Python final project report: Machine learning methods for stock return series analysis via python

Due on 2015/05/03

Instructor:Dr.Jinfeng Zhang 3:30PM

Jian Wang

Contents

Problem 1

[Summary]

1) Python consumed less CPU time than R for Logistic, Ridge, Lasso regression and support vector machine methods.

2) lasso (λ=0.01) and svm (rbf) performed good for both two languages.

3)Python performed better in most cases but the difference between the two languages was not significant.

4) For the same day return series analysis, the results were fairly good, however, for the prediction models, the results were close to random guess.

5) May reflected the efficient market theorem and can use the high frequency data to see if the results for prediction can be improved.

[Introduction]

- **Machine Learning:** Machine learning is a method of teaching computers to make predictions based on historical data. It is one important branch of artificial intelligence research area. Nowadays, those popular algorithms have been widely used in our financial engineering research area, for example, prediction of stock trend, deal with big data problem such as limit order book problem.
- **Goal:** In our research, we try to deal with the following tasks: **First**, combine Bayesian network and graphical lasso algorithm to deal with the high dimensional problem. **Second**, compare the efficiency of machine learning packages of language R and python. **Finally**, use machine learning methods to deal with the real world stock return data and try to predict the stock return series via those machine learning methods.

[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 price for a different stock within the Standard & Poor's 500.We also added 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 and price::

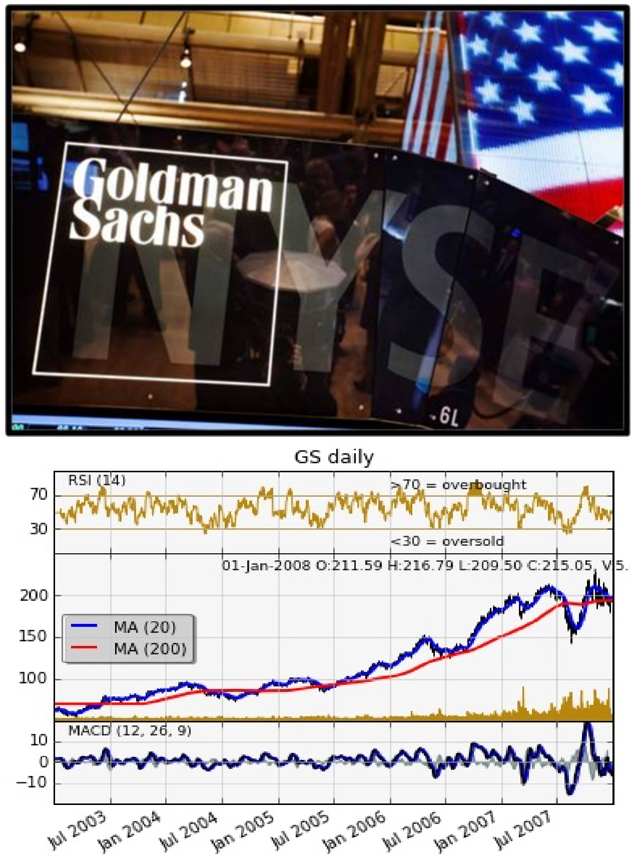

Figure 1: Goldman Sachs Company and prices

[Methodology]

1.Bayesian network

BN(Bayesian network) is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG)

The following chart is an example of BN:

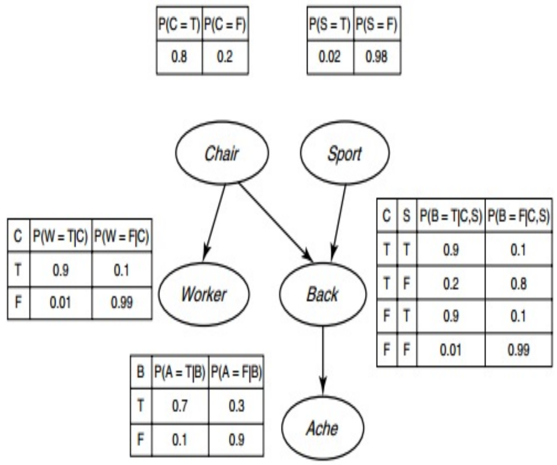

Figure 2: Bayesian networks example

Our main concern is on the structure learning process since the structure learning process is exponentially increasing complicated and is the most challenging part in Bayesian network research area. The standard that we used to choose the optimal structure is the score based method.
BIC score:$BIC = -2 * ln\hat{L} + K * ln(n)$, where:

$\hat{L}$ = maximum likelihood estimator, k is the number of free parameters and n is the number of observations, ie, the sample size.

2.Regression models
1) Linear regression:

$$\hat{\beta}^{ls} = argmin_{\beta}\{\sum_{i=1}^{p}(y_i - \hat{y}_i)^2\} \tag{1}$$

2) Logistic Regression

$$ln\frac{F(x)}{1-F(x)} = \beta_0 + \sum_i \beta_i x_i \tag{2}$$

3) Ridge regression:

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

Coefficients and paths:

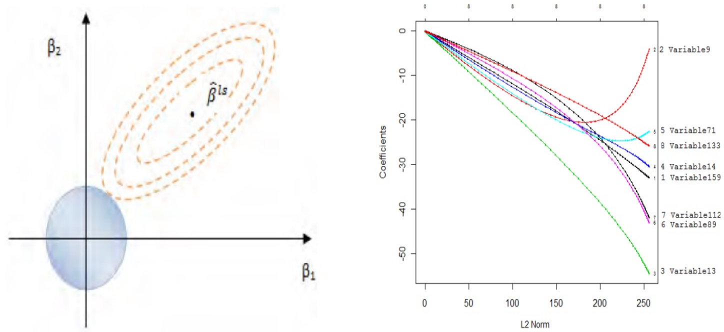

Figure 3: Ridge regression coefficients and Paths

4) Lasso regression:

$$\hat{\beta}^{lasso} = argmin_{\beta}\{\sum_{i=1}^{p}(y_i - \hat{y_i})^2 + \lambda \sum_{j=1}^{p} |\beta_j|\} \tag{4}$$

Coefficients and paths:

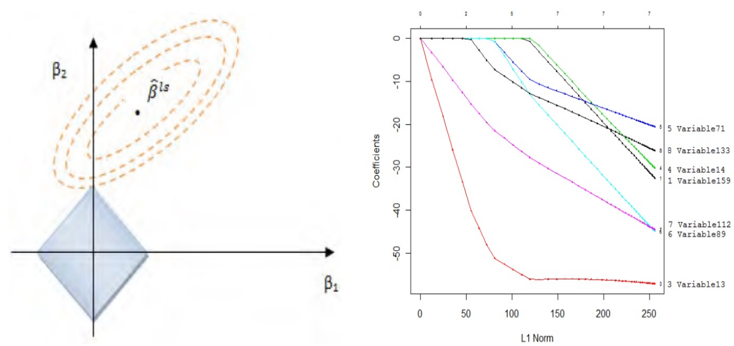

Figure 4: Lasso regression coefficients and paths

3.Combined models

1) Graphical lasso: Suppose we have N multivariate normal observations of dimension p , with mean μ and covariacne Σ. Let $\Theta = \Sigma^{-1}$ and S be the empirical covariance matrix, the problem is to maximize the log-likelihood

$$lnP(X|u, \Sigma) = -\frac{N}{2} ln|\Sigma| - \frac{1}{2}\sum(x_n - u)^T \Sigma^{-1}(x_n - u)$$

combined with the L_1 penalty

$$ln|\Theta| - tr(S\Theta)) - \lambda||\Theta||_1$$

Algorithm:

Many algorithms for this problem, The following might be the oldest and simple one by Meinshausen and Buhlmann(2006)problem

- Estimate a sparse graphical model by fitting a lasso model to each variable, using others as predictors
- Set Σ_{ij}^{-1} to be non zero, if either the estimated coefficient of variable i on j, or the estimated coefficient of variable j on i, is non-zero

2) Bayesian-Glasso model
For the high dimensional problem, it is not very easy to built the Bayesian network due to its exponentially increasing complexity.
Our idea is to first use the Glasso model to conduct the model selection and then use Bayesian network structure learning process to define the network structure.
Algorithm:

- Use Glasso algorithm to find the edges among variables
- Use greedy search methods to change the direction only on those existed edges
- Choose the direction which has the optimal BIC score
- Finish when all the edges are reached or attain the maximum iteration numbers

3) Support vector machine

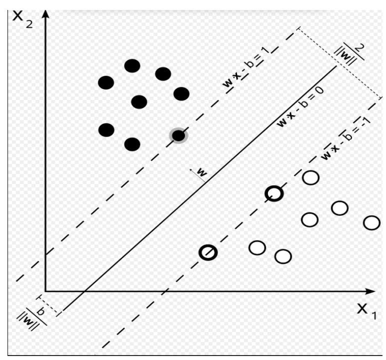

Figure 5: Support vector machine

Try to maximize the margin: $r = 1/||w||, y_j = 1, -1$

Primal form:
$\max\limits_{W,b}\ r = 1/||W||$
$s.t.(W^T x_j + b)y_j >= 1$

Dual form:
$\max\limits_{\alpha_1,\ldots,\alpha_M} \sum \alpha_l - \frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{M} \alpha_j\alpha_k y_j y_k < X_j, X_k >$
s.t.$\alpha_l \geq 0$, $\sum_{l=1}^{M} \alpha_l y_l = 0$

The SVM was trying to deal with high dimensional problems:
Sometimes, in lower dimension we can not separate the data properly, so we need to project the data to the high dimensions.
Examples:

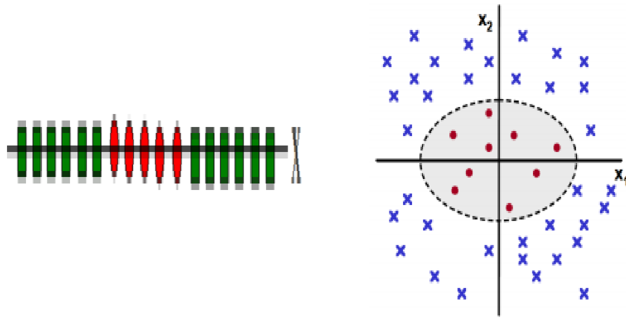

Figure 6: 1d to 2d example

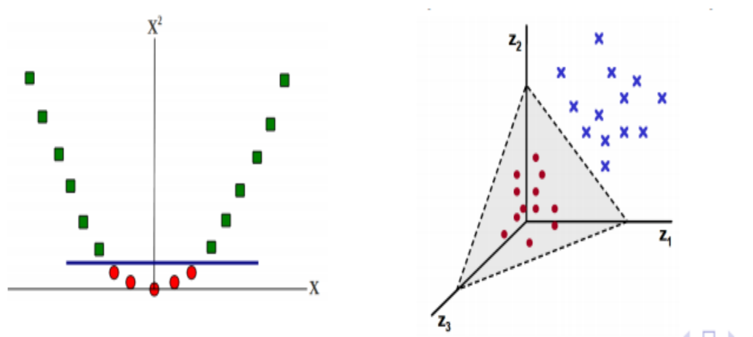

Figure 7: 2d to 3d example

Kernel functions
We can use the kernel function to calculate the inner product in high dimensional cases in its original feature spaces. Example:
$k(x, z) = (x^T z)^2$
$= (x_1 z_1 + x_2 z_2)^2$
$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2$
$= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T (z_1^2, \sqrt{2}z_1 z_2, z_2^2)$
$= \Phi(x)^T \Phi(z)$

Kernel functions that we used:

- Linear kernel

- Polynomial Kernel

- Radial basis function kernel

[Numerical results]

- Same day stock return series analysis:
  we use the same day stock return series to build the machine learning models. For the Bayesian network, we only use the R package. For the logistic regression, ridge regression, lasso and svm, we used different languages(R and Python) and also compare the CPU time. To test the accuracy rate of model, we choose the first 1000 data as training data and the last remaining 257 data as testing. GS as response(discretized as 1 and -1) and the other 453 stocks as predictors.

- Predict the stock data:
  we used the last one day, two day,... to last 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.

1. For the same day return series analysis:
Simple example for Bayesian networks, we chose 10 different world class company from 5 industries, 2 companies from each.

Table 1: 10 companies

Stock code	Industry	Company name
GS	Financials	Goldman Sachs Group
JPM	Financials	JPMorgan Chase & Co.
MSFT	Information Technology	Microsoft Corp.
IBM	Information Technology	International Bus. Machines
T	Telecommunications Services	AT&T Inc
VZ	Telecommunications Services	Verizon Communications
WMT	Consumer Staples	Wal-Mart Stores
KO	Consumer Staples	Coca Cola Co.
AMZN	Consumer Discretionary	Amazon.com Inc
BBY	Consumer Discretionary	Best Buy Co. Inc.

Results for simple cases:

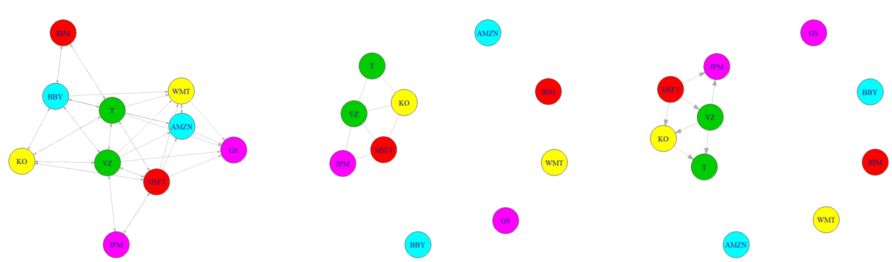

Figure 8: Bayesian network structures for 10 stock companies

Results for total stocks:

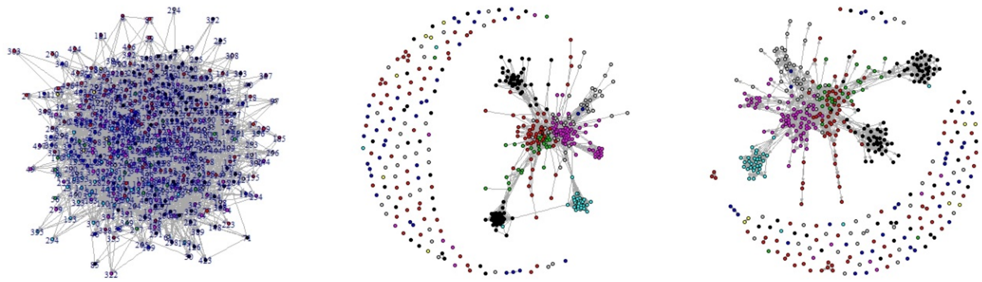

Figure 9: Bayesian network structures for total stock companies

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

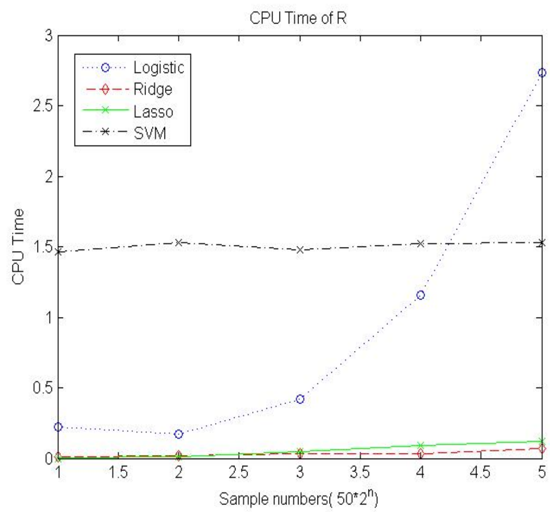

Figure 10: CPU time for R

For Python:

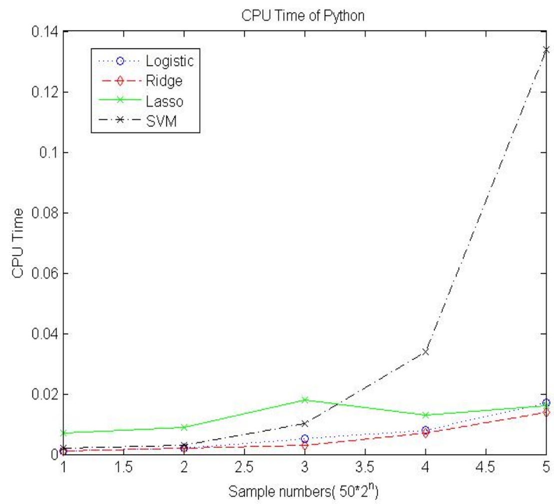

Figure 11: CPU time for python

For R and Python:

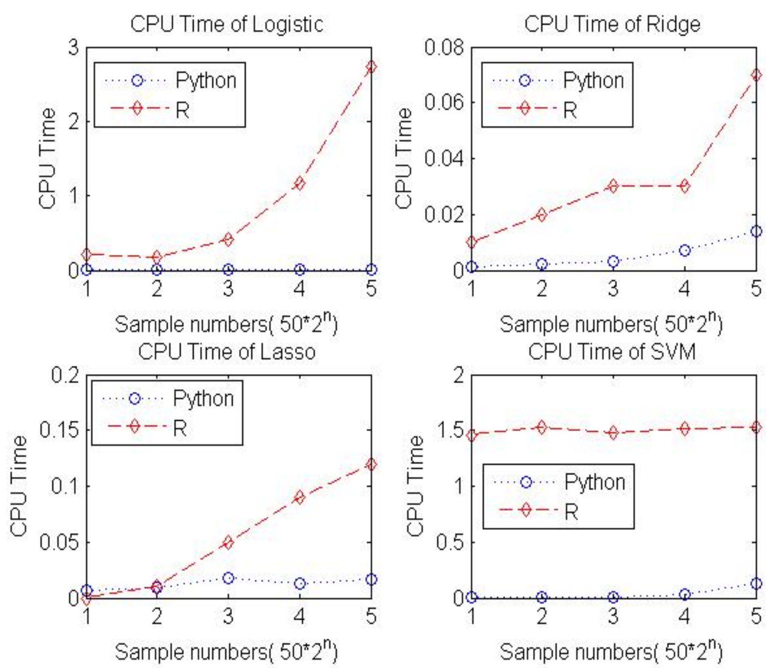

Figure 12: CPU time for R and python

Accuracy rate:

2.Prediction analysis

Table 2: 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%

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

Table 3: 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