APC524 Final project report
Time series analysis in python

Wenyan Gong, Zongxi Li, Cong Ma
Qingcan Wang, Zhuoran Yang, Hao Zhang
Princeton University

January 13, 2017

1 Project objective

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. It's widely used in statistics, signal processing, pattern recognition, mathematical finance, weather forecasting, earthquake prediction, control engineering, and largely in any domain of applied science and engineering which involves temporal measurements.

In this project, we will play a game with time series in finance. It has gained its popularity in Wall Street recently, since it is fundamental to the most promising quantitative investment strategies. We develop a system that can predict future prices of stocks, with different time series models.

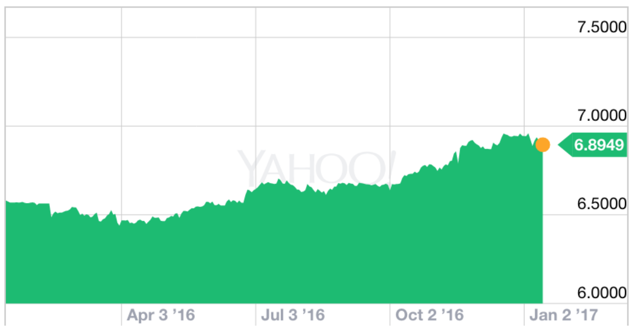


Figure 1: USD to CNY exchange rate. Jan 12 2016 - Jan 12 2017. Source, Yahoo finance.

Given input of a stock price series, our system will first fit some powerful and popular time series models, such as autoregressive (AR) model, moving average (MA) model and other derived models. This procedure will give you the estimation of parameters in these models.

The most important task in estimation is the optimization procedure. Users can select one of optimization methods in the system based on their preference. The optimization method includes but not limited to gradient descent and Newton's method.

After model fitting and estimation, our system provides a fast way to do statistical inference. Users can do different kinds of statistical test as well as confidence interval. Moreover, with fitted model, future price prediction is made and it's compared with real price data. Moreover, we provide different methods to assess the prediction accuracy, which will be visualized afterwards. With the identified model, we can further consider trading strategy and option pricing.

More interestingly, if users input multiple stock prices, we can divide these stocks into different groups, which is called clustering. Among each group, stocks share similarity. Different clusters will also be visualized. With collection of stocks (e.g., S&P 500), we can exploit the correlations within them and build a reduced order model for price prediction. For example, principal component analysis can be used to extract dominant features in the finanical market.

2 Design process

We decide to use python, since it's free, and widely distributed. Moreover, there are many powerful packages, e.g., numpy, scipy, etc.

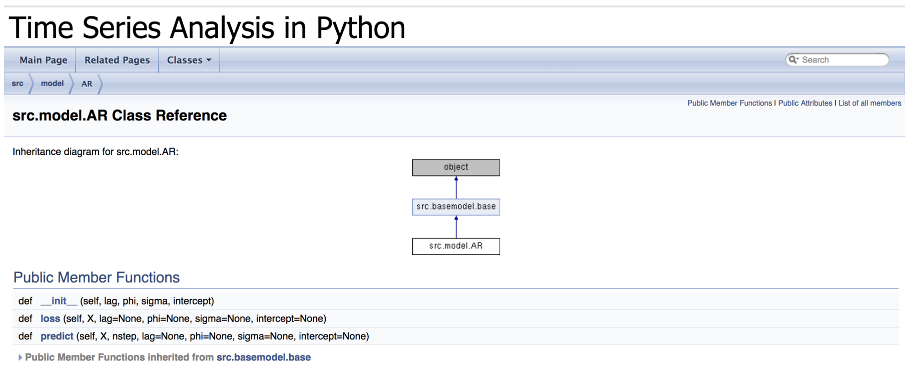
We make use of existing packages like numpy and matplot, but largely other parts are made by our group. For example, optimization method like gradient descent are implemented by us.

Github is used to track the progress of this project, and streamlines the colleboration within our group.

Along the way, we have adopted the philosiphy of object oriented programming (OOP) by carefully designing classes which include relevant functions.

Part of our code are tested using Python unittest, since there are analytical solutions. For more complicated functions, they are tested using canonical examples, and by comparing with the outputs of standard machine learning package scikit-learn.

Doxygen is used to document this project. Add explainations for doxygen.


Time Series Analysis in Python

Main Page | Related Pages | Classes | Search

src > model > AR

Public Member Functions | Public Attributes | List of all members

src.model.AR Class Reference

Inheritance diagram for src.model.AR:

Public Member Functions

```
def __init__ (self, lag, phi, sigma, intercept)
def loss (self, X, lag=None, phi=None, sigma=None, intercept=None)
def predict (self, X, nstep, lag=None, phi=None, sigma=None, intercept=None)
```

▸ Public Member Functions inherited from src.basemodel.base

Figure 2: Doxygen documentation

We have a few milestones for this project. (1) Prototype, Dec 15. In this release, we had at least one implementation for each step. Then given a time series data, we could apply at least one method from each class to do the analysis. (2) Alpha version, Jan 1. In this release, we completed most methods in each class. (3) Final version, Jan 12. In this relase, we finalized the project. Finish all the testing using various data including simulated data and real finanical data.

3 Architecture

The high-level program structure is shown below. The division of work is pretty even, and there are some minor work that are too trivial to mention extensively here. Overall, the program consists of two parts. The first part deal with a single time series, and exploits the time correlation within the time series. There is data preprocessing module that takes the raw data and get the right dataformat for later analysis. Optimization, model, and solver are used all together to identify the models. Finally, there is a post processing module that makes use of the information from model. The post processing module includes parameter inference, trading strategy, and option pricing. The second part deals with a collection of time series of data, basically it exploits the correlation between different time series. In this way, we can gain more insights into the finanical market. While these insights are impossible for a single time series.

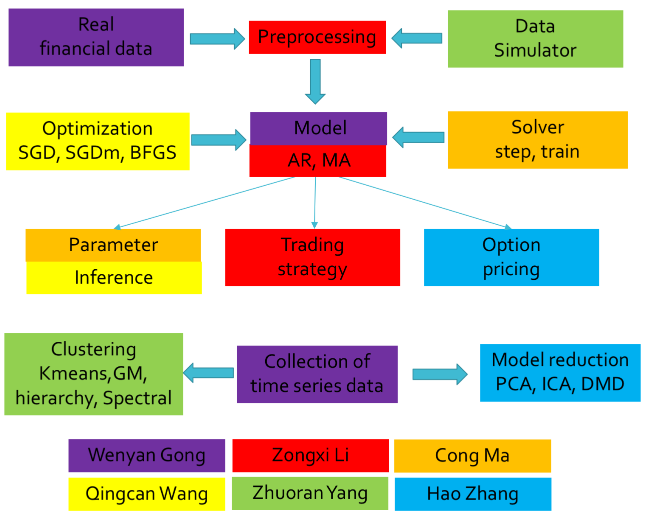


Figure 3: Program structure and division of work

3.1 Data collection & preprocessing

We develop simulator to simulate data for model testing. At the same time, real finanical data (S&P 500) are also collected from online source.

3.2 Model

For a given time series, there are several potential classes of models that could be used to describe its variation. Among them, the most frequent used are the autoregressive(AR) model, integrated(I) model and the moving-average(MA) model. The combination of these classes lead to the autoregres- sive moving-average(ARMA) model, the autoregressive integrated moving-average(ARIMA) model and the autoregressive fractionally integrated moving-average(ARFIMA) model. The user could choose the model class he/she wish to fit based on his understanding of the input time series.

3.3 Optimization

After the model type being fixed, there would be certain loss function regarding the selected model. The key of model fitting is to decide the parameters, which are derived by minimizing the loss function. Therefore, certain optimization techniques should be applied during the fitting. In our software, the user could choose the optimization tool between gradient descent, Newton method and stochastic gradient descent according to the scale of the problem.

3.4 Solver

We develop a solver module as the main driver program that uses both model and optimization method to find the solution. It allows the user to update the parameters in the model for one step each, and also allows the user to update them untill convergence.

3.5 Inference

After model fitting, in order to provide the user with more overall information about the model, our software also do inference work regarding the parameters. The software would construct confidence interval of given level for each parameter, conduct test to decide the non-zero parameters and calculate the corresponding P-value. This could help to identify the pattern of the model and help the user develop deeper understanding of the given time series.

3.6 Trading strategy

The software is able to predict the future price for each stock based on the fitted model. Hence, in the data visualization part, it would plot the estimated future price and an prediction interval along with the previous price that is already known. Together, some trading strategy would be made based on the prediction. The software could provide the user with the selling or buying point of certain stocks.

3.7 Option pricing

option pricing

3.8 Clustering

When a large number of time series are given, some of them may share similar patterns since they might be commonly affected by several intrinsic factors. With clustering techniques, the software would be able to identify the similar time series, i.e. stocks with similar variation, and divide them into groups. This would help the user gain a better knowledge of the stocks and their patterns.

3.9 Model reduction

model reduction

program structure, division of work, user interface, independent component analysis [1].

4 Demos of results

4.1 Modal validation

4.2 Finanical data

4.3 Trading strategy

4.4 Option pricing

4.5 Clustering

4.6 Model reduction

5 Lessons learned

References

[1] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.