AI for Trading

1st Utkarsh Baviskar
Dept. Of IT
(VIIT Kondhwa)
Savitri Bai Phule Pune University
Pune, India
utkarsh.21810834@viit.ac.in

2nd Saurabh Muneshwar
Dept. Of IT
(VIIT Kondhwa)
Savitri Bai Phule Pune University
Pune, India
saurabh.muneshwar@viit.ac.in

Dept. Of IT
(VIIT Kondhwa)
Savitri Bai Phule Pune University
Pune, India
qasim.21810959@viit.ac.in

***Abstract*— The following Project gives the idea of how Ai can be used for trading and predicting the future of a stock based on it's past record**

***Keywords*—Stocks , Money , Business , People , Prediction , AI , Artificial Intelligence , Cash Machine Learning**

INTRODUCTION

Artificial intelligence (AI) is rapidly transforming the global financial services industry. As a group of related technologies that include machine learning (ML) and deep learning (DL), AI has the potential to disrupt and refine the existing financial services industry. I review the extant academic, practitioner and policy related AI literature. I also detail the AI, ML and DL taxonomy as well as their various applications in the financial services industry. A literature survey of AI and financial services cannot ignore the econometric aspects and their implications. ML methods are all about algorithms, rather than asymptotic statistical processes. Unlike maximum likelihood estimation, ML's framework is less unified. To that end, I will discuss the ML approaches of unsupervised and supervised learning

Use Cases for AI in Financial Trading

Artificial Intelligence (AI) and Machine Learning (ML) have already been used in financial trading in areas such as asset price prediction, simulation of the stock market, investment strategy development, portfolio management and diversification, technical analysis of financial charts data, and more. We will discuss the basic use cases for AI in financial trading and show how the integration of this technology can lead to a qualitatively new model of financial trading close to Artificial General Intelligence (AGI).

AI-based Asset Price Prediction

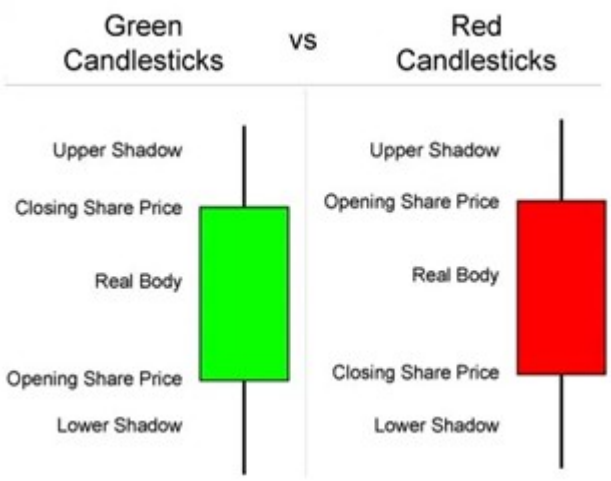
Predicting the movement of asset prices has been one of the most challenging tasks of investment decision-making and analysis. Over the past several decades, the financial sector has developed complex statistical and probabilistic methods for technical analysis which search for patterns in the financial time series to predict the future movement of prices. However, until very recently, most approaches to technical analysis such as Japanese candlesticks, heavily relied on the trader's intuition, financial rules of thumb, and other pseudo-scientific practical approaches

The arrival of supervised ML and Reinforcement Learning (RL) propelled by vast datasets of financial data and facilitated by the cheap and fast computing resources makes a huge difference in the accuracy of asset prices prediction.

Supervised learning is a sub-discipline of Machine Learning that uses accurately labeled and regularized data (i.e structured data) to learn the optimal model/hypothesis that explains patterns and relationships in this data. If applied to financial trading, supervised learning algorithms search for patterns in the financial time series to derive a generic model of the price dynamics over time.

The end result is achieved through the iterative optimization/fitting of the initial hypothesis (normally some parametric function) to the training data until it fully explains the underlying pattern/model that generates such data. The model derived in this iterative optimization process can be then applied to price prediction using real-world financial data. The main assumption of this supervised learning model is that if it can explain the past movements of prices, it can also predict the future prices.

As it turns out, supervised models for financial trading are quite strong in predicting short-term price dynamics. If the training data fed to the supervised learning algorithm belongs to one homogeneous market trend (e.g one-week trading period) and the trained model is used for the real-



world data that falls within the same trend, we can expect the supervised model to predict prices quite accurately. However, supervised models are less efficient in predicting long-term price dynamics because the hypothesis and data fed to them often reflect transient trends rather than universal laws. Also, they are often based on the manually engineered features that are biased by some implicit understanding of how financial markets work. Finally, it's

rare that supervised models are able to formulate some explicit trading policy. Being based on rigid trading rules and decision thresholds, they fail to react to changing market trends and have a narrow scope of available investment strategies. That's where Reinforcement Learning (RL) models can shine.

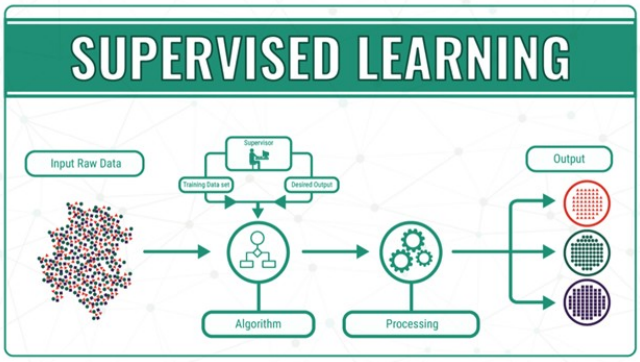
Reinforcement Learning (RL) to Develop a Sophisticated Trading Policy

The development of the AI-based agents with a flexible and automatically adjustable investment policy has recently become possible with the advances in Reinforcement Learning – a sub-field of ML widely used in robotics, autonomous driving, and board games engines (e.g Go and chess)

In a conventional RL model, we have an agent (e.g algorithm) that acts in the environment that continually emits certain rewards and punishments for the agent to refine his behavior. Based on this feedback, the agent evaluates the success of a particular action (e.g selling or buying a stock) and gradually adjusts its policy to become more efficient. At the end of the day, by experimenting with the environment and evaluating its feedback, the RL agent is able to come up with the optimal policy framework or strategy.

As it turns out, using this self-learning potential of RL agents in financial trading, we can develop more nuanced and sophisticated learned policies that enable more accurate and timely strategies and actions and price prediction. An RL agent has no hard-coded rules to follow: instead, it can formulate its own rules and policies depending on the market response and accumulated experience. For example, an RL agent can dynamically change a high-frequency trading strategy for the medium or low-frequency trading and vice versa, decide on the volume of trade, and adjust trading decision thresholds depending on the emerging market trends, historical movement of prices, and responses of other agents. In this way, an RL agent learns how to behave by experimenting with its environment (e.g stock market) continuously refining its strategies and fixing past errors.

The RL approach discussed above can be further enhanced by combining RL with powerful Deep Learning (DL) architectures that enable learning of complex non-linear hypotheses from unstructured data. The conjunction of RL and DL can produce investment policies and rules more powerful than any human trader could possibly formulate relying solely on his knowledge, intuition, and experience



.

RL for Modeling Other Trading Agents

Over the past few years, RL models such as AlphaGo and AlphaZero invented by the Deep Mind company acquired by Google Inc. outperformed all existing game engines in the games of chess and Go displaying a fantastic level of strategic thinking that goes beyond brute force and dynamic programming approaches of the past. A great success achieved by RL models in board and computer games owes to their powerful ability to act in simulated environments and model behavior of other agents (e.g other players, contenders).

This feature is directly relevant for financial trading where the ability to predict the behavior of other agents has been always regarded as a powerful advantage. However, the enormous size and the 'black box' nature of the stock market pose challenges which are hard to address using traditional game-theoretic approaches. However, as it turns out, the ability to learn through trial-and-error and generate one's own experience make RL agents powerful in understanding possible motivations of other stock traders. In particular, RL models can identify various market signals that emerge from their trading decisions and understand whether they are attributable to the market response of other agents. Such usage of RL is a promising field of research that is still in its infancy though. However, the progress is already on the horizon.

Deep Learning for Financial Technical Analysis

Deep Learning (DL) is a new sub-field of ML that has fueled recent advances in video and image recognition, machine translation, and Natural Language Processing (NLP). The main difference between DL and supervised learning is in the use of *feature learning* or representation learning instead of manually engineered features. In *feature learning*, the model automatically extracts meaningful features from data and creates multiple layers of abstraction to represent this data. Such an approach is motivated by the fact that unstructured data such as images, audio, video or speech is hard to define in terms of the manually extracted features. The same is true of the financial charts data for technical analysis which can contain hidden features not known by the researcher before the formulation of the ML model. The solution is to let the intelligent machine find those features automatically and derive more efficient and non-linear models which are hard to formulate using underlying theories or assumptions.

AI for Stock Ranking

These days, there are over . The abundance of stocks, bonds, and equities and the press 630,000 companies traded publicly all over the worldence of thousands of hedge funds competing for revenue, makes deciding upon the right investment target more complicated today than ever. Human traders are struggling to make sense of the non-stop flow of news and financial data to figure out the best compositions for their portfolios. However, with state-of-the-art AI stock ranking and performance prediction, they can breathe a sigh of relief. Stock ranking AI tools are normally based on parametric models (like linear or logistic regression) powered by multi-layer neural networks like LSTMs or RNNs. As a training set, AI stock rankers can use diverse types of data such as financial time series data, technical indicators (e.g candlestick patterns), SEC filings, corporate reports, news, and social media posts. In particular, Deep Learning allows for the integrating of these diverse types of structured and unstructured data into one model. Putting these types of data together and powering them with neural networks yields powerful prediction models that can identify the best candidates for investment. The main benefit of the ML stock ranking models is their ability to take numerous economic and extra-economic factors into account and learn non-linear features of stock trends that cannot be easily recognized even by the most experienced stock traders.

Portfolio Diversification with Machine Learning

Portfolio management is one of the most important fields of financial trading that studies efficient combinations of

stocks to generate the highest returns while keeping risks to a minimum. Until very recently, the central paradigm of portfolio management was Modern Portfolio Theory (MPT) according to which there is an optimum strategy for a given portfolio that maximizes returns at a given level of risk. However, recent advances in Deep Learning and Reinforcement Learning are revolutionizing conventional approaches to portfolio management and risk management.

In particular, advances have been made in the ML-based portfolio diversification. AI stock advisors can compose a portfolio by analyzing hundreds of stock features like alpha, beta, Sharpe ratio, Maximum Drawdown (MDD) Risk Measure. Based on this analysis, the algorithm identifies stocks with different risk premiums and return potential and builds an efficient diversification strategy for individual stocks, industries, and geographies. By adding risk measures into the equation, the algorithm ensures that stable returns are secured while losses are kept to a minimum.

If the target weight of an asset increases, the RL agent would buy an additional amount of it and sell if the weight decreased. At the same time, the RL agent would take various risk metrics such as Sharpe ratio into account to ensure that the proper balance between high-risk and low-risk stocks is achieved. It will also ensure that assets are diversified by asset type and economy sector. The discussed RL model is highly adjustable and dynamic – the RL agent constantly refines its policies using the model's feedback to develop more sophisticated diversification strategies. It also keeps the memory of previous states in Portfolio Vector Memory (PVM) layer that allows the agent to take the effect of transaction costs and various extra-market factors into account.

Evaluating Market Sentiment with Natural Language Processing (NLP)

Price prediction is usually based on three types of analysis: technical analysis, fundamental analysis and the study of the market sentiment. While the methodology for technical and fundamental analysis has been substantially improved over the past four decades, the analysis of market sentiments has been traditionally regarded as the field where intuition and educated guesses played a greater role than science. As a result, until recently, we were limited in our ability to explain the dynamic of market sentiment. However, with recent advances in Natural Language Processing, we can now make a significant progress in the understanding of various subjective variables like risk-averse or risk-seeking behavior, investor confidence, and psychological factors that drive the arrival of bearish or bullish markets. In particular, we can use the sentiment analysis with RNNs and built-in memory mechanisms to evaluate the sentiment (e.g positive, negative, neutral) of various types of data like financial news channels, data trading sets, customer feedbacks, investor blogs etc. This can be achieved by character-to-character and word-to-word mappings that turn each language token into word vectors that represent the semantic proximity of words. The NLP algorithm can then use these vectors to study the sequence of text (e.g blog post) identifying 'positive' and 'negative' signals/words and comparing them with the previous signals stored in memory. During this process, the algorithm will gradually adjust 'positive'/'negative' scores calculating the probability of the entire text being 'positive' and 'negative'. If we feed thousands or even millions of finance-related texts in such an algorithm, we can derive a pretty accurate picture of the market sentiment in a given period. Thus, NLP can become a powerful tool for identifying the changing market trends and the impact of news on the trajectory of the market.

Abbreviations and Acronyms

ML:- Machine Learning

AI:- Artificial Intelligence

AGI:- Artificial General Intelligence

DL:- Deep learning

RL:- Reinforcement Learning

MDD:- Maximum Drawdown

PVM:-Portfolio Vector Memory

Impact of AI and ML on technical analysis

We have to make a distinction between traditional and quantitative technical analysis because all methods that rely on the analysis of price and volume series fall under this subject. Traditional technical analysis, i.e., chart patterns, some simple indicators, certain theories of price action, etc., was not effective to start with. Other than a few incomplete efforts of limited scope and reach, publications that touted these methods never presented their longer-term statistical expectation but offered only promises that if this or that rule is used there would be profit potential. Since profits and losses in the markets follow some statistical distribution, there were always those who attributed their luck to these methods. At the same time, a whole industry developed around these methods because there were easy to learn. Unfortunately, many thought they could profit by being better at using methods known to everyone else and the result was massive wealth transfer from these naïve traders to market makers and other well-informed professionals.

In the early 1990s, some market professionals realized that a large number of retail traders were trading using these naive methods. Some developed algos and AI expert systems to identify the formations in advance and then trade against them, causing in the process volatility that retail traders, also known as weak hands, could not cope with. In a more fundamental way, the failure of traditional technical analysis can be attributed to the disappearance of high serial correlation from the markets starting in the 1990s. It was basically the high serial correlation that offered the wrong impression that these methods worked. Nowadays, with few exceptions, markets are mean-reverting, not leaving room to simple technical analysis methods to work. However, some quantitative technical analysis methods often work well, such as mean-reversion and statistical arbitrage models, including ML algorithms that use features with economic value.

Note that this type of arbitrage is unlikely to be repeated in the case of AI and ML because of the great variety of models and the fact that most are being kept proprietary but the main problem with this new technology is not confirmation bias, as in the case of traditional technical analysis, but data-mining bias.

In my opinion, observing the market and looking at charts is becoming an obsolete process. The future of trading is about processing information, developing and validating models in real-time. The hedge fund of the future will not rely on chart analysis. Some still do this because they are at the transition boundary where old ways meet with a new era. Many traders not familiar with AI will find it hard to compete in the future and will withdraw.

0.A Equations

1)Nadaraya-Watson estimator

$$m(x)=E[Y|X=x]$$

$$=\int yfY|X$$

$$=x(y)dy$$

=∫yf(x,y)dy/fX(x).
n∑i=1(Yi−^m(Xi))2

2)Local Polynomial Regression

n∑i=1(Yi−^m(Xi))2

#m^β(x):=argminβn∑i=1(Yi−mβ(Xi))2.
#m(Xi)≈m(x)+m′(x)(Xi−x)+m″(x)2(Xi−x)2+⋯+m(p)(x)p!(Xi−x)p.
#∑i=1(Yi−p∑j=0m(j)(x)j!(Xi−x)j)2.
#∑i=1(Yi−p∑j=0βj(Xi−x)j)2.
#^βh:=argminβ∈Rp+1n∑i=1(Yi−p∑j=0βj(Xi−x)j)2Kh(x−Xi)

3)Asymptotic Properties

Y=m(X)+σ(X)ε

4)Bandwidth selection

hAMISE=[R(K)∫σ2(x)dx2μ22(K)θ22n]1/5

Authors and Affiliations

1)Utkarsh Anil Baviskar

Student , IT DEPT. , VIIT KONDHWA,

2)SAURABH MUNESHWAR

Student, IT DEPT. , VIIT KONDHWA

3)QASIM GHANIZADA

Student, IT DEPT. , VIIT KONDHWA

Coding of the program made

lstm.py

```
#importing libraries
from sklearn.preprocessing import MinMaxScaler
#from sklearn.exceptions import DataConversionWarning
#warnings.filterwarnings(action='ignore', category=DataConversionWarning)
from keras.models import Sequential
from keras.layers import Dense, LSTM
#import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def Ls():
    #reading the csv file
    headers=['Date','Open','High','Low','Last','Close','TotalTradeQuantity','Turnover(Lacs)']
    df = pd.read_csv('NSE-TATAGLOBAL11.csv')
    #print(df)

    #plotting the historical data graph
    df['Date'] = pd.to_datetime(df.Date)
    df=df.sort_values(by='Date')
    #df.ix[pd.to_datetime(df.Date).order().index]
    #print(df)
    df = pd.read_csv('NSE-TATAGLOBAL11.csv')
    df['Date'] = pd.to_datetime(df.Date)
    df=df.sort_values(by='Date')
    x = df['Date']
    y = df['Close']

    #plt.plot(x,y)
    #plt.xlabel('date',fontsize=18)
    #plt.ylabel('close',fontsize=18)
    #plt.show()

    #creating dataframe
    data = df.sort_index(ascending=True, axis=0)
    data['Date'] = pd.to_datetime(data.Date)
    data=data.sort_values(by='Date')

    new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
    for i in range(0,len(data)):
        new_data['Date'][i] = data['Date'][i]
        new_data['Close'][i] = data['Close'][i]

    new_data['Date'] = pd.to_datetime(new_data.Date)
    new_data=new_data.sort_values(by='Date')

    #setting index
    new_data.index = new_data.Date
    new_data.drop('Date', axis=1, inplace=True)

    #creating train and test sets
    dataset = new_data.values

    #plt.plot(new_data['Close'])
```

```
train = dataset[0:987,:]
valid = dataset[987:,:]


#converting dataset into x_train and y_train and for normalizing data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)



x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)


x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))


# create and fit the LSTM network designing lstm model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))


model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)


#predicting 246 values, using past 60 from the train data
inputs = new_data[len(new_data) - len(valid) - 60:].values


inputs = inputs.reshape(-1,1)



inputs  = scaler.transform(inputs)


X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)


X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)


closing_price = scaler.inverse_transform(closing_price)


closing_price=closing_price.reshape(-1,1)


#for plotting


train = new_data[:987]
valid = new_data[987:]


#slen=len(valid['Close'])
valid['closing_price']=closing_price
#valid=valid.insert(slen,'closing_price',closing_price)
final = valid['closing_price'][-1]
#valid = valid.assign(closing_price=pd.Series(np.random.randn(slen)).values)
#valid['closing_price'] = pd.Series(np.random.randn(slen), index=valid.index)


plt.plot(valid['Close'])
#plt.plot(train['Close'])
plt.plot(valid['closing_price'])
#plt.plot(valid[['Close','closing_price']])
plt.xlabel('date',fontsize=18)
plt.ylabel('close',fontsize=18)


plt.savefig('static/new_plot.png')
plt.close()
return final
```

Frontend.py

```
from flask import Flask
from flask import render_template, url_for
from lstm import Ls
global i
app = Flask(__name__)
i=1
@app.route('/')
def hello_world():
    global i
    if i ==1:
        val = Ls()
        i=2
    return render_template('index.html',sr="/static/images/new_plot.png",val=val)
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Output of the program