Machine learning methods for stock return series analysis via Python

Jian Wang

Financial math Ph.D. candidate

Florida state university *jwang@math.fsu.edu*

April 7, 2015

Overview

Introduction

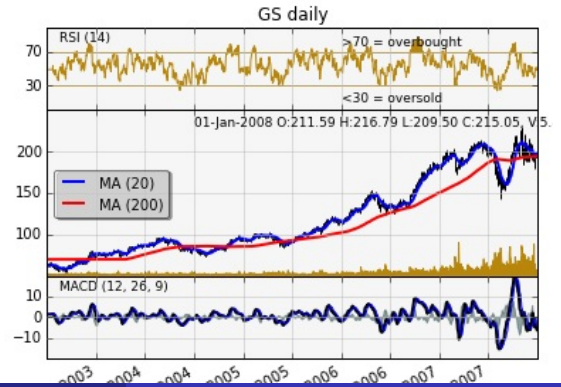
- **[Popular:]** Nowadays, Python becomes more and more popular in the financial industry. For example, Bank of America Merrill Lynch and JP Morgan Chase, use python to establish and enhance their core IT systems. In the mean time, many of today's financial math engineering programs use Python as one of the core languages for teaching the quantitative finance knowledge into executable computer code.
- **[Reason:]**There are a couple of reasons that Python has had such recent success. For example, it is an open source program language and its ease of integration with almost any other technology.Beisdes, the computing speed for python is fast, so it is a powerful tool to handle the big data.
- **[Purpose:]** In this project, we show that how to use python to analyze the financial data. Built machine learning models for stock return series. Compared the efficiency for language python and R. Finally, predict the future stock return based on the historical data.

Dataset

The dataset is named stockdata which from huge package in R. It contained data that were originally obtained from Yahoo! Finance. There are 1,258 observations representing 1,258 sequential trading days(form Jan 1 2003 to Jan 1 2008) and 452 variables, each of which was the day's closing share price for a different stock within the Standard & Poor's 500.We also add two index into the data set, one is S&P 500 and another is Nasdaq(so totally 454 variables).
Among all the stock data , we used Goldman Sachs stock return series as our response variable and other stocks as the predictors to analyze the stock return series movement.

Company::

Price::



Methodology

Stock Return series

$$R_t = ln(S_{t+1}) - ln(S_t) \tag{1}$$

Where R_t is the stock return at trading day t and S_t is the closing price of stock at trading day t.

Ridge regression

$$\hat{\beta}^{ridge} = argmin_\beta \{\sum_{i=1}^{p} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2\} \tag{2}$$

Lasso regression

$$\hat{\beta}^{lasso} = argmin_\beta \{\sum_{i=1}^{p} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j\} \tag{3}$$

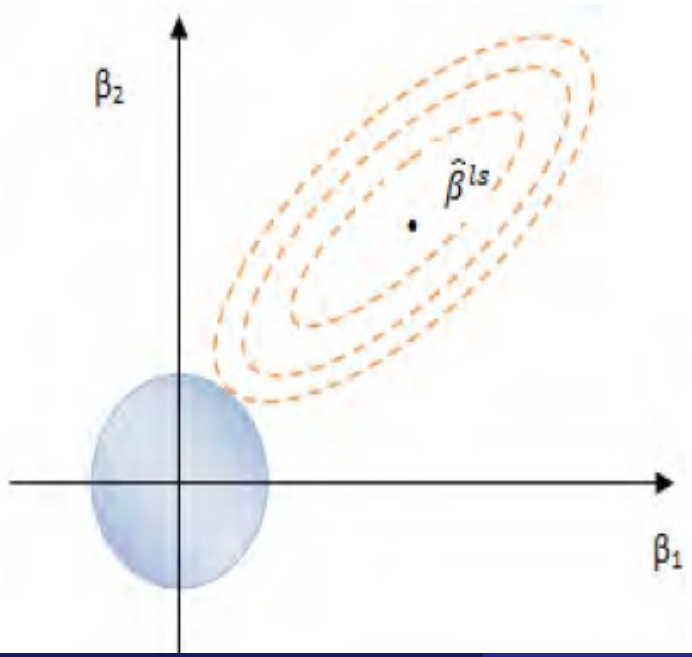
Methodology

Comparison of L1 and L2 Penalized Model

Ridge regression

$\hat{\beta}^{ridge} = argmin_{\beta}\{\sum_{i=1}^{p}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\}$

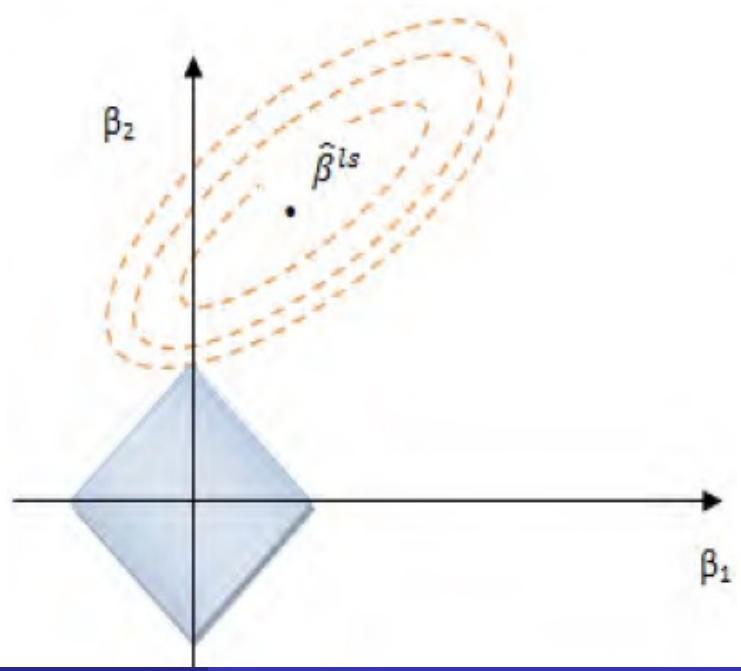
Coefficients:



Lasso regression

$\hat{\beta}^{lasso} = argmin_{\beta}\{\sum_{i=1}^{p}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}\beta_j\}$

Coefficients:



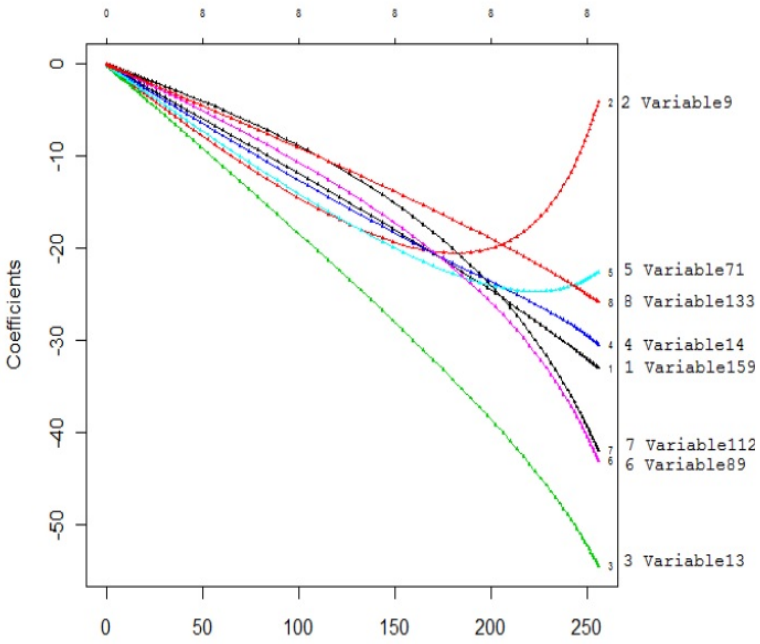
Methodology

Comparison of L1 and L2 Penalized Model

Ridge regression

$$\hat{\beta}^{ridge} = argmin_{\beta}\{\sum_{i=1}^{p}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\}$$

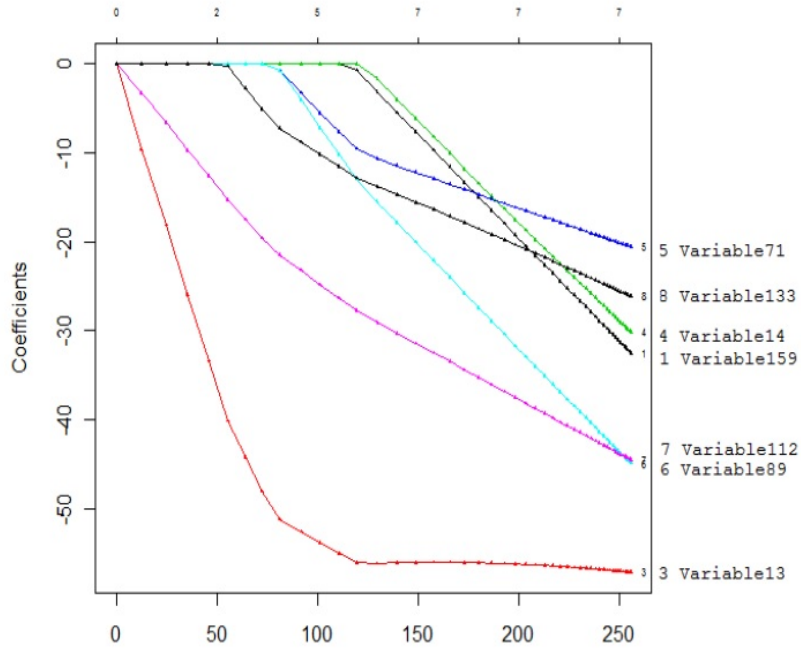
Path::



Lasso regression

$$\hat{\beta}^{lasso} = argmin_{\beta}\{\sum_{i=1}^{p}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}\beta_j\}$$

Path::



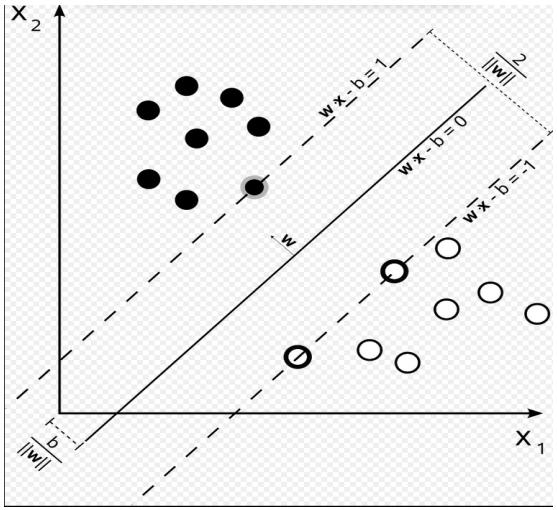
Methodology

Logistic Regression

$$ln\frac{F(x)}{1-F(x)} = \beta_0 + \sum_i \beta_i x_i \tag{4}$$

Support vector machine

$$min\sum_i [1 - y_i f]_+ + \frac{\lambda}{2}||\beta|| \tag{5}$$



Kernel Function:

1) Linear
2) Polynomial
3) Radial basis

Methodology

Reference

- Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso". Journal of the Royal Statistical Society, Series B 58 (1): 267288. JSTOR 2346178
- Hoerl, A.E. and Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12: 55-67
- Vapnik, V. (1995). "Support-vector networks". Machine Learning 20 (3): 273. doi:10.1007/BF00994018

Packages

- R packages: glm, glmnet, e1071
- Python packages: sklearn (svm, ridge, lasso, logistic)

Methodology

Reference

- Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso". Journal of the Royal Statistical Society, Series B 58 (1): 267288. JSTOR 2346178
- Hoerl, A.E. and Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12: 55-67
- Vapnik, V. (1995). "Support-vector networks". Machine Learning 20 (3): 273. doi:10.1007/BF00994018

Packages

- R packages: glm, glmnet, e1071
- Python packages: sklearn (svm, ridge, lasso, logistic)

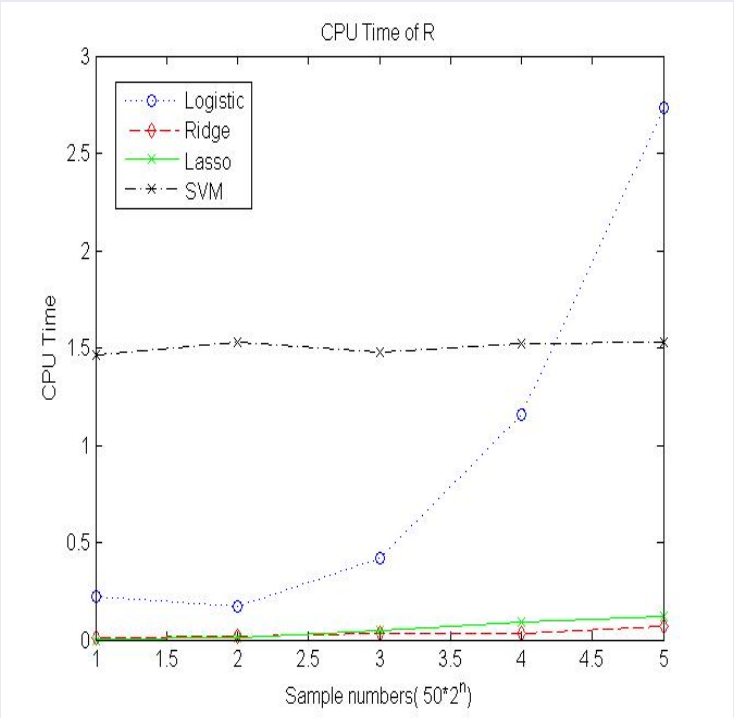
Numerical results

- Python and R package comparison:
  we use the same day stock return series to compare the cpu time and accuracy date of the machine learning methods based on different languages(R and Python). Choose the first 1000 data as training data and the last remaining 257 data as testing. GS as response and the other 453 stocks as predictors.
- Predict the stock data:
  we used the last one day, two day,... to five day stock returns as the predictors and today's GS return series as response to see if our model can be used to predict the stockdata.
  Still use the first 1000 data as training and the remaining 252 data as testing. GS as the response and the other 2270 variables as predictors.

Numerical results

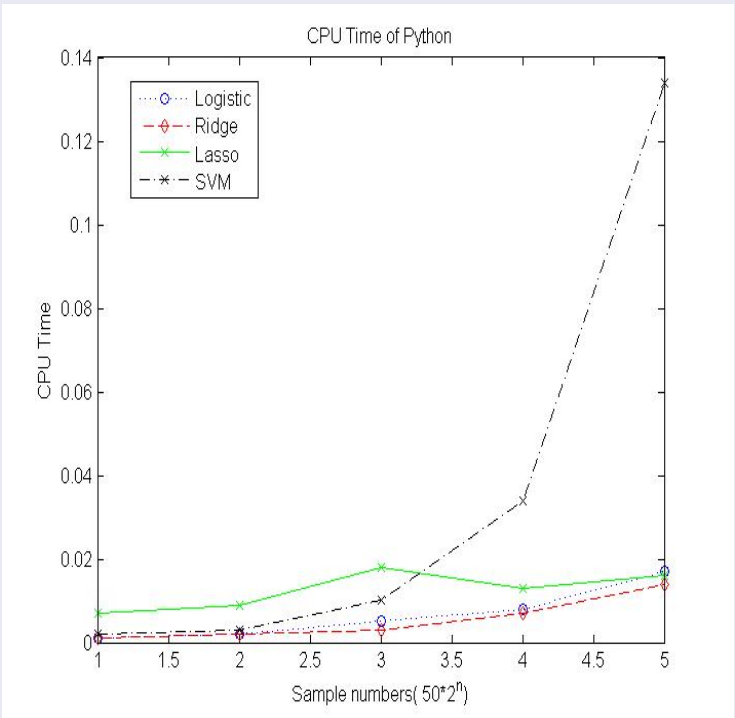
CPU Time for R

We changed the number of samples from 50 to 800, doubled each time to test the running time for the different machine learning methods:



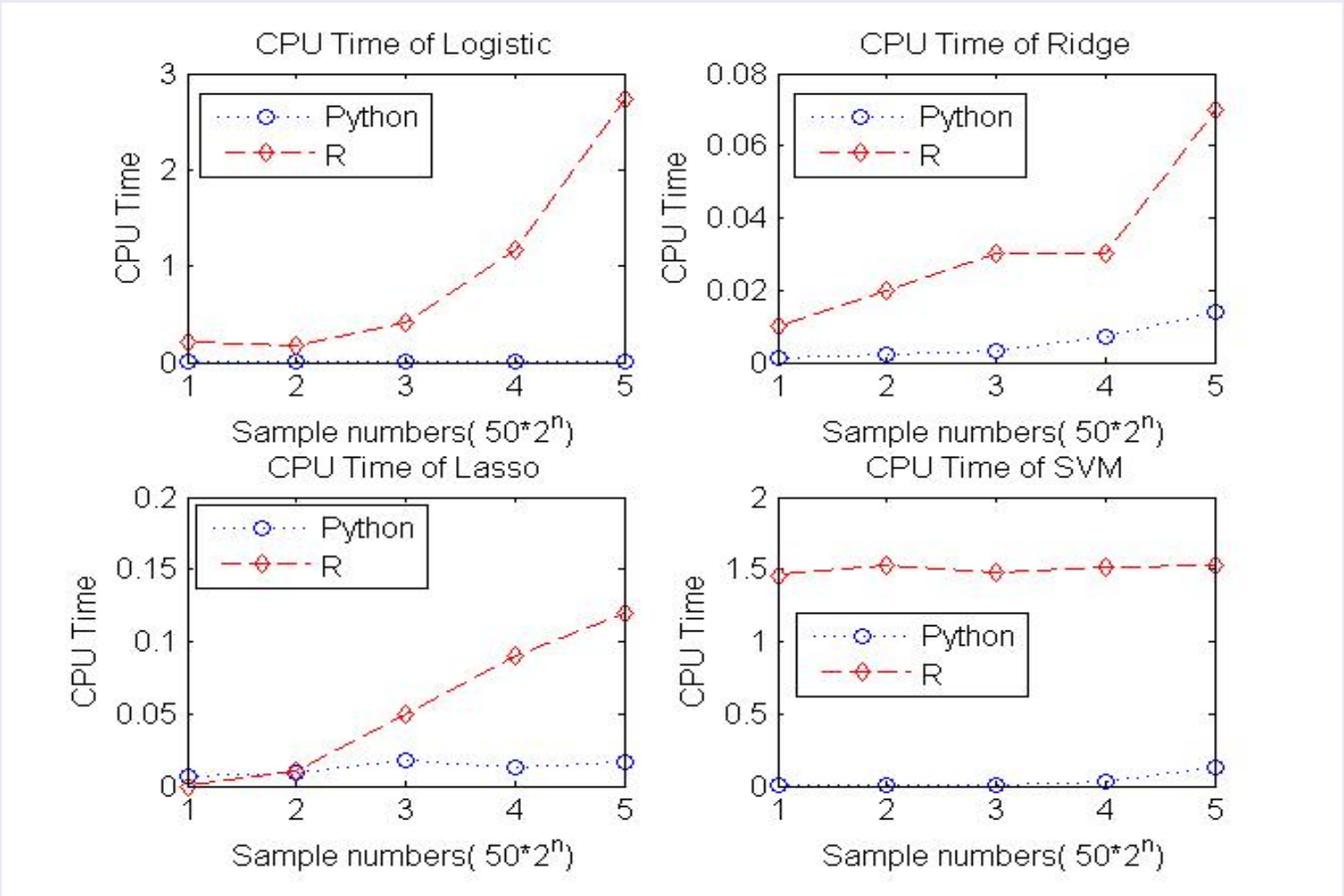
CPU Time for Python

We changed the number of samples from 50 to 800, doubled each time to test the running time for the different machine learning methods:



Numerical results

Comparison of CPU time for python and R



Results:

Numerical results:

Accuracy rate:

Table: Accuracy rate

Methods	Python	R
Logistic	77.8%	65.4%
Ridge(λ=1)	73.9%	77.4%
Lasso(λ=0.01)	78.6%	79.0%
svm(linear)	72.4%	65.8%
svm(poly)	74.3%	65.0%
svm(rbf)	75.1%	72.8%

Results:

- svm (rbf) and lasso (λ=0.01) performed better for both two languages.
- Python performed better in most cases.

Numerical results:

Predicted

$$R_t^{GS} = \sum_{i=1:5} \sum_{j=1:454} \beta_{i,j} R_{t-i}^{j} \tag{6}$$

Predict:

Table: Accuracy rate and CPU time

Methods	Accuracy rate	CPU time
Logistic	51.2%	0.1210
Ridge(λ=1)	54.0%	0.1230
Lasso(λ=0.01)	49.2%	0.0940
svm(linear)	52.8%	1.1931
svm(poly)	45.6%	1.2800
svm(rbf)	47.2%	1.2921

Future work

- Compare with the time series model, such as Garch
- Deal with the high frequency data instead of daily data

QA

Thanks a lot and Questions