Comparison of machine learning methods for stock return series

Jian Wang

Financial math Ph.D. candidate

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

October 27, 2015

Overview

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.

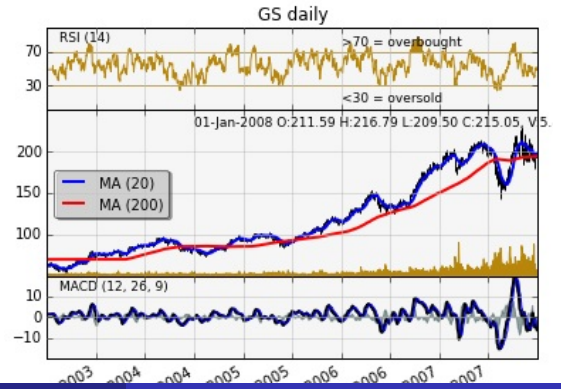
Dataset

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.

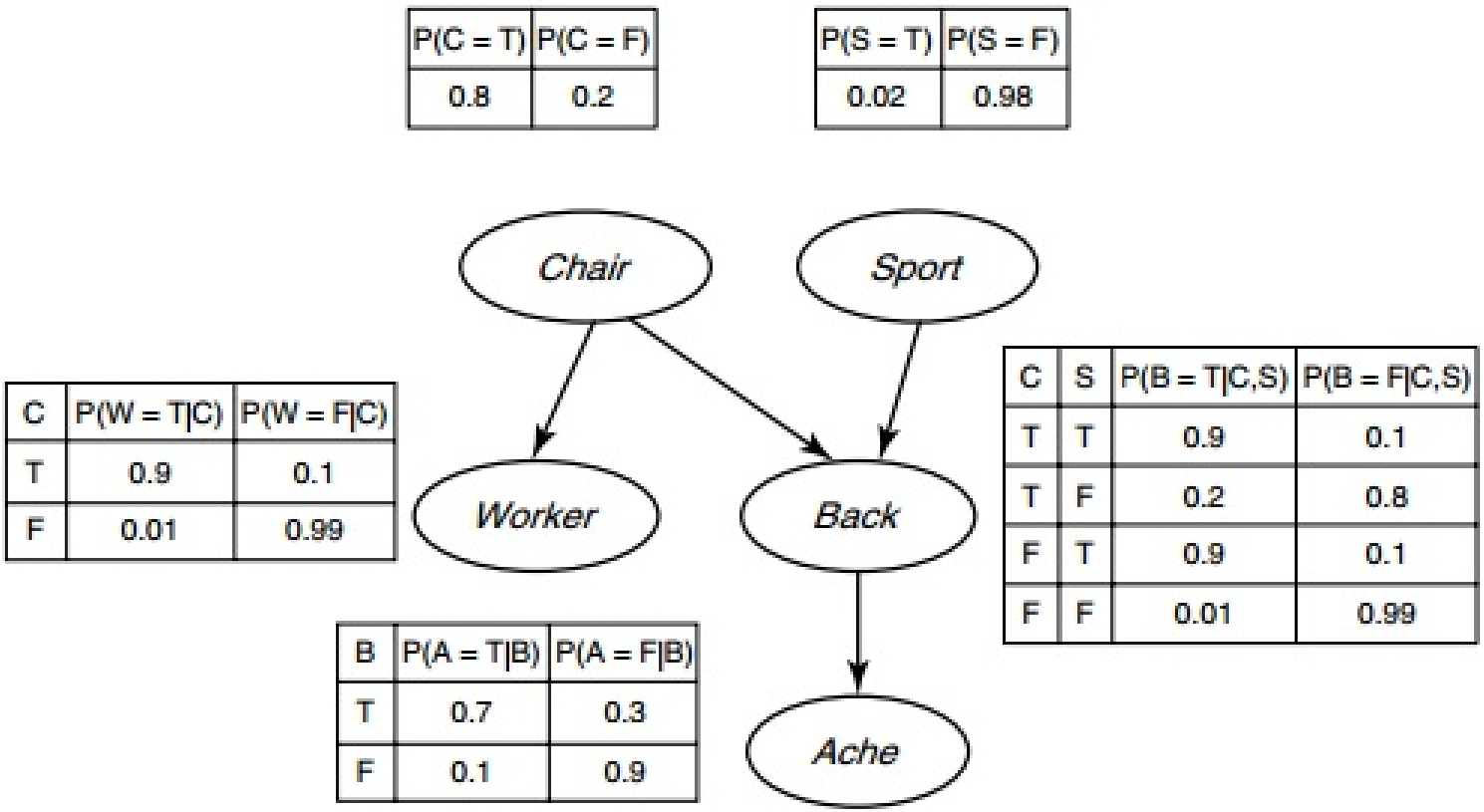
Company::

Price::



Methodology

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:

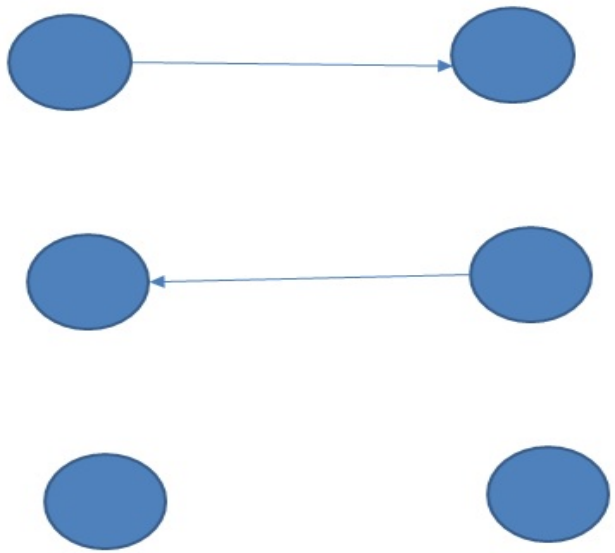


Methodology

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.



Methodology

Linear regression

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

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

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

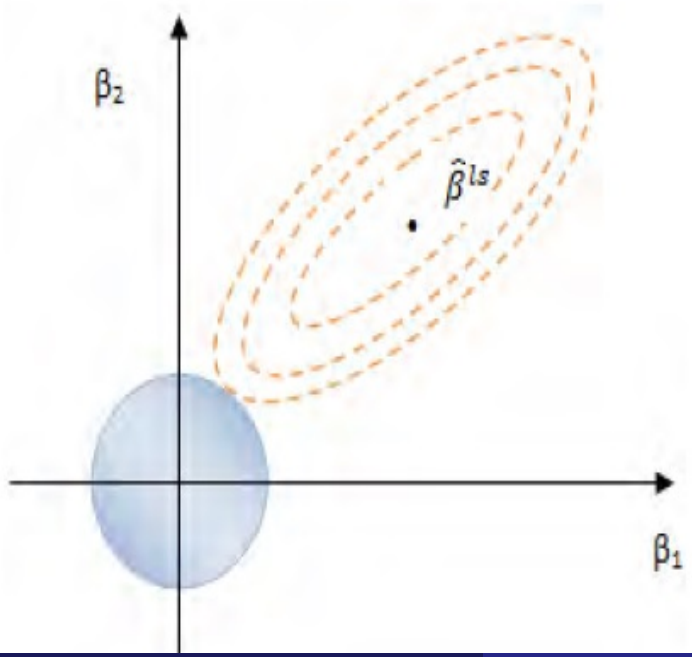
Methodology

Comparison of L1 and L2 Penalized Model

Ridge regression

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

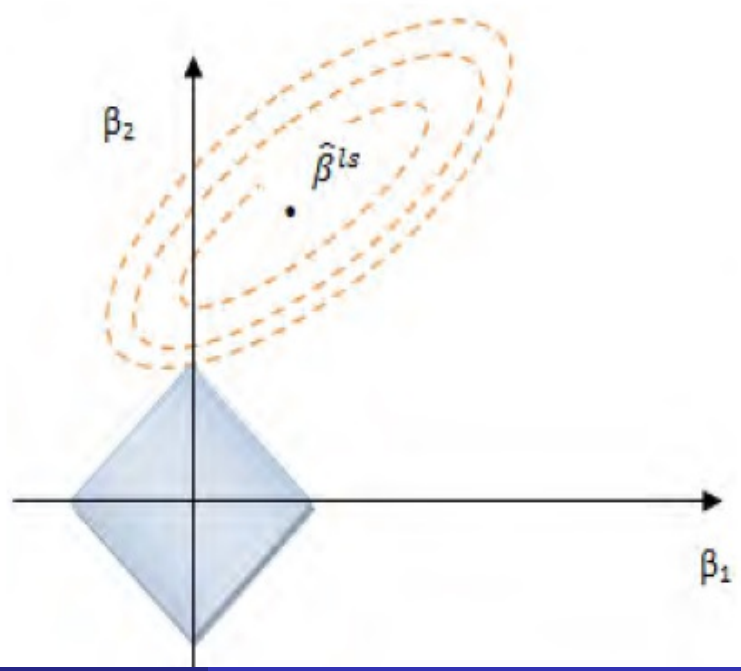
Coefficients:



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:



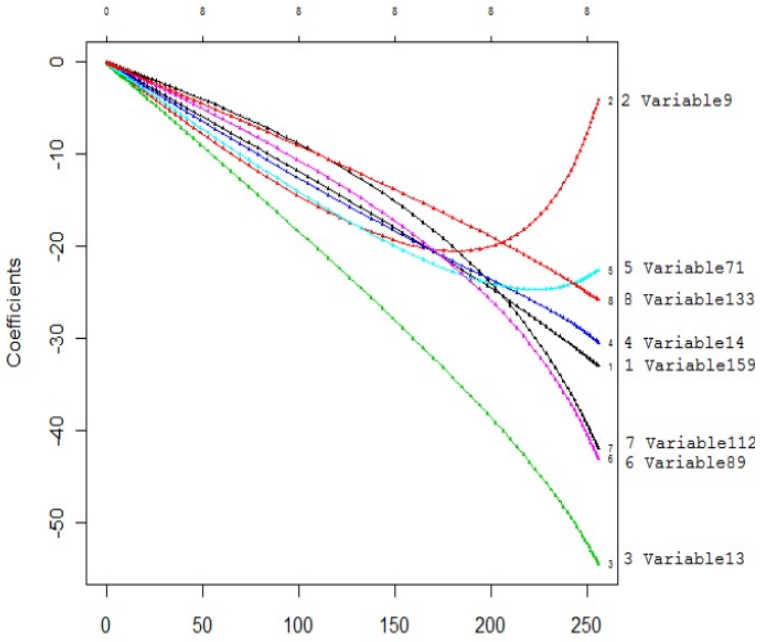
Methodology

Comparison of L1 and L2 Penalized Model

Ridge regression

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

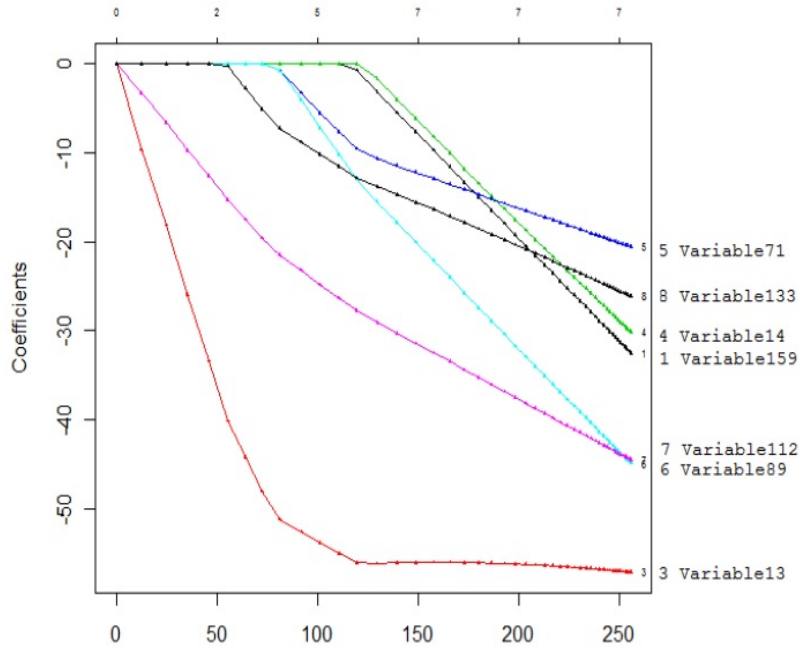
Path::



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

Glasso:

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)

- 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

Methodology

Glasso:

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)

- 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

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

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

Methodology

Logistic Regression

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

Support vector machine

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

Primal form:

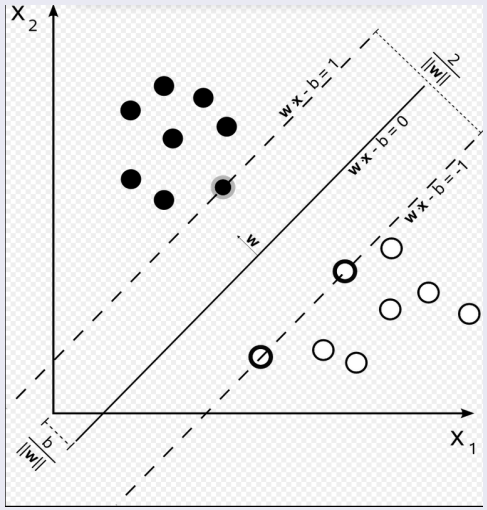
$\max_{W,b} \; r = 1/||W||$

$s.t.(W^T x_j + b)y_j >= 1$

Dual form:

$\max_{\alpha_1,\ldots,\alpha_M} \; \sum \alpha_l - \frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{M} \alpha_j\alpha_k y_j y_k < X_j, X_k >$

s.t.$\alpha_l \geq 0$, $\sum_{l=1}^{M} \alpha_l y_l = 0$

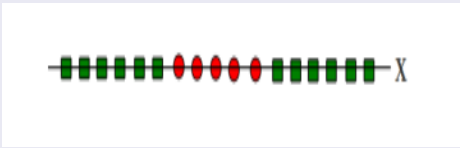


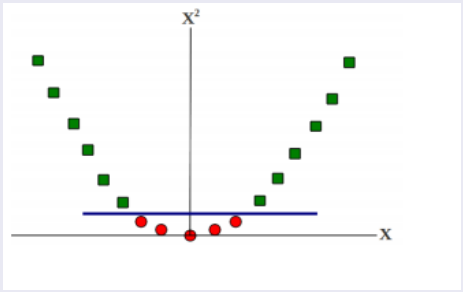
Methodology

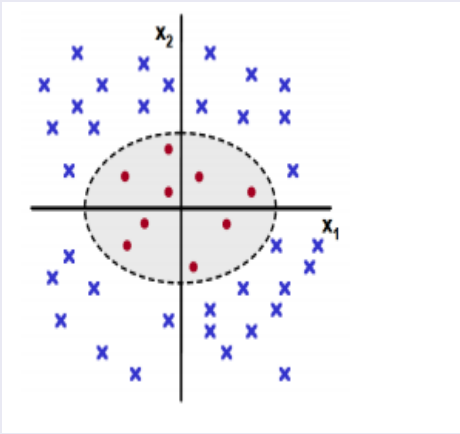
Higher dimensional situations

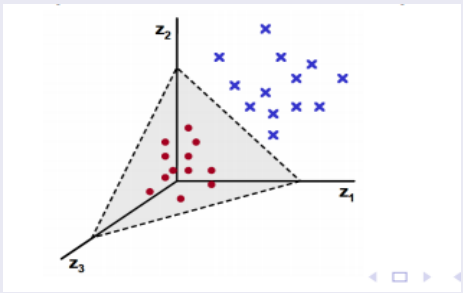
sometimes, in lower dimension we can not separate the data properly, so we need to project the data to the high dimensions

Examples









Methodology

Kernel functions

We can use the kernel function to calculate the inner product in high dimensional cases in its original feature spaces.

Example

$$
\begin{aligned}
&k(x,z) = (x^T z)^2 \\
&= (x_1 z_1 + x_2 z_2)^2 \\
&= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\
&= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T (z_1^2, \sqrt{2} z_1 z_2, z_2^2) \\
&= \Phi(x)^T \Phi(z)
\end{aligned}
$$

Kernel functions that we used

- Linear kernel
- Polynomial Kernel
- Radial basis function kernel

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,bnlearn,huge
- 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,bnlearn,huge
- Python packages: sklearn (svm, ridge, lasso, logistic)

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.

Numerical results

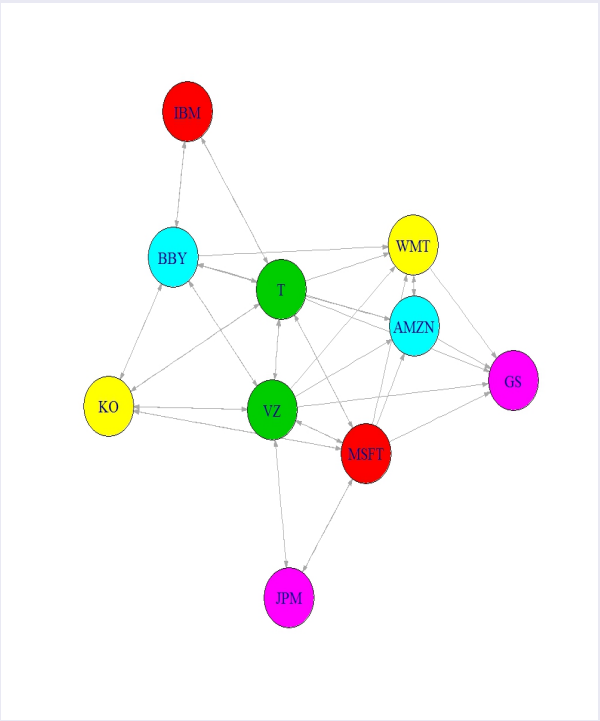
Bayesian network

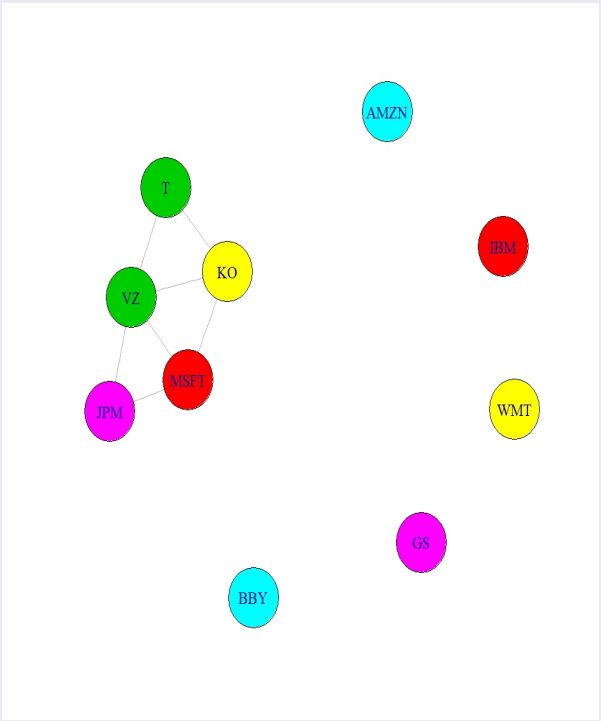
Table : 10 companies

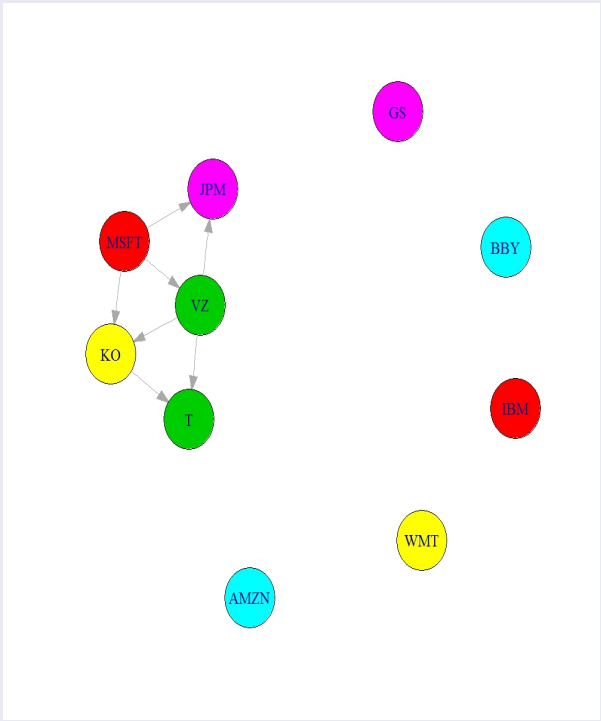
Stock code	Industry	Company name
GS	Financials	Goldman Sachs Group
JPM	Financials	JPMorgan Chase & Co.
MSFT	Information Technology	Microsoft Corp.
IBM	Information Technology	International Bus. Machines
T	Telecommunications Services	AT&T Inc
VZ	Telecommunications Services	Verizon Communications
WMT	Consumer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

Numerical results

Results for simple cases

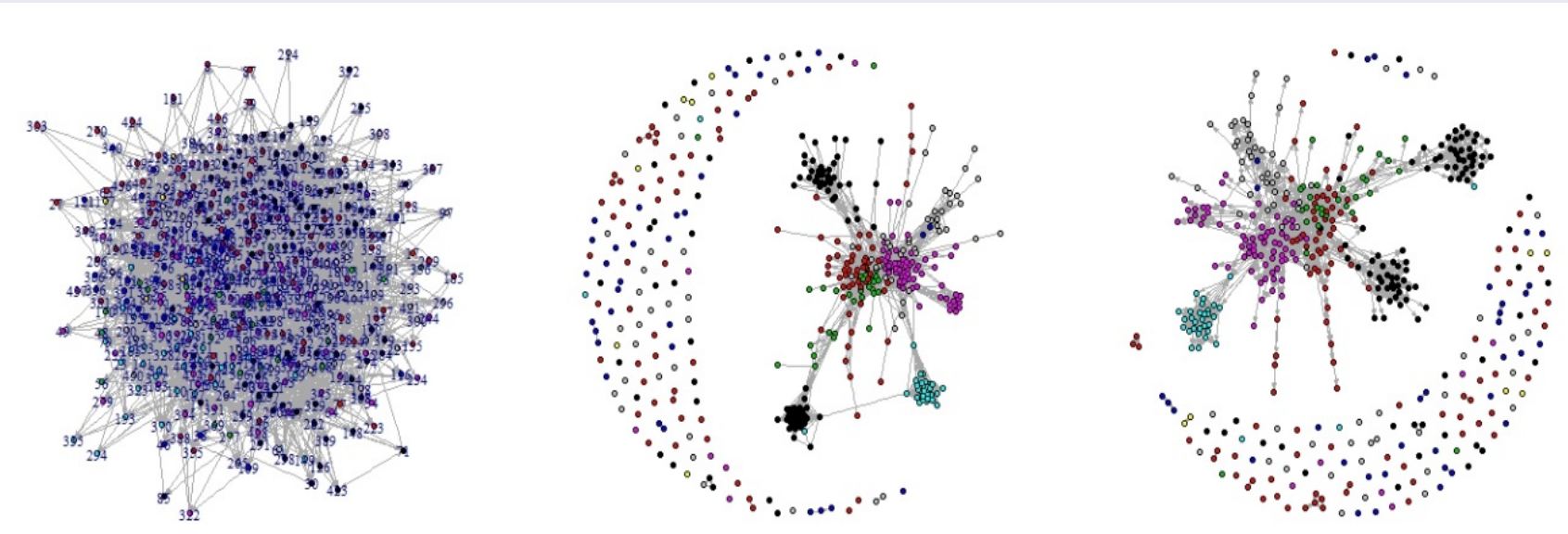






Numerical results

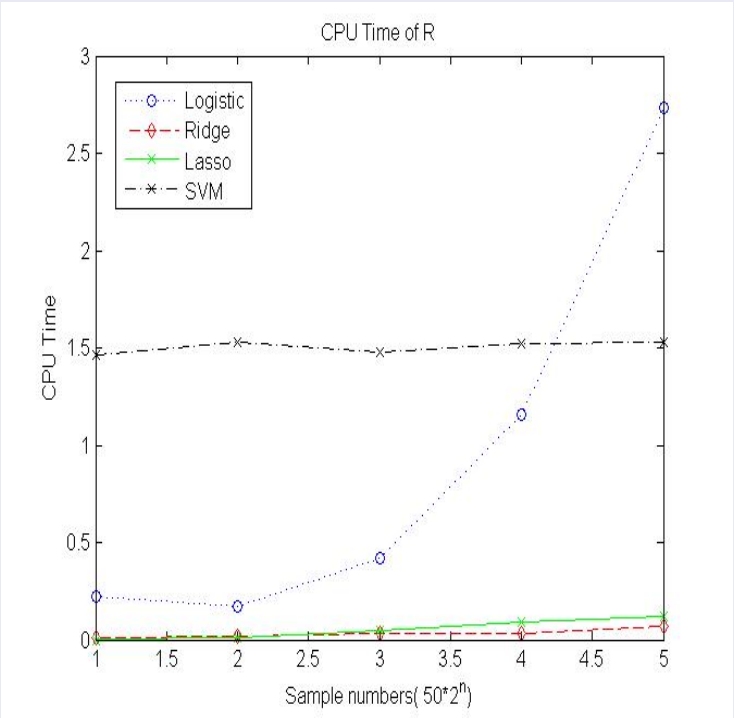
Results for total stocks



Numerical results

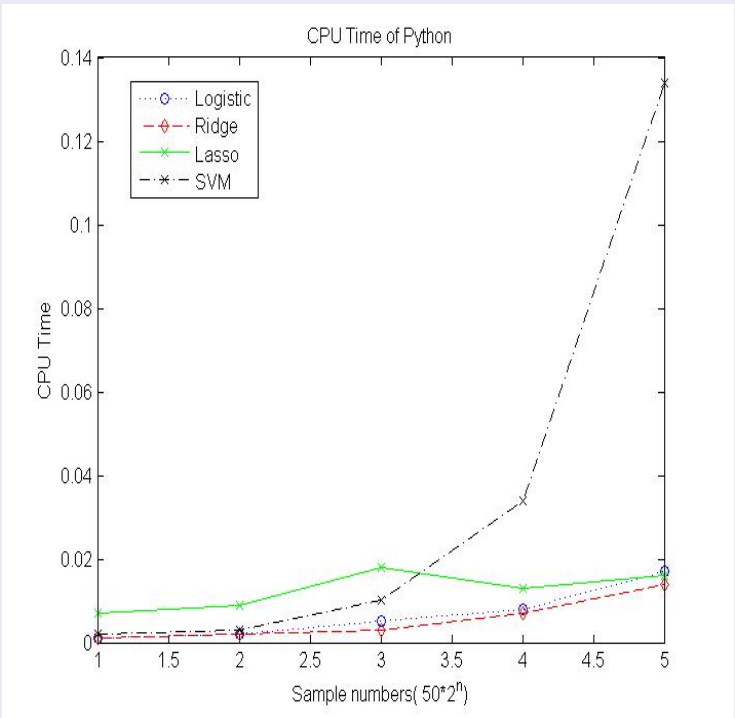
CPU Time for R

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



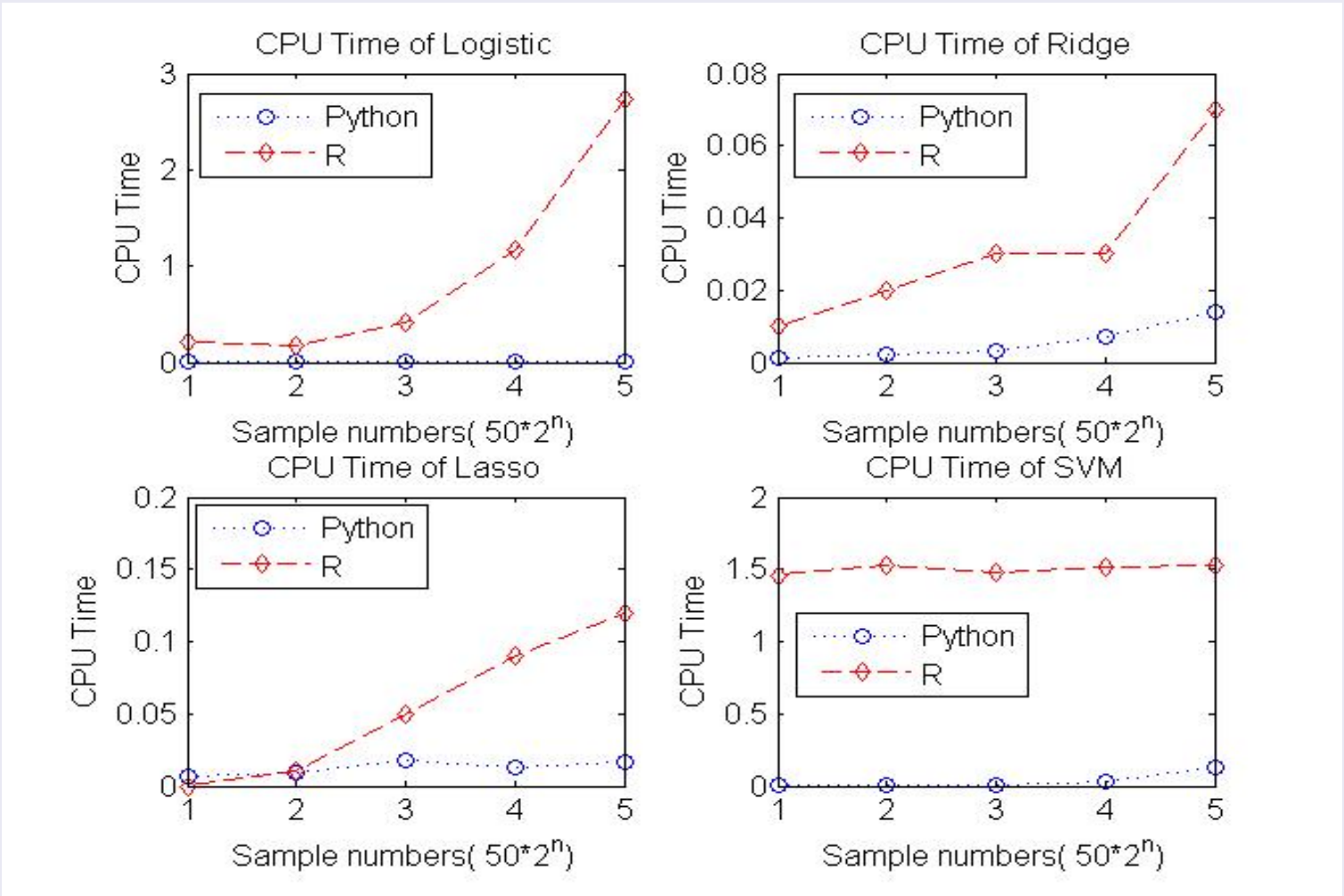
CPU Time for Python

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	69.8%	68.4%
Ridge(λ=1)	73.9%	77.4%
Lasso(λ=0.01)	78.6%	79.0%
svm(linear)	72.4%	71.8%
svm(poly)	74.3%	71.2%
svm(rbf)	75.1%	72.8%

Results:

- lasso (λ=0.01) and svm (rbf) performed good for both two languages.
- Python performed better in most cases but the difference is not significant.

Numerical results:

Predicted

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

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
Bayesian Glasso	51.6%	around two hours

Future work

- Compare with the time series model, such as Garch(machine learning methods can consider the whole economic environment while time series cannot)
- Deal with the high frequency data instead of daily data

QA

Thanks a lot and Questions