Capstone Project

Jianhuan Wang

Data Science Nanodegree

December 13, 2018

BUILD A STOCK PRICE INDICATOR

Definition

Project Overview

This project will train a model that can establish stock price indicators based on stocks and funds historical trends. Models can assist stock market traders to help them make decisions about stocks and funds trading, and more.

This project involves the LSTM model. LSTM (Long-Term and Short-Term Memory) is a long-term and short-term memory network. It is a time recurrent neural network suitable for processing and predicting important events with relatively long intervals and delays in time series.

The data set selected by this project comes from JQData, you can apply for API usage at this web site[1]. The API is generated by the company's professional field design and data collection based on the Shanghai Stock Exchange, Shenzhen Stock Exchange, third-party data providers and other publicly available raw data on the market.

Problem Statement

The data set selected by the project is the stocks and funds trend data collected from the world. The trend of stocks and funds is complicated. The project expects to train a model and use it to build a stock price indicator with a 7-day average error rate of less than 5%.

The project is divided into two steps. One is to download a stock data set and try to establish an LSTM model with different hyperparameters. And record the training hyperparameters of the best performing models. The second is to use the above training hyperparameters to establish a set of data acquisition, data processing, and modeling in a pipeline. Finally, make a script for the customer to run this pipeline

[1] **Apply for using JQData**: https://www.joinquant.com/default/index/sdk?f=home&m=banner#jq-sdk-apply

Metrics

This project uses a regression model to build a stock price indicator. The following are commonly used for regression model indicators:

1. *Mean Square Error (MSE)*
2. *Mean Absolute Error (MAE)*
3. *Huber Loss*
4. *Log cosh Loss*
5. *Quantile*
6. *Root Mean Squared Logarithmic Error*

Mean Absolute Error is a commonly used regression loss function. MAE is the sum of absolute distances between our target variable and predicted values. Its formula is shown in Figure 1-1.

$$\mathrm{MSE} = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$

Figure 1-1

However, there is a problem with MSE. For samples with a difference between the predicted value and the actual value of 1%, the larger the value of the label of the sample, the larger the corresponding loss value. In this project, I hope that the loss value they bring is the same. So, I modified it to the following loss function.

$$Loss = \frac{\sum_{i=1}^{n} \left|y_i - y_i^p\right| / y_i}{n}$$

Figure 1-2

Analysis

Data Exploration

The data set selected by this project comes from JQData[2]. This website provides a query of the price history of each stock and fund, including daily open, close, high, low and volume data. This project chooses to use the historical trend data of the stock whose code is **000300.XSHG** as the data set for training. The data set contains a total of 3199 working days of historical data.

The data set contains 5 features, respectively, opening price (open), highest and lowest price the stock traded at (high and low), how many stocks were traded (volume) and closing price adjusted for stock splits and dividends (close). The following Kline chart shows the daily price trend of the **000300.XSHG**.

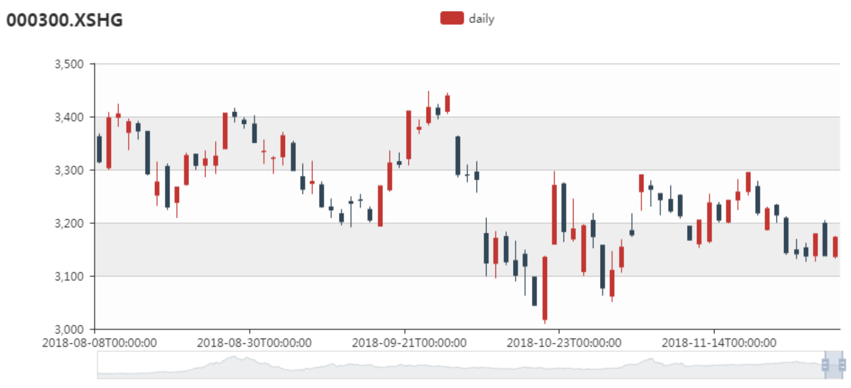


Figure 2-1

Methodology

Data Preprocessing

Filling missing values

JQdata's API may return missing data for a certain period of time, because during this time the stock exchange suspends the stock to trade on the stock market. The reason for this may be that the stock is due to a certain message or a certain Activities caused a continuous rise or fall in stock prices. After the situation is clarified or the company returns to normal, the stock will be listed on the exchange. I want to fill in the missing

[2] **JQData:** https://www.joinquant.com/default/index/sdk?f=home&m=banner

values. If there is price or volume data before, fill it with the previous value. If there is no price or volume data before, it will be filled according to the latter value.

Logarithmic processing

Because the stock price is roughly logarithmically distributed, so it is better to use a logarithmic processing on stock historical data. After logarithmic processing, the distance between the y-axis coordinate scales is proportional to the price increase rate. However, the corresponding evaluation metric also make changes. Modified as follows:

$$Loss = \frac{\sum_{i=1}^{n} \left| e^{y_i} - e^{y_i^p} \right| / e^{y_i}}{n}$$

Figure 2-2

Generating input and output data

This project first uses 30 working days of data as a training window, and trains an LSTM model to predict the closing price of the 5th working day after the end of the training window. First, I produced the input tensor. I converted the 2D tensor into a 3D tensor. The 2D tensor is the original data set, and the dimensions are samples and features. The 3D tensor is the processed input tensor, the dimensions are samples, timesteps and features. For each of the 3D tensors, it contains an entire timesteps of data, which is all data for 30 consecutive working days, the parameter ‘input_dim’ is the number of features in the dataset. Then, the label corresponding to each sample (the closing price after five working days of the last day in timesteps) is combined into a one-dimensional tensor.

Splitting data

Finally, the sample and its label are then divided into training data, testing data and validating data at a scale of 0.6, 0.2 and 0.2. Then I disrupted their order.

Implementation

Algorithms

Time series

A time series[3] is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus, it is a sequence of discrete-time data.

This time-meaning sequence is also called dynamic data. Dynamic data is very common in the natural, economic and social fields. For example, the number of animal and plant populations is increasing month by month or year by year under certain ecological conditions, the daily closing index of a stock exchange, the gross national product per month, the number of unemployed or the price index, and so on.

LSTM

Long short-term memory[4] (LSTM) units are units of a recurrent neural network (RNN). An RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

As of 2016, major technology companies including Google, Apple, and Microsoft were using LSTM as fundamental components in new products. Further in 2017 Microsoft reported reaching 95.1% recognition accuracy on the Switchboard corpus, incorporating a vocabulary of 165,000 words. The approach used "dialog session-based long-short-term memory"[5]

---

[3] **Time series**: https://en.wikipedia.org/wiki/Time_series

[4] **LSTM:** https://en.wikipedia.org/wiki/Long_short-term_memory

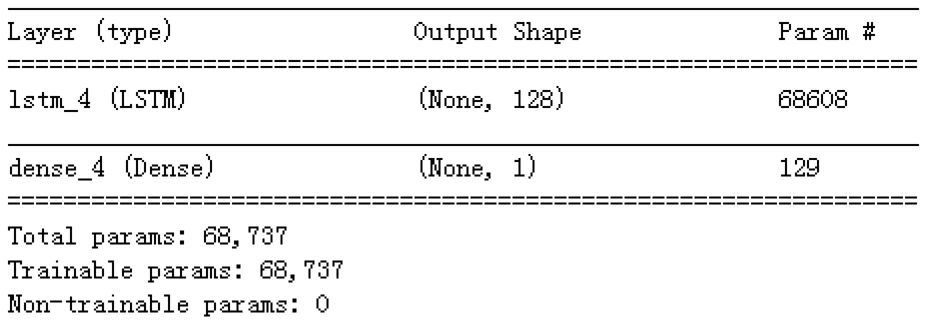
[5] **Haridy, Rich (August 21, 2017).** "Microsoft's speech recognition system is now as good as a human". newatlas.com. Retrieved 2017-08-27.

Techniques

This project uses a deep learning framework Keras (with tensorflow as a dependency library), and Keras is based on A highly modular neural network library for the Python language that supports GPUs and CPUs. Its development purpose is to make neural network experiments faster. Suitable for early network prototyping, supporting convolutional networks and repetitive networks.

Training model

First, I built an LSTM model, the first layer of the network the LSTM layer including 64 units, and then connected it with an output layer which contains 1 neuron to make a regression output.



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 128)               68608
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129
=================================================================
Total params: 68,737
Trainable params: 68,737
Non-trainable params: 0
```

Figure 3-1: Model structure

The training process uses an Adam optimizer with a learning rate of 1. It combines the advantages of the two optimization algorithms AdaGrad and RMSProp, it considers comprehensively about the first moment estimation of the gradient (the mean of the gradient) and the second moment estimation of the gradient (the unconcentrated variance of the gradient) and calculates the update step size. It has the following features, the parameter update is not affected by the gradient transformation, the hyperparameter is very explanatory because it usually does not need any adjustment.

In addition, I chose to add callback function, EarlyStopping and ModelCheckpoint to prevent overfitting. Setting the callback function allows the model to calculate the selected metric at the end of each epoch. The effect of EarlyStopping is that the model terminates training when the metric continues to decrease or rise for n consecutive epochs, n is the value of the parameter patience. This project uses EarlyStopping to observe the loss value of the validation set. The learning will be terminated if the loss value of the validation set is no longer falling within 10 epochs. ModelCheckpoint will save the structure and parameters of the model when the model gets better results during training.

The initial training trains 256 samples at each epoch. The loss of the training data after 83 epochs was 0.0439, the validating data’ s loss was 0.0388 and the error rate of the testing data was 0.0415.

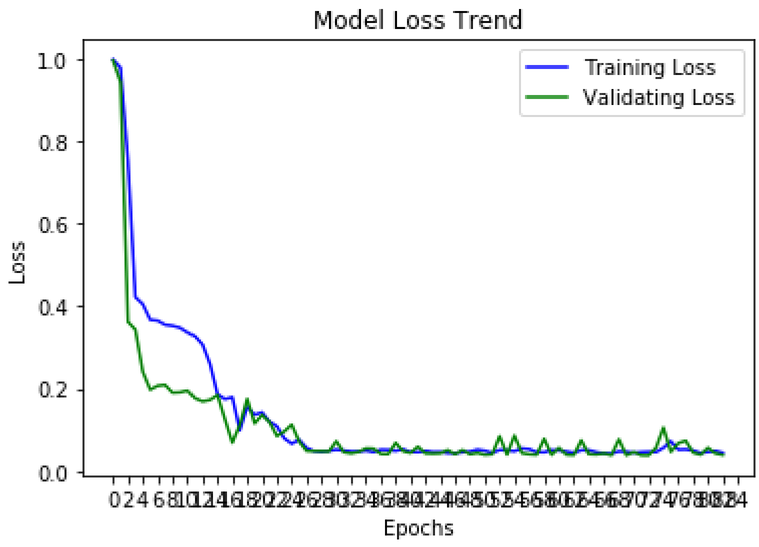


Figure 3-2: The loss trend of first training

Refinement

The first training has achieved good results, but the performance of the algorithm can be further improved by adjusting the hyperparameters. The hyperparameters that can be adjusted in the model have the "batch_size" of the optimizer, the number of LSTM layer units.

The number of units in the LSTM layer is a very important parameter. In general, a single-layer network large enough is close to any neural network. The number of units in the LSTM layer has a great influence on the training effect of the neural network. If the number of units is small, it will easily cause the model to be under-fitting, and vice versa.

In addition, the way to solve the over-fitting is to set the Dropout. Setting the dropout will make the model randomly (temporarily) delete a certain proportion of hidden neurons in the network, and the input and output neurons will remain unchanged. The hidden neurons do not participate in the forward propagation process, so there is no update in the weight of the neurons during backpropagation. The process of dropout is equivalent to averaging a number of different neural networks to achieve the effect of preventing overfitting.

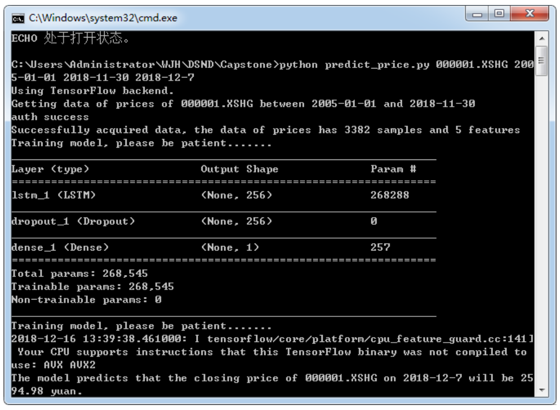
For the above hyperparameters, I set 3 different values for each hyperparameter, and there are 27 combinations. I use the 27 combinations to train models respectively. Finally, record the combination which has the best performing.

The optimal hyperparameter combination is adjusted to be:

- batch_size: 128,
- number of neurons: 256
- dropout is 0.0

After the final training is completed, the loss of the validating data is 0.03616.

Finally, I create a script for the user. The user can obtain the prediction of the close price of the stock on the day by inputting the stock code, the start date of training data set, the end date of training data set and the expected predicted date.



```
C:\Windows\system32\cmd.exe
ECHO 处于打开状态。

C:\Users\Administrator\WJH\DSND\Capstone>python predict_price.py 000001.XSHG 200
5-01-01 2018-11-30 2018-12-7
Using TensorFlow backend.
Getting data of prices of 000001.XSHG between 2005-01-01 and 2018-11-30
auth success
Successfully acquired data, the data of prices has 3382 samples and 5 features
Training model, please be patient.......
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 256)               268288
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 268,545
Trainable params: 268,545
Non-trainable params: 0
_________________________________________________________________
Training model, please be patient.......
2018-12-16 13:39:38.461000: I tensorflow/core/platform/cpu_feature_guard.cc:141]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX AVX2
The model predicts that the closing price of 000001.XSHG on 2018-12-7 will be 25
94.98 yuan.
```

Figure 3-3: Script execution

Model Evaluation and Validation

The error rate of the validating data of the final model is only 3.616%. The model uses grid search to optimize batch_size, number of units, and dropout. In order to verify the reliability of the model, I use the daily price and volume data of 100 stocks until November 30, 2018, use them to train the corresponding model by the final hyperparameters, predict the close price of December 7, 2018, and calculate the error between the predicted price and the real price. The figure below shows that most stocks have errors of less than 5% and the average error is 3.56%. The model is applied to other stocks or funds and their error between true value and predicted value is also low. I think the model has good stability and its predicted results have reference value.

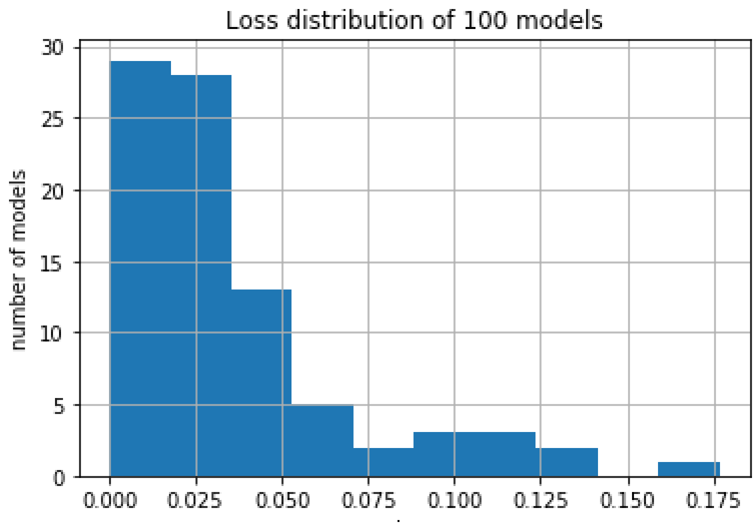


Figure 3-3 The histogram of the error rate of 100 models

Justification

In order for the machine to predict the future price of the stocks or funds, I used a time series model, first using the LSTM layer including 64 units as the first layer of the network to receive input tensor, and then connecting the output layer which contains 1 neuron to make a regression output. Then I adjust the hyperparameter to make the model achieve the best effect and record the hyperparameter. Finally, I create a script that customers can get the close price of the day they expect by inputting the stock code, the start date and the end date of the training data and the date they expect. The script can provide support for stock or fund traders.

Conclusion

Reflection

In order to build a price indicator. First, I built an LSTM model, the first layer of the network the LSTM layer including 64 units, and then connected it with an output layer which contains 1 neuron to make a regression output.

During the training, I tried to use different hyperparameter combinations to build the model, and used the grid search method to find the optimal hyperparameters. Finally, I used the optimal hyperparameter to create 100 models for 100 different stocks. The 7-day close price error rate of the model was 3.56%, which was better than my expected result. Finally, I encapsulate the process of data acquisition, data processing, modeling and predicting into a script so that users can easily obtain the predicted price by simply running the script.

Probably because of hardware problems, this script will run on my machine for a few minutes, which may make the user experience very poor. If this feature is to be applied to the website, then the run time of the script must be shortened. I thought Two solutions, one is to reduce the parameters of the model in order to reduce the training time of the model. The second is to let the machine run the script to build a model of all stocks or funds in the early morning. Then the user wants to get the predicted price, and the script they run can load the saved model, this way can omit the process of modeling.

Improvement

This project uses the LSTM model designed for time series problems. The final model predicted stock value 7 days out is within +/- 5% of actual value, on average.

In the subsequent experiments, I can try more hidden layers and neurons. The more hidden layers and neurons the model has, the more powerful the model can fit data, and the model can process more complex data. However, with the more hidden layers and neurons, the model is prone to overfitting. Therefore, the selection of the appropriate number of neurons and the appropriate number of layers have become the key to improving the generalization ability of the model. This project uses only one layer of LSTM units. In the subsequent experiments, I will try to use more layers of models to improve the approximation of the prediction of the model.

This project only uses open price, close price, high price, low price and volume as features. In addition, there are many other indicators on stocks and funds that can be selected. In the subsequent experiments, I will try to use more combinations of indicators as features to improve the approximation of the prediction of the model.