Machine Learning Nanodegree Capstone Report

Stock Price Predictor with LSTM

Zhuang Xiu

2018.4.2

DEFINITION

Project Overview

Investment firms, hedge funds and even individuals have always had a strong need in better understanding the market behavior in order to make profitable investments and trades. Various technologies and techniques have been developed to meet this requirement including fundamental analysis, technical analysis, and the newly popularized machine-learning/deep-learning based approach. [1] A general trend is that people are relying increasingly on complex financial models. And to harvest more predictive power, the models being built are only getting more complexed and consuming more data. [2, 3]

As a matter of fact, financial industry pioneered the use of machine learning in predicting the financial market just like the adult industry did with online video.

But exactly how powerful is the neural-network based model and how does it compare to models on the traditional side of the spectrum?

This project aims to explore the making or rather prototyping of a machine-learning-based stock predictor. The approach is comparative in nature. Two models will be built using the SVR (support vector regression) model [4] – this serves as the benchmark model, and the LSTM (long-short term memory) [5] model respectively and their performance will be evaluated and compared.

Problem Statement

The major problem addressed in this project is to build a stock price predictor that takes daily trading data over a certain period of time as input, and outputs projected estimates for given query dates.

Note that the inputs will contain multiple metrics, such as opening price (Open), highest price the stock traded at (High), how many stocks were traded (Volume) and closing price adjusted for stock splits and dividends (Adjusted Close); your system only needs to predict the Adjusted Close price.

With this definition, the data source is confined to only stock price historical data. From a machine learning perspective, this is a regression problem.

Metrics

The primary evaluation metrics that will be used is MSE (mean squared error) or RMSE (root mean squared error).

The formula for this metrics is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

RMSE measures the difference between values predicted by the model and the values actually observed. It is considered to be one of the most popular metrics for evaluating regression models. [6]

ANALYSIS

Data Exploration and Exploratory Visualization

The machine learning libraries used in the project provides a few very convenient ways in exploring the dataset and having a visual presentation. Here the pandas ***Dataframe.head()*** method provides the following table slice:

Date	open	high	low	close	volume	adj close
2017-05-03	936.050000	950.200000	935.210000	948.450000	1824800	948.450000
2017-05-02	933.270020	942.989990	931.000000	937.090027	1748800	937.090027
2017-05-01	924.150024	935.820007	920.799988	932.820007	2323600	932.820007
2017-04-28	929.000000	935.900024	923.219971	924.520020	3797800	924.520020
2017-04-27	890.000000	893.380005	887.179993	891.440002	2158000	891.440002

It shows that the latest records are on the top of the table, so a reordering is needed to align the records chronologically. After the reordering, the trend is then visualized in the figure below.

Figure 1. Changes of GOOL's stock price 2007 - 2017

From the figure 1, an abnormal price change took place in 2014 where the stock price (before adjustment) was cut in half. Digging into this, it was recorded that Google split its stock in half in 2014 which explained the drop. However, this could also pose a potential challenge to our model because of this discontinuity.

Algorithms and Techniques

The core machine learning algorithms used here are support vector regression(SVR) and long-short term memory in recurrent neural networks(LSTM). This article [7] has a fairly detailed explanation on the problem that LSTM solves and this page [8] contains a series of slides that best explain how SVR works in the context of support vector machines.

Benchmark Model

The support vector regression is selected as the benchmark model due to its versatility and reported predictive power [9]. With the kernel approach, SVR has an advantage in handling non-linear regressions.

METHODOLOGY

Data Preprocessing

The primary task in this step is data normalization.

The standard scaler (z-score standardization or standardization for short) was selected for preprocessing task for the SVR model and the min-max scaler (min-max scaling or normalization

for short) for the LSTM model. Researches have been done to compare the applicability of these two methods. [10] The mathematical difference of the two is that standardization rescales the data to center on 0 and aligns the deviation of all features to be 1 whereas normalization rescales all features into a range of [0, 1] using the following formula:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

From my understanding, in the SVR case, standardization would suffice the need to put volume at the same scale as other features plus the values would still be large enough for the calculation of cost function (too small and the cost function is automatically below the error threshold. Granted this problem can be addressed by lowering the error threshold, it still requires finding an appropriate one). From my experimentation, the default error threshold works well with standardized data.

From a survey of different LSTM implementations, it appears that normalization is a common practice. As the primary objectives of normalization is to bring the data close to zero, which makes the optimization problem more numerically stable.

Here is a review of the results after feature scaling for both methods and the difference can be best illustrated by the visualizations.

1. Results after standardization

Date	open	high	low	close	volume	adj close
2009-05-22	-2.847057	-2.862058	-2.840449	-1.669832	-0.230709	-1.365145
2009-05-26	-1.676333	-1.623121	-1.661317	-1.603971	0.744205	-1.337579
2009-05-27	-1.593471	-1.581758	-1.571210	-1.596694	0.694843	-1.334532
2009-05-28	-1.575071	-1.583205	-1.572427	-1.567342	0.437718	-1.322247
2009-05-29	-1.554310	-1.549379	-1.535375	-1.525921	0.423247	-1.304910

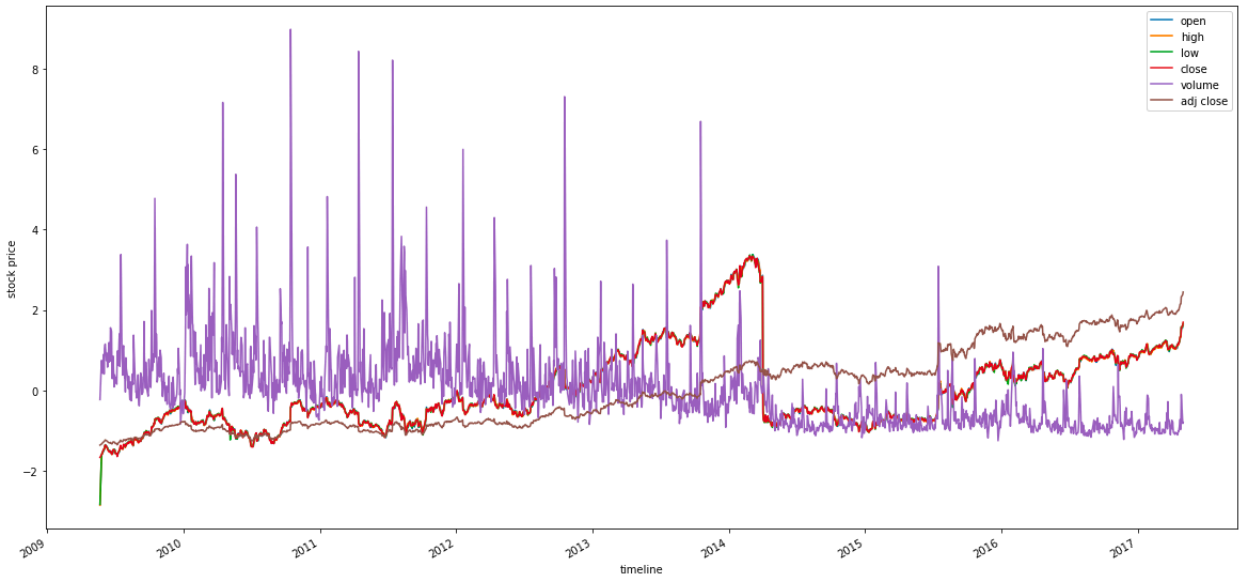

Figure 2. Stock price after standardization

2. Results after normalization

Date	open	high	low	close	volume	adj close
2009-05-22	0.000000	0.000000	0.000000	0.000000	0.100109	0.000000
2009-05-26	0.188104	0.199616	0.189557	0.013137	0.195266	0.007233
2009-05-27	0.201417	0.206280	0.204042	0.014589	0.190448	0.008032
2009-05-28	0.204374	0.206047	0.203847	0.020443	0.165351	0.011255
2009-05-29	0.207709	0.211497	0.209803	0.028706	0.163939	0.015804

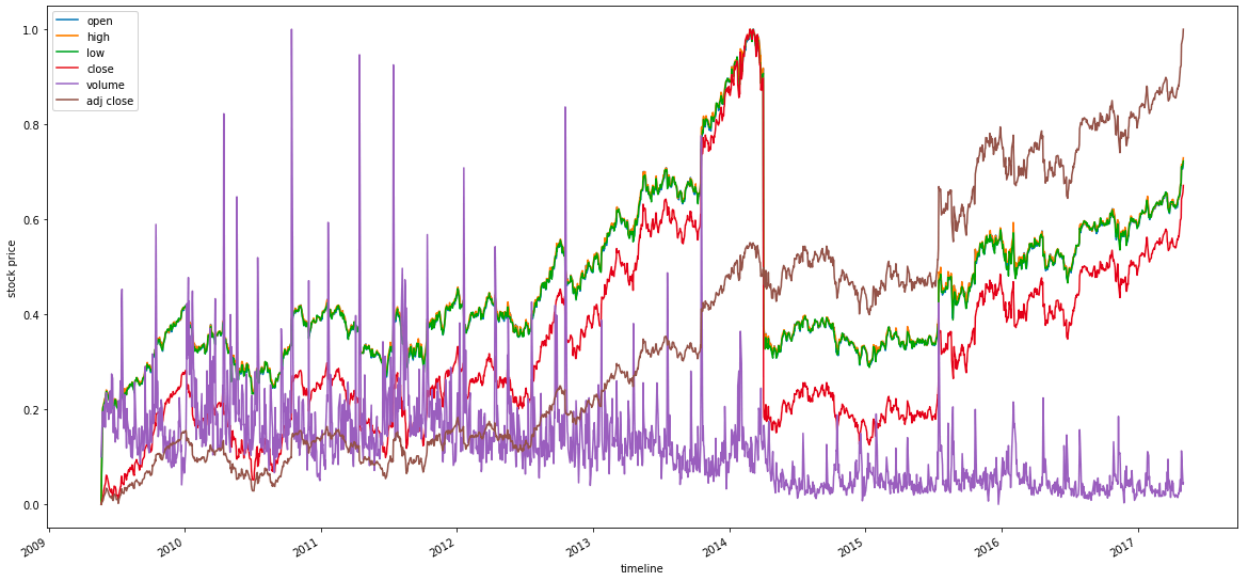

Figure 3. Stock price after normalization

One caveat is that now that the data is rescaled in two ways we need to de-normalize the predictions before calculating our performance matric – RMSE for it to be comparable.

Implementation

SVR

Using the 'rbf' kernel with C=1, gamma=0.1, (these are default settings), and cache_size=200.

To better train the SVR model, a few technical indicators were selected and calculated to feed into the SVR model instead of using the raw data. Since this is specific to the SVR model, I feel that it belongs to this part of the discussion. The chosen indicators are as follows:

technical indicator	description	formula
momentum	general trend over a certain period	price(t) - price(t-n)
rolling average	mean of prices during a certain period	sum(price(t-n),...,price(n))/n
rolling standard deviation	std of prices during a certain period	std(price(t-n),...,price(t))
average true range	volitility of the market	ATR(t)=((n-1) * ATR(t-1)+Tr(t))/n
triple exponential moving average	smooth the insignificant movements	TR(t)/TR(t-1) where TR(t)=EMA(EMA(EMA(Price(t)))) over n days period

LSTM

The initial hyperparameter settings were inspired by a post on kaggle [11]. Customized the "batch_size" and "window_size" to accommondate the data and optimize.

Refinement

SVR

Manually experimented with different window sizes (for calculating the technical indicators), and the best performance come out when the window size is 8~9.

LSTM

Manually experimented with different window sizes (this also affects the actual structure of the neural network), and the window size of 1 favors the performance the most. This will be further discussed in the reflection part.

RESULT

Model Evaluation and Validation

Here is the performance metrics table of two models:

	SVR	LSTM
Training Score	9.207596297502324	33.59647918777672
Testing Score	7.711845916677005	9.004305540494096

Visualizations of the predictions made by the two models:

Figure 4. Predictions by the SVR model

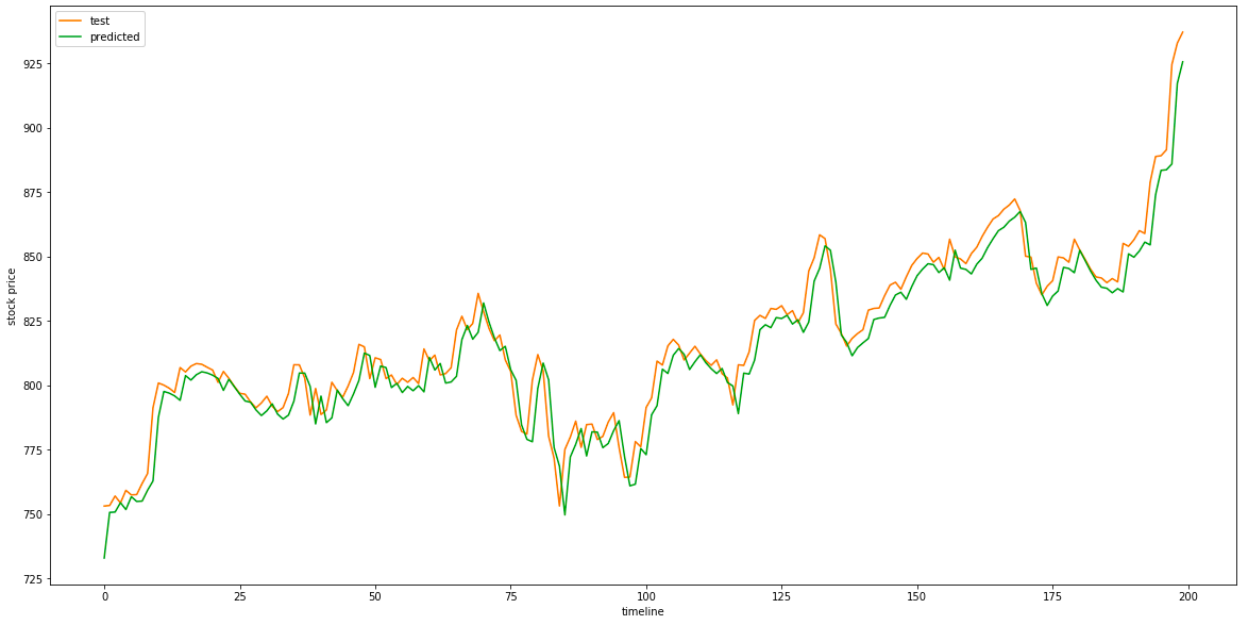

Figure 4. Predictions by the LSTM model

Justification

From a scientific perspective, no readily available method is capable of exhaustively survey all possible variations of these two models thus get the extreme performance measurements. Therefore, a definitive conclusion cannot be made that LSTM model is superior than the SVR model in time series forecasting. Nonetheless, a few meaningful observations can still be made through the implementation and evaluation process. We will attempt to compare them within the following dimensions.

1. **Innate Structure**
   Compared to LSTM or RNN's network-based structure, SVR is limited by its simpler mathematical model and its lack of expandability (kernel aside, the structure is not able to evolve to be more complex). Hence, to improve the SVR's performance, manual processing of the data is required to extract and condense the information in the data.
2. **Ease of Understanding and Optimizing**
   This is related to 1. Because of its simple structure and mathematical model, SVR is easier to understand and has a very limited set of parameters to tune. The roles of different parameters are also more comprehensible thus the results for altering any of them are more predictable. Comparatively, the process of hyperparameter tuning of LSTM falls more towards the trial-and-error side.
3. **The Overfitting and Underfitting Dilemma**
   As shown in the metrics table, the SVR model suffered from overfitting whereas the LSTM model does not. Deep learning in general is better at avoiding the overfitting-underfitting dilemma/tradeoff i.e. to improve the training score and testing score at the same time.
4. **Performance**
   For this example, this LSTM instance outperforms the SVR instance which to some extend demonstrates the power of deep learning over traditional ML algorithms.

CONCLUSION

Free-Form Visualization

(included in the result section)

Reflection

1. In the support vector regression model, we could further add technical indicators that depicts the general trend of the market, the volatility of the market etc.

2. When the window size is increased in constructing the training data, both the SVR model and the LSTM model show a trend where the predictions become and smoother i.e. the ups and downs on individual dates are smoothed out. This argument is only accurate when we're predicting the next-day stock price, if the prediction is further away in the future, having a longer window size might render a better performance.
3. There are a lot of literatures available on how to implement machine learning algorithms. It's important to be critical about if the analysis is in accord with the current paradigm of the data science and pay close attention to the procedures that the author took.

Improvement

1. Data Pipelining – a further endeavor would be to break the script into different functional modules and build a data pipeline.
2. Expand this script into an application. Use real time data for short-term forecasting and include more stocks.
3. SVR – to improve the SVR model, a general idea is to introduce new technical indicators that incorporate ample information on the market e.g. the market trend, the market volatility, tech sector trends etc.
4. LSTM – experiment with different structures and different hyper-parameters.
5. Another idea worth trying out is applying reinforcement learning to stock forecasting (inspired by Udacity's machine learning for trading course) Although to be able to simplify the state space the forecasting could be limited to only determining if the stock is going to go up or down.

REFERENCE

[1] *https://en.wikipedia.org/wiki/Stock_market_prediction*

[2] https://www.technologyreview.com/s/419341/ai-that-picks-stocks-better-than-the-pros/

[3] https://web.stanford.edu/class/cs221/2017/restricted/p-final/davdnich/final.pdf

[4] https://en.wikipedia.org/wiki/Support_vector_machine

[5] https://en.wikipedia.org/wiki/Long_short-term_memory

[6] https://en.wikipedia.org/wiki/Root-mean-square_deviation

[7] http://colah.github.io/posts/2015-08-Understanding-LSTMs

[8] https://stats.stackexchange.com/questions/13194/support-vector-machines-and-regression

[9] Di, Xinjie. *Stock Trend Prediction with Technical Indicators using SVM*

[10] http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-standardization

[11] https://www.kaggle.com/benjibb/lstm-stock-prediction-20170507/notebook