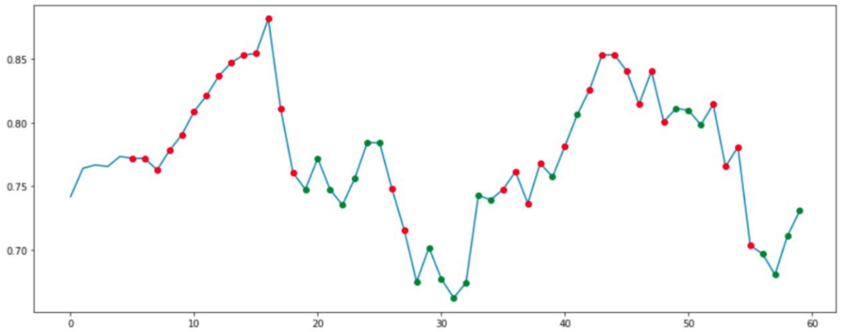
- Testing different layer sizes
- Determining optimal learning rate
- Overfitting!
  - Activations: relu -> elu
  - Fixing dead weights

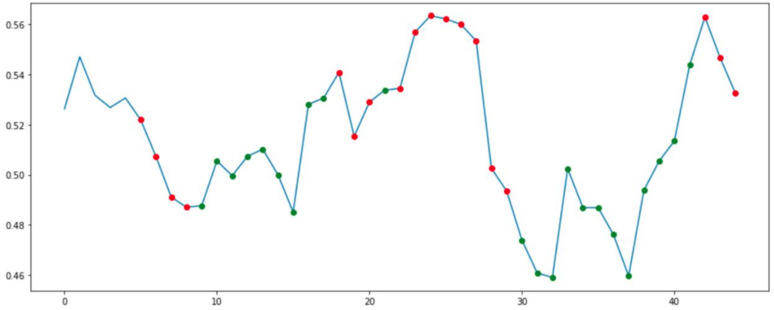
Results

$TSLA: Total Profit 1.4926	$M: Total Profit: 0.8961
$SPY: Total Profit 1.1276	$DD: Total Profit 1.2611









Future Work

- Try new models (using LSTM)
- Implement Q-learning improvements
  - Experience Replay
  - DQNs
- Generate more accurate sentiment scores
- Exploratory data analysis to isolate more inputs
  - E.g. other technical indicators