PARALLELIZING MACHINE LEARNING AND STATISTICAL ALGORITHMS FOR STOCK PRICE PREDICTION

Course: High-Performance Scientific Computing (ME 766)
Guide: Prof. Shivasubramanian Gopalakrishnan

Team members:
Chinmay Gandhareshwar (193109018)
Manthan Dhisale (193109014)
Supriyo Roy (193109013)
Abhishek Chikane (193109004)

INTRODUCTION

Stock market analysis:
Stock market analysis is divided into 2 parts:

- **Fundamental analysis:**

Fundamental analysis involves analyzing the company's future profitability on the basis of its current business environment and financial performance.

- **Technical Analysis:**

Technical analysis involves reading the data charts and using statistical figures to identify the trends in the stock market.

This project is mainly focused on technical analysis of the stock market where future stock prices are predicted by analyzing the past data.

Why machine learning, deep learning algorithms, and parallelization:

As stock price data is a time-series data, where future prices of stock depend on its past prices and also on many other factors. To cater to these dependencies, we need to perform analysis using various regression models starting from averaging to linear regression and then move on to advanced techniques like Auto ARIMA and LSTM. Since, these models involve a lot of operations like calculating errors, weights, gradient descent, etc which can be parallelized to save time.

LITERATURE REVIEW

The stock prices depend on many complex factors but the available datasets have very few features like highest stock price in a day, lowest stock prices, opening stock price, closing stock, etc. These features are not enough to predict futures stock prices accurately. So, many times statistical models fail to give accurate results. This is why researchers started working on alternative methods for stock prediction like machine learning and deep learning. Then it was found that machine learning and deep learning algorithms work far better than traditional statistical algorithms. But the only disadvantage is that these models are very time-consuming.

So, in this work, we have given an attempt to parallelize these machine learning and deep learning algorithms to save time and predict future stock prices faster. In this work, we have parallelized the following algorithms:

- **Statistical algorithms** - Moving average, Auto Regression, AutoRegression Integrated Moving Average
- **Machine learning algorithms** - Linear regression, KNN
- **Deep learning** - Artificial Neural Networks

LIBRARIES, LANGUAGE, COMPILER

- Programming language - Python
- Libraries - Numpy, Scipy, Pandas, Matplotlib, Statsmodel
- Compiler - Numba
- OpenMP threads

DATASET AND FEATURES

For our project, we have taken a time series stock market data of Tata Global. Time series data is a series of data points indexed in time order. Most commonly, it is a sequence taken at successive equally spaced points in time.

No of features used for the prediction is five. Features are date, open, highest, lowest and last value of stock. Date is the most important parameter used in prediction and our predicting label is the closed value of the stock market.

Sequential variation of closed value of stock:

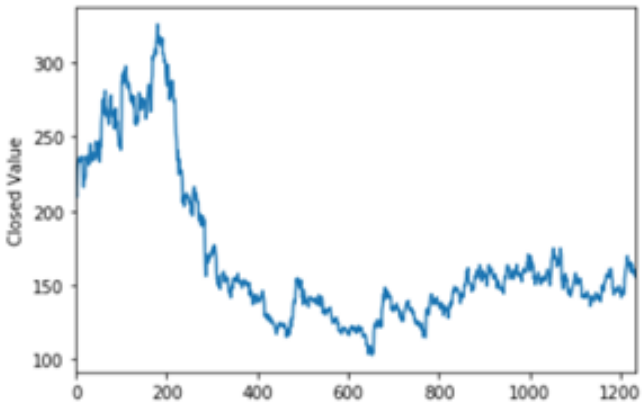


Fig.1: Closed Value of Stock

Dataset Source:
https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python

ARTIFICIAL NEURAL NETWORK

A **neural network** (also called an ANN or an artificial **neural network)** is a deep learning of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

In our project, we used a network of 2 hidden layers and 7 nodes in each layer. For the stock prediction, we optimized mean square error with L2 regularization using mini batch gradient descent. Gradient descent is parallelized using openMP.

Hyper parameter used in this model are learning rate(η): 1e-10 & regularization parameter(λ): 0.1

Time taken in serial Code: 32720.68 millisecond.Parallelized the NN algorithm using 2, 4, 6, 8 threads. The root mean square error and time distribution w.r.t no of threads is shown.

Threads	Time (millisecond)	RMSE
2	19256.1	16.33
4	19347.42	16.33
6	18189.87	16.33
8	19492.73	16.33

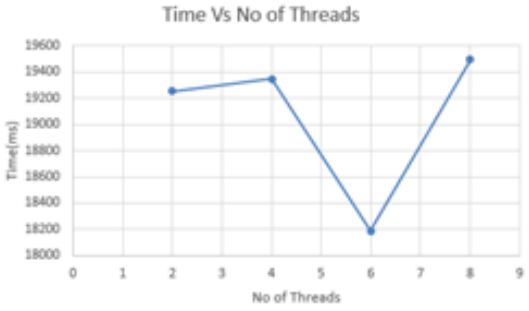
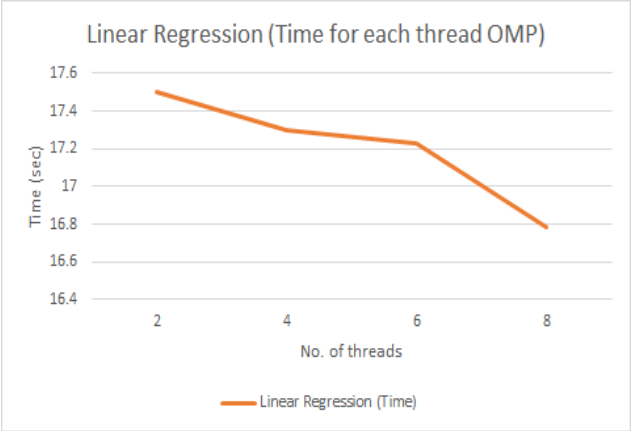


Fig.2: Time Variation w.r.t. Threads

MULTIVARIATE LINEAR REGRESSION

While exploring the ML models, one of the basic algorithms we tried to parallelize was the Multivariate Linear Regression. The code was parallelized for 4 looping processes within viz. Calculating error, applying Gradient Descent, evaluating weights, and giving the predictions. Parallelization was done using NUMBA in Python for the following threads.

Threads	Time (sec)	RMSE
2	17.499	8.718546
4	17.3	8.718546
6	17.228	8.718546
8	16.782	8.718546



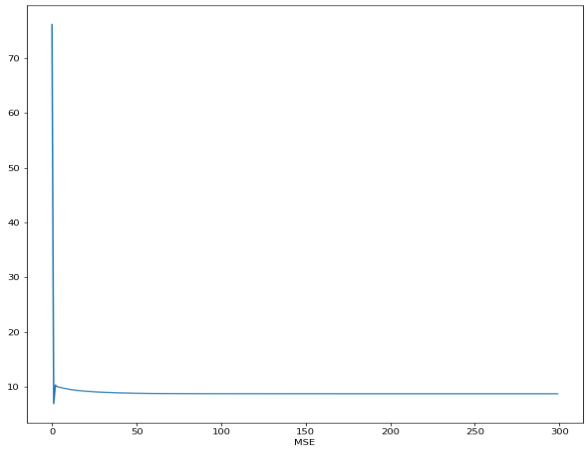


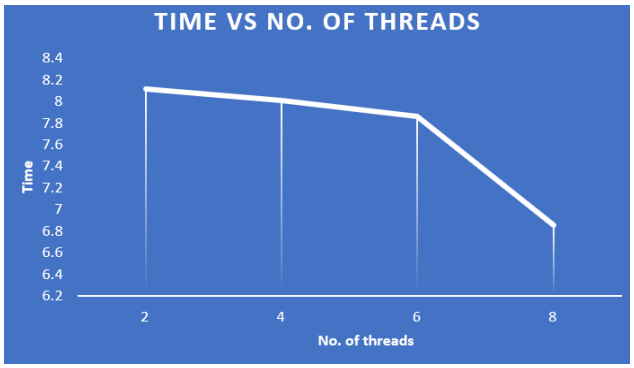
Fig.3: Time Study Plot for Linear Regression

Fig.4: RMSE variation with Iterations

K-Nearest Neighbors (KNN)

KNN is a machine learning algorithm that is used for both classification and regression. Mainly it has two parameters, distance & no. of nearest neighbors. Class of KNN is parallelized for euclidean distance & predict function. The error is least for (K=2), the time distribution w.r.t no. of threads is as shown.

Threads	Time (milliseconds)	RMSE
2	8113.9	2.118
4	8006.6	2.118
6	7863.07	2.118
8	6857.826	2.118



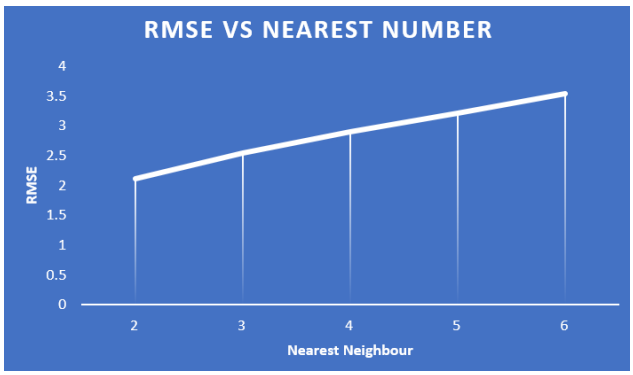


Fig.5: Time variation w.r.t no of threads

Fig.6: RMSE variation w.r.t "K"

STATISTICAL MODELS

In statistical models, we have implemented Moving Average and Auto Regression. Just to give a rough idea about implementation and parallelization, we have implemented these models from scratch. Moving average and auto regression models use past values of stocks so before calculating previous values, we cannot calculate future values. So, while implementing this model, we have kept the outer for loop in series and parallelized whatever operations happen inside the for a loop.

Moving Average:

After performing basic data preprocessing, we plotted the autocorrelation function (ACF). On the basis of that plot, we decided to use 1st ordered MA. I have discussed the final results here.

- Time taken by the serial code - 24.98 msec
- RMSE = 40.65
- The plot of time taken by parallelized code vs the number of threads:

Threads	2	4	6	8
Time (msec)	1.91	4.49	6.68	11.9

Auto Regression model:

After performing basic data preprocessing, we plotted the partial autocorrelation function (PACF). On the basis of that plot, we decided to use 1st order AR. I have discussed the final results here.

- Time taken by the serial code - 78.73 msec
- RMSE = 15.82
- The plot of time taken by parallelized code vs the number of threads:

Threads	2	4	6	8
Time (msec)	41.28	45.73	51.12	67.59

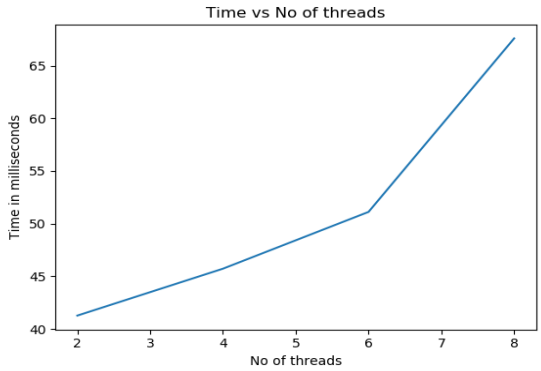


Fig.7: Time vs no of threads of MA model

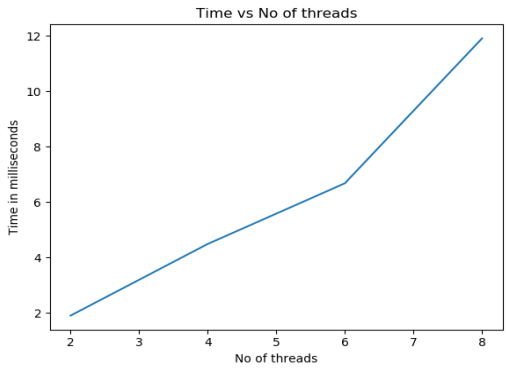


Fig.8: Time vs no of threads of AR model

From both MA and AR model plots, the time taken by parallelized code is increasing with the increasing no of threads. The main reason behind this is the small sequence length we are considering. For both models, we have considered sequence length as 30 which is very small. For large sequence lengths, 6 and 8 threads perform better but then the error was increasing so for this particular dataset we decided to keep sequence length as 30.

RESULTS AND CONCLUSIONS

Threads	Linear Regression	KNN	Moving Average	Auto Regression	Neural Network
2	17499.2954	8113.9	1.91	41.28	19256.10
4	17300.5695	8006.6	4.492	45.73	19347.42
6	17228.4250	7863.07	6.68	51.12	18189.87
8	16782.7267	6857.8265	11.904	67.59	19492.73
Series	20456.8569	11635.958	24.98	78.73	32720.68
RMSE	8.718546	2.1178	40.65	15.82	16.33

(Time in Millisecond)

From the above collective time study and stats, we see that the Moving Average method gives a better time economy as compared to others, while the Root Mean Square error (a measure of accuracy) is lowest for K-Nearest Neighbours. Thus from this experimentation, we can recommend Moving Avg. if one wants time efficiency OR KNN if one wishes better accuracy. To have the best of both worlds, one can choose the remaining models based on the application.

Reference:

1. https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python
2. https://github.com/ManthanND/PARALLELIZING-MACHINE-LEARNING-AND-STATISTICAL-ALGORITHMS-FOR-STOCK-PRICE-PREDICTION