Introduction

Ishan Chadha

MGT 4067 Final Essay

Project Code

GitHub link: https://github.com/ishanchadha01/finance-intelligence/tree/master

Data: https://github.com/ishanchadha01/finance-intelligence/blob/master/data/input.csv

Preprocessing: https://github.com/ishanchadha01/finance-intelligence/blob/master/data/preprocessing.py

Web scraper: https://github.com/ishanchadha01/finance-intelligence/blob/master/data/webscraper.py

LSTM code: https://github.com/ishanchadha01/finance-intelligence/blob/master/models/lstm.py

Sentiment analysis with BERT code: https://github.com/ishanchadha01/finance-intelligence/blob/master/models/sentiment_analysis.py

Currency Basket Anomaly Detection with SOM/PSO: https://github.com/ishanchadha01/finance-intelligence/blob/master/models/som.py

Saved LSTM models: https://github.com/ishanchadha01/finance-intelligence/tree/master/models/saved_models/lstm

Basic Trader simulation: https://github.com/ishanchadha01/finance-intelligence/blob/master/stock_prediction/simulation.py

Plots: https://github.com/ishanchadha01/finance-intelligence/tree/master/plots

Abstract

Stock price prediction has long been a focus of both financial professionals and individuals looking to make short and long-term profits. Recently, researchers have uncovered that hybrid machine learning models perform better than models that use individual trend indicators. This paper examines the success of an LSTM-CNN with basic information such as the stock's close price as well as more complex indicators, including sentiment analysis using BERT and anomaly detection for a currency basket. The stock examined in this study was that of GE over the past 10 years. Although the results showed that in the short-term, these strategies will not work, feature boosting alongside long-term simulations using deep reinforcement learning are promising future implementations.

Keywords: LSTM-CNN, BERT, anomaly detection

Introduction

Stock forecasting has long been a focus of many investing professionals as well as amateurs looking to make profits on the market. Alongside this, general timeseries forecasting methods are useful in other fields, such as risk detection, disease modeling, or bodily rhythm monitoring in the medical field [1]. Simple stock forecasting methods work based on keeping tracking of momentum-based indicators, such as moving averages for different lengths of time. Individual stocks do not operate in a black box – they are influenced by real world events, such as the recent COVID-19 pandemic. Because of this, it is also useful to keep track of complementary factors that can yield information about external events. One of these pieces of information is currency, or more specifically, a basket of world currencies. Currencies can give us information about global events, especially their fluctuations relative to each other. Currencies comprise large datasets on their own, however, and analyzing multiple sets of currencies poses a computationally expensive challenge. To reduce dimensionality while still retaining trend information, a self-organizing map can be employed, which can also provide information about the presence of outliers as well as clusters in the data. Training a self-organizing map (SOM) is similar to training a deep learning network though, which means that backpropagation over gradients would still prove to be computationally expensive. Thus, particle swarm optimization (PSO) was utilized as the update step [2]. Particle swarm optimization is an effective nonlinear optimization technique based off the social and cognitive behaviors of animals that travel in herds.

Alongside currency fluctuation prediction, a powerful indicator of global events is the news itself. For this reason, sentiment analysis can be a useful tool for stock price prediction. The most effective natural language approach for sentiment analysis is called BERT. BERT uses

a tokenizer to encode information about positive and negative words and phrases and then an autoencoder to reduce the dimensionality to a vector space that only the model can interpret. These features are then trained bidirectionally, containing both left-to-right and right-to-left connections throughout the text to fully capture the emotions present. After this, they are decoded and returned. In conjunction, the currency basket and SOM with PSO are a strong potential candidate for predicting external fluctuations in the market.

LSTMs are promising tools which, unlike neural networks that only move in the forward direction, also include backward connections to retain memory over different timespans. LSTMs have both short term and long-term memory units, which make them useful for this application.

Data

Data for the GE stock was collected from Yahoo Finance for a 10-year period, from which seven day moving average, 21-day moving average, upper Bollinger band, lower Bollinger band, 12-day exponential moving average, 26-day moving average, and moving average convergence divergence were calculated. As an additional element, the GE options average daily volume (ADV) was passed in as an additional column.

After this, external factors were added to the data. S&P Futures for the same 10-year timespan were included, as well as EURUSD, USDJPY, and EURJPY currencies. These currency baskets were passed through the SOM with PSO, after which a line was found which gave EURUSD values in terms of USDJPY and EURJPY. The difference between predicted EURUSD and actual EURUSD was appended to the dataset as "EURUSD Fluctuations." Finally, Reuters headlines about GE from the past 3 years was scraped and given a rating of -1, 0, or 1 corresponding to negative, neutral, or positive sentiment, respectively. All dates other than the ones from the past 3 years were given neutral rating, and then the data was appended to the dataset as "Sentiments."

All raw data can be found at this link:

https://github.com/ishanchadha01/finance-intelligence/tree/main/data/raw_data

The full dataset can be downloaded here:

https://github.com/ishanchadha01/finance-intelligence/blob/master/data/input.csv

Technique

The basic indicators were calculated using the following formulas:

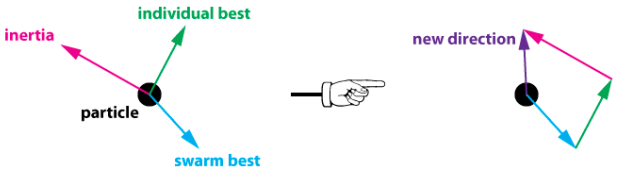
$$Moving\ Average: MA_n = \frac{\sum_{i=1}^{n} A_n}{n}$$

$$\text{Upper/Lower Bands: BOL} = MA_{21} \pm 2\sigma_{20}$$

$$Exponential\ Moving\ Average: EMA_n = \frac{2}{n+1} A_n + \frac{n-1}{n+1} EMA_{n-1}$$

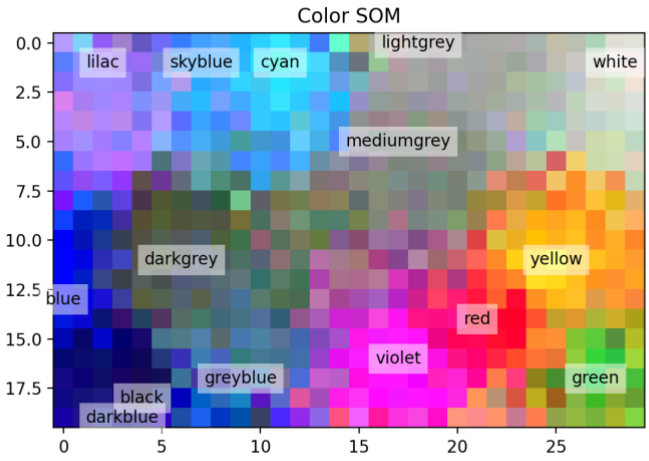
$$MACD = EMA_{12} - EMA_{26}$$

For the anomaly detection on the currency basket, 24 particles were used in a 4x6 grid. A SOM network structure was coded in PyTorch, and the update step for particle weights was replaced by particle swarm optimization. Each particle, which in this case corresponded to the value of each node the in hidden layers of the SOM, was assigned a position and a velocity which gravitated towards local centroids as well as the global optimum after each step. The social constant used was 0.1, whereas the cognitive constant was 0.01 since the technique was ineffective as clustering with too high of a weight on the cognitive step.

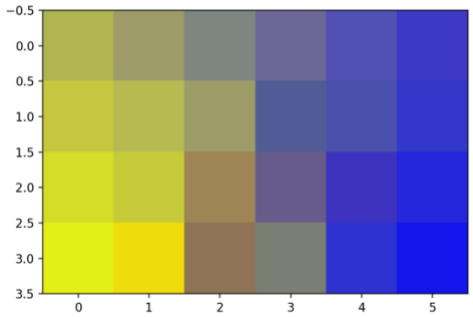


Currency inputs were passed in as tuples containing (EURUSD, USDJPY, EURJPY) for each date of the 10 years, and a 4 by 6 particle grid was used for the SOM. Since updating all the particles at every step would be inefficient as well as ineffective at forming good clusters, every time an input was passed in, the closest particle in the cluster was located, called the best

matching unit (BMU). Only points within a certain radius, called the learning radius, were updated. The initial learning radius used was 0.5, which reduced exponentially as the iterations increased. The learning radius did not correspond to the actual distance the new input was away from the BMU, but rather than distance in terms of the radial basis function (RBF), otherwise known as the Gaussian kernel function. This model was first tested on a color dataset, yielding the following promising result:

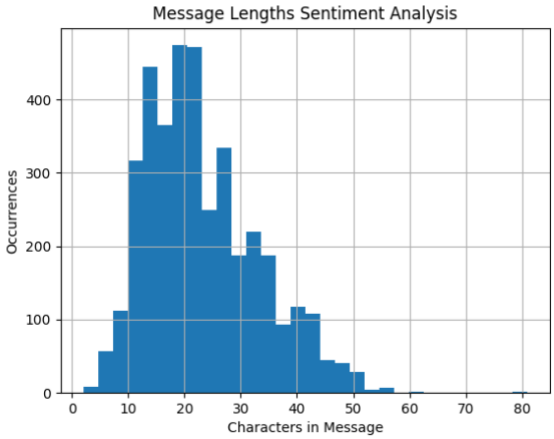


The outputs of the SOM were normalized over RGB values and plotted in a similar manner. Since many outliers existed, any particles outside of 2 standard deviations of the mean were omitted.

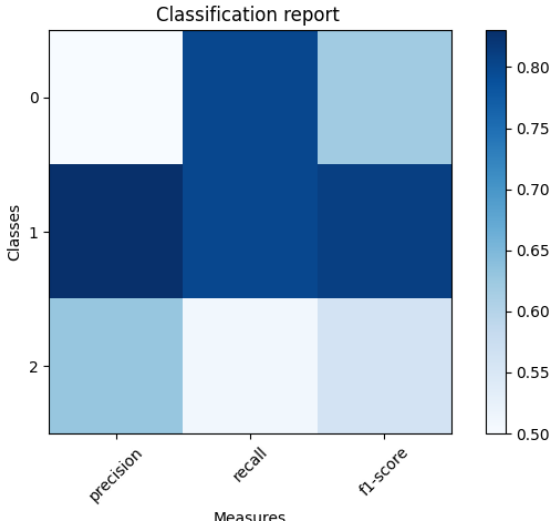


Clearly, two clusters existed for the data, which after further analysis, formed a linear relationship. This relationship was expressed as EURUSD in terms of USDJPY and EURJPY and the differences between this and the actual EURUSD value was calculated at each step and called “EURUSD fluctuations.” Many fluctuations were high, indicating that this method did not work as intended, probably focusing in on one region of the trend line between the currencies rather than the trendline as a whole.

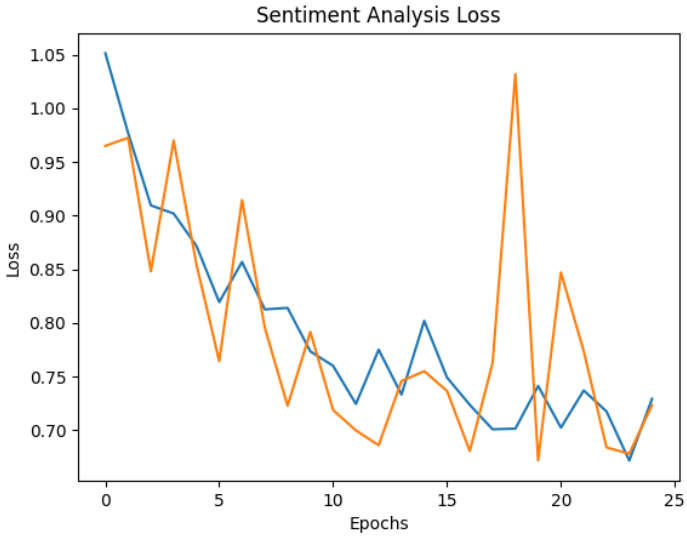
For sentiment analysis, first a BERT model had to be trained. Transfer learning was applied to this problem, where the data was trained on labeled Reddit sentiments and then tested on stock news since a good dataset did not exist for stock news sentiments. Labeled Reddit data was acquired online, whereas testing data was curated by building a “bot” which scraped a Reuters page containing headlines about GE. The BERT model had a linear layer with input size 768 and output size 512, a rectified linear unit (ReLU), a dropout component, a linear layer of input size 512 and output size three, and a softmax unit, yielding a -1, 0, or 1 corresponding to negative, neutral, or positive sentiments, respectively. Messages passed in were tokenized, or put in terms of message units that the model can understand. A max sequence length of 50 was used and the rest of the data was padded to be of this length based on the following calculated message lengths:



Overall, the precision of positive sentiments was high, and the recall of negative sentiments was very high, as the following classification report shows:

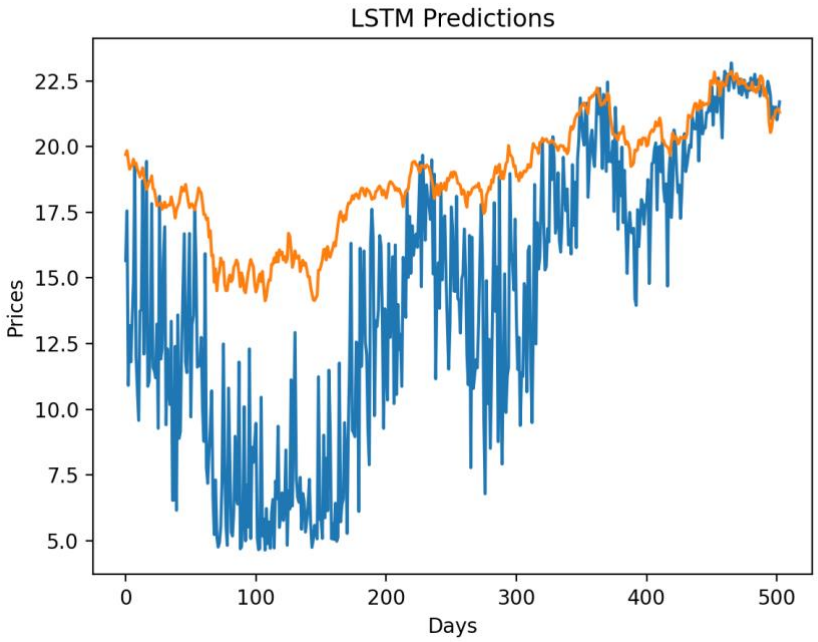


The sentiment analysis model did show great improvement over the trained iterations, despite the relatively low precision of negative sentiments:

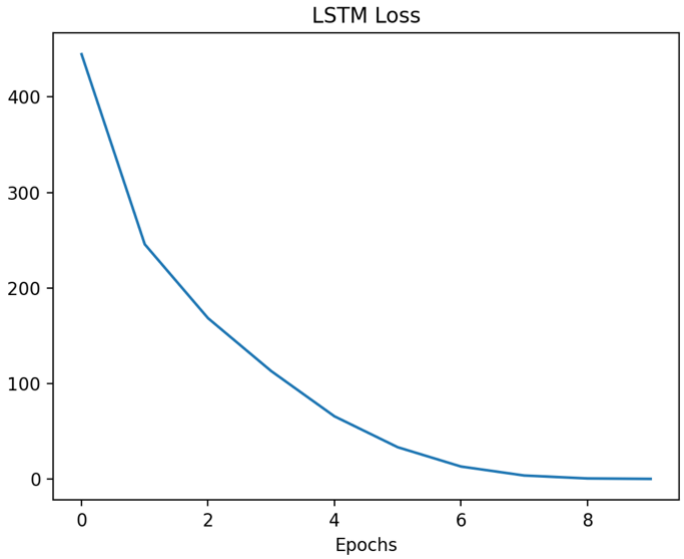


The model training itself occurred on a LSTM-CNN build using PyTorch. The first 20% of the dataset was used for training and the final 80% was used for testing, where all columns of the dataset were used for the input and the closing price of the subsequent day was the output.

Two hidden layers were used in the CNN, each with size 32, and a learning rate of 1e-5 was used. Predictions on the test set had the following results:



Here, the blue line represents predictions, and the orange line represents actual prices. The loss reduced vastly over the 90,000 epochs of training, which took approximately four hours:



Finally, to test the overall short-term model performance in a simplistic manner, a basic trading bot was programmed which bought one share if the stock was forecasted to go up the next day and sold all its shares if the stock was forecasted to go down. The trading bot ended with approximately $0.31 less than when it started, indicating that improvements must be made to this model before it can be applied to real world trading.

Conclusions

Conclusions can be drawn on the performance of each of the three models: sentiment analysis with BERT, currency basket anomaly detection using SOM with PSO, and the LSTM for training the forecasting tool.

BERT is a powerful model, but the reddit data might not have been the best match to stock price prediction for the transfer learning approach. The model took a long time to train (~6 hours) and struggled to detect negative sentiments, which are extremely important in detecting fluctuations.

The currency basket analysis provided promising efficiency, converging in about 10 minutes with PSO. However, PSO may have converged too fast, finding a local trendline rather than a global one. This speculation would explain the consistent difference between actual EURUSD prices and predicted EURUSD prices over the course of the 10-year timespan. In the future, longer training time as well as smaller parameters and more particles could be utilized to find a more holistic trendline.

LSTMs have recently been outperformed by GRUs, which are similar in structure to LSTMs and less computationally expensive. This would allow faster convergence on training time and a possible ensemble training approach which would give better results over a portfolio of stocks rather than just one. Another approach to speed up the LSTM is the use of multithreaded programming, which may be used in the future to take advantage of parallel processing on GPUs. Besides this, the LSTM did a relatively good job of forecasting predictions other than the amount of noise. This issue could be tackled by conducting some feature engineering prior to passing in the inputs, such as locating heteroscedasticity and feature importance via XGBoost.

For better testing, the simple trading bot can be upgraded to utilize value iteration or deep reinforcement learning to fully use both the long- and short-term elements of the LSTM model. A possible approach for this is the use of a DQN, or deep Q network, which has previously been used in robotics applications. This hybrid forecasting approach could also be applied to other timeseries problems, such as analyzing brainwaves for detection of early onset Alzheimer's. With further development, this hybrid model gives promising results for long term stock forecasting.

References

1. https://link.springer.com/chapter/10.1007/978-981-10-4361-1_138#:~:text=In%20medical%20applications%2C%20time%20series,assess%20the%20time%20dependent%20risk.
2. https://www.sciencedirect.com/science/article/pii/S1026309811001751
3. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0212320
4. https://arxiv.org/abs/1810.04805
5. https://arxiv.org/pdf/1812.04199.pdf