CVaR Optimization and Multivariate Joint Density Modeling

by

Yi-Fei Chen

B.Sc., Peking University, 2003
M.Sc., The University of British Columbia, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Applied Mathematics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

February 2011

Abstract

One attractive candidate as standard risk metric is Conditional Value at Risk (CVaR), which has a lot of advantages compared with Value at Risk (VaR). In this paper, I study CVaR as an objective function in a series of optimization problems. I compare the CPU time of the linear programming approach proposed in [14] with that of the fast gradient descent method [5] when increasing the number of scenarios and assets, respectively. The discrepancy between the two methods are also discussed. I also fit the market data to a set of joint densities and compare the market efficient frontiers and the returns ex-post. Yi-Fei Chen, y.chen@math.ubc.ca

Table of Contents

Appendices

List of Figures

Acknowledgements

I am grateful for much support from many people, particularly, my supervisor Ivar Ekeland. Even in the days that he is not in Vancouver, it is always him to remind me that I have to drag my lazy body off the bed to keep up with my schedule and talk with him in the morning of Thursday every week, wherever he is, Paris, Vienna or some random places on the map. I am really grateful to Ivar for helping me to complete my project and for pointing out the weakness of my essay.

Christopher Bemis, Sohhyun Chung, Fernando Fontove, Daniel Jordan and Heng Ye have worked with me on this project and provide constructive conversation which help me in writing the essay.

Lee Yupitun, Marlowe Dirkson and Verni Brown in the math office have provided me enormous help in my research, daily study and teach duty. Johnny Cao provided delicious food and drink every week. I wouldn't have experienced the wonderful two years without you guys.

Li Wang, Jie-Ren Wang, Zi-An Zhao, Tao Cheng, Sa Zhai, Zheng-Zheng Yang, Xiao-Hu Ji, Jia Gou have often been around in the office day and night to talk about the turmoil financial world nowadays and help me in finding what I can contribute in the future.

Thanks, all!

To Kairong and Xunyuan

Statement of Co-Authorship

This project was designed by Dr. Chris Bemis, who is a fund manager at Whitebox LLC. The initial work was done in a 10-day summer workshop about the math modelling in the industry.

In the whole project, I wrote all the codes provided at the back of this essay. I did all the data analysis. After the workshop, I work extensively on this project and further studied the possibilities of fitting our data to the several multivariate distributions other than the normal distribution. Introduction and conclusion were included to complete this essay. The two examples to visibly explain the advantage of CVaR at the beginning of the Chapter 2 are included by me.

Chapter 1

Introduction

When managing a portfolio of assets, the financial risk of the underlying securities is always a big concern of investors. Some quantitative property of the joint density distribution of a pool of assets is chosen to proxy risk. One of the very fist formulations is the Capital Asset Pricing Model (CAPM) [15], where the return of the securities is assumed to have a joint normal distribution, therefore the variance is a perfect choice of risk metric. In this case, as a manager of a pool of assets, the objective becomes minimizing the variance under the constraints about the expected return and allocation rates as all and for individual assets as well. This optimization problem was first brought up by Markowitz in 1952 [8]. Although, normal distribution is popular and widely used due to its easy implementation, it may not be a best choice for the returns of securities in the reality. Therefore, alternative risk metric is considered when using different type of distributions.

We particularly consider two types of risk of measure: Value at Risk (VaR) and Conditional Value at Risk (CVaR). VaR_β is a method of estimating the exposure of a portfolio of assets to potential losses. Suppose that we have a portfolio of assets whose return satisfies certain probability distribution. VaR_β is the minimal potential loss that the investor could suffer with the probability of $1-\beta$. In other words, with probability β, the loss will be less than VaR_β. VaR_β is the most popular risk metric because of the suggestion in the *Besel II accord*. But it has its intrinsic flaw intuitively, which is not coherent due to its non-subadditivity [14]. Also, it doesn't tell what we will expect to lose by just giving the minimal loss. In such a sense, CVaR_β (also called expected shortfall) is more useful and reasonable, which is the expected loss exposed to the investor if the losses do exceed the value of VaR_β. With the risk metric of VaR or CVaR, as a manager of a portfolio of assets, we aim to minimize the value of both VaR and CVaR as objective functions under certain set of constraints. Compared to the risk measure of the variance used by Markowitz [8], there have many appealing properties with CVaR_β. One of them is that it leaves the task of defining the joint density distribution to use to practitioners. However, if the return/loss associated with each asset is normally distributed, $\beta \leq 0.5$ and

the constraint on the value of expected return is active, then the solutions to the three problems of (P1) minimizing VaR_β, (P2) minimizing CVaR_β and (P3) minimizing σ^2 (x) are the same [14] .

At this time, the speed and the accuracy of the optimization algorithm are the two biggest concerns. Since the lack of enough historical data to figure out the very accurate joint density distribution, if possible, we would like to sacrifice certain accuracy to gain the speed of the optimization algorithm. In the following sections, we compared the CPU time and accuracy of several algorithms. We focused on two optimization algorithms for CVaR. One of them is proposed by [14], which converts the process of finding the minimal CVaR_β into a linear programming problem with introducing extra variables, while the fast gradient descent method suggested by [5] uses Nesterov procedure [11] to estimate the CVaR_β within a preset acceptable error and involves three quadratic programming for every iterations. It claimed to be faster for a larger number of assets. The detail of the comparison of these two methods are in the Chapter 2.

Since the market is non-stationary, so we won't track back very far in the historical data. The market data of three years ago are not necessarily have the predictability for the return of the portfolio now. So with the limited size of data sets, we would like to find some more stationary characteristics of the market, and then generate the scenarios based on them and find the optimal allocation of assets. We would like to find a better joint density distribution in terms of better fitting and consistent better ex-post test result. If necessary, besides the several popular joint density (e.g. normal, student-t, skewed student-t, normal inverse Gaussian), mixture density is considered. The efficient frontiers for the portfolio are also calculated and plotted with the historical data fitted by different statistical distribution models. Details are discussed in the Chapter 4.

Chapter 2

CVaR Optimization

Suppose we have the loss function for a given portfolio $f(w,\, y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, where the allocation is $w \in \mathbb{R}^n$ and the 'scenario' or return $y \in \mathbb{R}^m$ and p, the probability distribution of the returns. We define the cumulative distribution function (CDF) as

$$\Psi(w, \alpha) := \int_{f(w,y)\leq\alpha} p(y)\, dy.$$

Then the VaR_β of a given portfolio w is defined as the smallest possible value of α where the probability of the loss exceeding α is no more than $1 - \beta$. Express this in formula, it is

$$\mathrm{VaR}_\beta(w) := \min\left\{\alpha \in \mathbb{R} :\ \Psi(w, \alpha) \geq \beta\right\}.$$

However, VaR_β just tells about the minimal possible loss that investor will suffer. It doesn't give much information about the extreme case. For instance, if we have a series of 10 daily returns sorted in the descending order as,

$$[-1\%,\, -2\%,\, -3\%,\, -4\%,\, -5\%,\, -6\%,\, -7\%,\, -8\%,\, -9\%,\, -50\%]$$

Based on this set of historical data, the maximal daily loss is -50%. However, the risk given by the value at risk with $\beta = 0.9$ is only $\mathrm{VaR}_{0.9} = -9\%$, which may make the practitioner mistakenly estimate the possible loss in the prevailing market if the real distribution of the market return has a fat tail. In the mean while, the conditional value at risk CVaR, taken as the average loss if the investor did suffer a bad loss in the extreme case, can give a more reasonable estimation. In our example shown above, the CVaR with $\beta = 0.9$ is $\mathrm{CVaR}_{0.9} = -29.5\%$, which takes into account all the possible cases in the tail. Therefore, it more accurately reflects the risk exposed to investors.

Another example as shown in the Fig. 2.1, the two slightly different loss distributions with the density functions are plotted in the colors of red and green. Due to the fat tail in the green curve, it will introduce a riskier

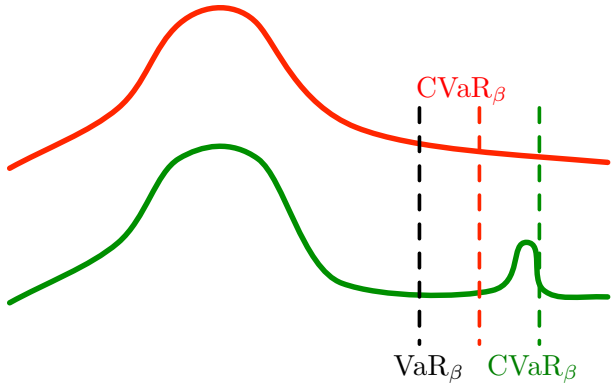

Figure 2.1: Red and Green curves are the density functions of two distributions. The main difference is that the distribution plotted in the green color has a bump in the tail, which means it has a larger possibility for the extreme case to happen for the distribution represented in the green color. For a certain level of confidence β, the two distributions have the same value at risk VaR$_\beta$, however quite difference level of conditional value at risk CVaR$_\beta$.

investment. However, if we use the VaR as our risk measure, there won't be able to pick out the fat tail in the risker investment represented by the green density function. Instead, the two distributions have the same VaR $-\ \beta$. However, CVaR$_\beta$ considers all the possibilities in the extreme case, then picks out the fat tail in the green distribution and values the green one as a risker investment in terms of CVaR$_\beta$, which is in line with our intuitions.

We define the CVaR$_\beta$ of a portfolio w to be the expected losses given the losses do exceed VaR$_\beta$. Mathematically, it is expressed as

$$\text{CVaR}_\beta(w) := \frac{1}{1-\beta} \int_{f(w,y)\geq \text{VaR}_\beta(w)} f(w,y)p(y)\,dy.$$

The pre-scaling factor $\frac{1}{1-\beta}$ is a normalization factor, since the probability $\mathbb{P}(f(w,\, y) \geq \text{VaR}_\beta) = 1 - \beta$. It is noticed that directly using the definition of CVaR$_\beta$ needs to calculate VaR$_\beta$ first. [14] found a way around calculation of VaR$_\beta$ to calculate CVaR$_\beta$ using linear programming.

2.1 Linear Programming

Let us assume that $y \in \mathbb{R}^m$ is the random scenarios vector for n assets. We note the following alternative definition of CVaR in the portfolio optimiza-

tion problem [14]:

$$\mathrm{CVaR}_\beta(w) = \min\left\{t \in \mathbb{R} :\ t + \frac{1}{1-\beta}\int_{\mathbb{R}^m}[f(w,y)-t]^+ p(y)\,dy\right\}, \quad (2.1)$$

If $\mathcal{W}$ is a set of allowable portfolios, our objective is to minimize $\mathrm{CVaR}_\alpha(w)$ over $\mathcal{W}$. It can be shown that we may minimize (2.1) as [14]:

$$\min_{w\in\mathcal{W}} \mathrm{CVaR}_\beta(w) = \min\left\{(w,t) \in \mathcal{W}\times\mathbb{R} :\ t + \frac{1}{1-\beta}\int_{\mathbb{R}^m}[f(w,y)-t]^+ p(y)\,dy\right\},$$

Let $\{y^k\}_{k=1}^N$ be a sampling from the distribution. Then the discretization of the above definition of CVaR_β can be written as

$$\min\left\{(w,t) \in \mathcal{W}\times\mathbb{R} :\ t + \frac{1}{N(1-\beta)}\sum_{k=1}^{N}[f(w,y^k)-t]^+\right\}. \quad (2.2)$$

The optimization problem (2.2) then can be reduced to a linear programing problem. Introducing N auxiliary variables u_k for $k = 1,\ldots,N$, minimizing CVaR_β is equivalent to minimizing the expression

$$t + \frac{1}{N(1-\beta)}\sum_{k=1}^{N} u_k$$

subject to the linear constraints

$$u_k \geq 0 \text{ and } u_k - (f(w,y^k) - t) \geq 0 \text{ for } k = 1,\ldots,N.$$

However, having N number of auxiliary variables u_k is inefficient when N is large ([2], [5]). We are therefore motivated to examine alternative methods for solving the above optimization problem.

2.2 Fast Gradient Descent Method

Iyengar and Ma [5] suggest an alternative definition of CVaR given below:

$$\mathrm{CVaR}_\beta(w) = \max_{\mathbb{Q}\in\mathcal{Q}} \mathbb{E}^{\mathbb{Q}}[f(w,\cdot)] \quad (2.3)$$

where $\mathbb{Q}$ denotes a probability measure on the returns y and the set of measures

$$\mathcal{Q} = \left\{ \mathbb{Q} \,\middle|\, 0 \leq \frac{\partial \mathbb{Q}}{\partial \mathbb{P}} \leq \frac{1}{1-\beta} \right\}.$$

The proof of the equivalence of the alternative definition of the CVaR given by Iyengar [5] is given below.

Proof. We first prove that $\mathbb{E}_y^{\mathbb{Q}}(f(w,y)) \leq \text{CVaR}_\alpha(f(w,y))$ for all $\mathbb{Q} \in \mathcal{Q}_N$ as defined in (2.5). Let Z denote $\frac{\partial \mathbb{Q}}{\partial \mathbb{P}}$. Then, for $\mathbb{Q} \in \mathcal{Q}_N$,

$$\begin{aligned} \mathbb{E}_y^{\mathbb{Q}}(f(w,y)) &= \int_{\mathbb{R}^m} f(w,y) d\mathbb{Q}(y) \\ &= \int_{\mathbb{R}^m} f(w,y) Z(y) d\mathbb{P}(y) = \int_{\mathbb{R}^m} f(w,y) Z(y) p(y) dy \end{aligned}$$

By adding and subtracting $\tau = VaR_\beta(f(w,y))$ in the integrand, we get

$$\begin{aligned} &= \tau + \int_{\mathbb{R}^m} (f(w,y) - \tau) Z p(y) dy \\ &\leq \tau + \int_{\mathbb{R}^m} [f(w,y) - \tau]^+ Z p(y) dy \end{aligned}$$

Since $[f(w,y) - \tau]^+ p(y) \geq 0$, the integral is maximized when Z is at maximum i.e. $Z = \frac{1}{1-\beta}$. Thus,

$$\leq \tau + \frac{1}{1-\beta} \int_{\mathbb{R}^m} [f(w,y) - \tau]^+ p(y) dy,$$

which is given in (2.1). Now we prove $\text{CVaR}_\beta(f(w,y)) \leq \mathbb{E}_y^{\mathbb{Q}}(f(w,y))$ under the same conditions.

$$\begin{aligned} CVaR_\beta(f(w,y)) &= \frac{1}{1-\beta} \mathbb{E}^{\mathbb{P}}[f(w,y) 1_{\{f(w,y) \geq \tau\}}] \\ &= \mathbb{E}_y^{\mathbb{P}}[f(w,y)(\frac{1}{1-\beta} 1_{\{f(w,y) \geq \tau\}})] \end{aligned}$$

which is less than $\max_{\mathbb{Q} \in \mathcal{Q}_N} \mathbb{E}_y^{\mathbb{Q}}(f(w,y))$ because $0 \leq \frac{1}{1-\beta} 1_{\{f(w,y) \geq \tau\}} \leq \frac{1}{1-\beta}$. □

Therefore, our objective function becomes

$$\min_{w \in W} \max_{\mathbb{Q} \in \mathcal{Q}} \mathbb{E}^{\mathbb{Q}}[f(w, \cdot)] \tag{2.4}$$

When the distribution $\mathbb{P}$ is approximated by N scenarios, the set of measures $\mathcal{Q}_N$ is given by

$$\mathcal{Q}_N = \left\{ q \in \mathbb{R}^N \;\middle|\; \sum_{i=1}^{N} q_i = 1, 0 \leq q_i \leq \frac{1}{1-\beta} p_i \right\} \tag{2.5}$$

where $p = (p_1, ..., p_N)^T$. Let $Y = [y^1, \ldots, y^N] \in \mathbb{R}^{n \times N}$ denote the matrix where the i-th column is the return vector for the i-th scenario, $i = 1, ..., N$. Therefore, the scenario-based mean-CVaR problem reduces to the saddle-point problem

$$\min_{w \in W} \max_{q \in \mathcal{Q}_N} \{-q^T Y w\}. \tag{2.6}$$

2.3 Comparison of Methods

The fast gradient descent method has its advantage over the linear programming (LP) method introduced by [14]. In order to convert the convex programming problem into a linear programming problem [14] introduces N auxiliary variables, where N is the number of scenarios. Therefore, as we increase the number of assets in a portfolio, we require more scenarios and auxiliary variables, which will slow the LP-algorithm. The fast gradient descent method proposed by [5], overcomes this problem, which runs very quickly and accurately when the number of assets in the investment pool is small[1]. However, since this algorithm requires solving two quadratic programming problems at each iteration, where the number of variables is equal to the number of securities being considered, its speed is very sensitive to the number of securities. The CPU-time versus the number of assets as well as CPU-time versus the number of scenarios for both algorithms are shown side-by-side in the Section 2.3 for comparison. As noted by Iyengar and Ma, the fast gradient descent method is not exact in the estimated minimal CVaR. Our results indicate that there are considerable inaccuracies when using fast gradient decent on portfolios containing hundreds of assets. Details are discussed in the Section 2.3.

[1] Fast Gradient Descent is very accurate for portfolios containing less than 100 assets.

Convergence

Although the optimization process of CVaR_β is deterministic, which means the same set of scenarios will result in the same optimal allocation and CVaR_β, every time we generate a new random set of scenarios based on the chosen joint density, the Monte Carlo simulation is stochastic. So the first thing to do is to test if the algorithm converges and how fast it converges. The box plots in the Fig. 2.2 and the Fig. 2.3 show the convergence of both algorithms. For each number of scenarios, the Monte-Carlo simulation (MC) is made 50 times. When the number of scenarios is $10,000$, the range of the optimal $\text{VaR}_{0.95}$ and $\text{CVaR}_{0.95}$ are already very small and the number of outliners (marked by the red plus sign) is not significant.

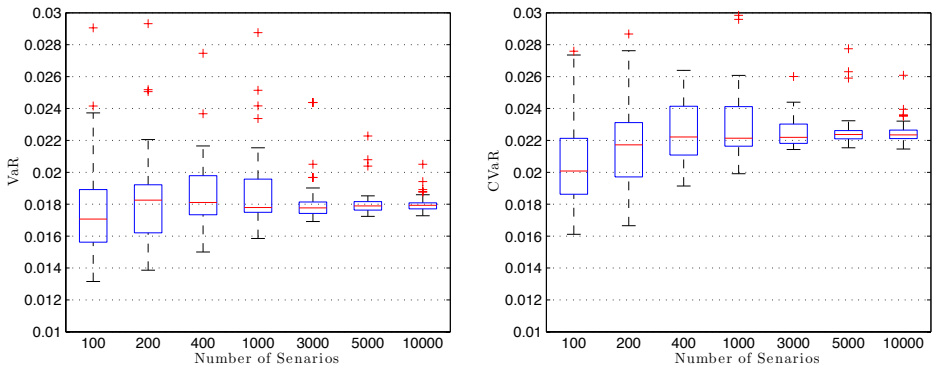

Figure 2.2: The box plot of the optimized VaR (on the left) and CVaR (on the right) by the method introduced in Rockafellar's paper [14] with 50 sets of samples for every number of scenario. The number of scenario for the Monte-Carlo simulation is ranging from 100 up to 10000.

CPU Time versus the Number of Scenarios

We conducted our simulations using a 2.8 GHz Intel Quad-core i7 CPU with 8GB of memory. The optimization toolbox is the academic trial version of MOSEK 6.0 coupled with MATLAB 2010b. We compare the speed of the LP approach with the speed fast gradient descent approach as the number of scenarios increases. We set the number of assets to be 10 and use the historical returns for the 10 biggest companies in a 2001 data set. The number of scenarios ranged from 100 to 10,000 as shown in Fig. 2.4. Both

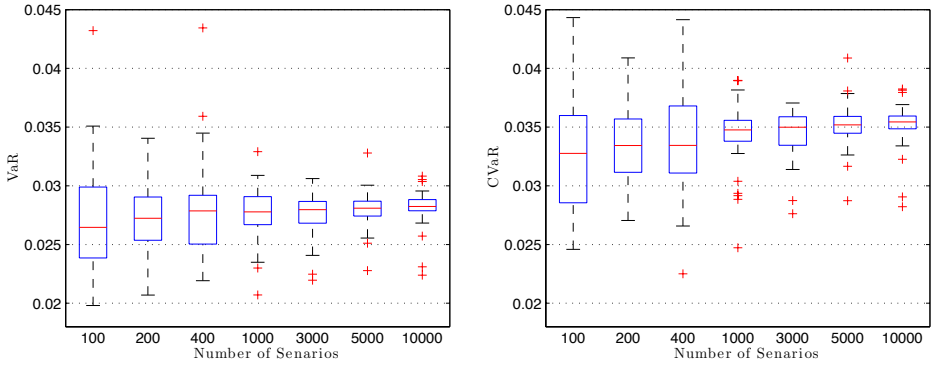

Figure 2.3: The box plot of the optimized VaR (on the left) and CVaR (on the right) by the method introduced in Iyengar's paper [5] with 50 sets of samples for every number of scenario. The number of scenario for the Monte-Carlo simulation is ranging from 100 up to 10000.

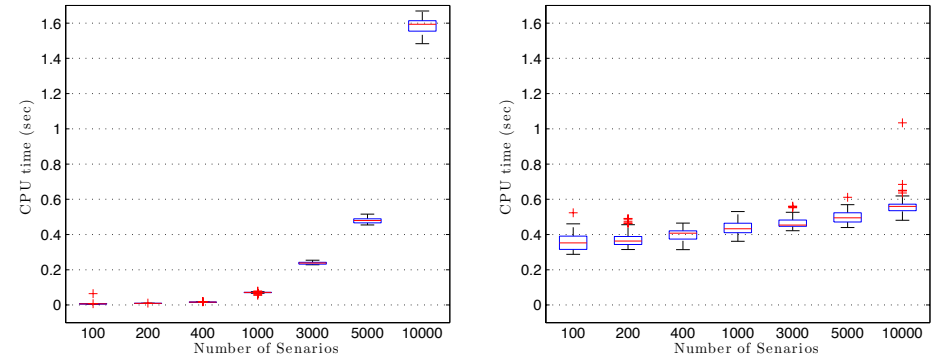

Figure 2.4: The CPU times of Rockafellar's (left) and Iyengar's (right) methods needed for CVaR optimization with 10 assets. The number of scenarios is ranging from 100 up to 10000.

methods were used to optimize over the same constraint set:

$$\mathcal{W} = \left\{ w \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} w_i = 1,\ w_1 \leq w \leq w_2,\ \mathbb{E}^{\mathbb{P}}\left(R\right)^T w = r \right\}.$$

For a given number of scenarios, the simulation is repeated 50 times and depicted in Fig. 2.4. It is quite visible that the LP-approach (left in Fig. 2.4) runs faster than the fast gradient descent method (right in Fig. 2.4) when the number of scenarios is small. When using 5000 scenarios, the LP-approach is still faster, but its CPU time explodes exponentially with the number of scenarios, whereas the CPU time of the fast gradient descent method increases linearly against the number of scenarios at a moderate rate.

CPU Time versus the Number of Assets

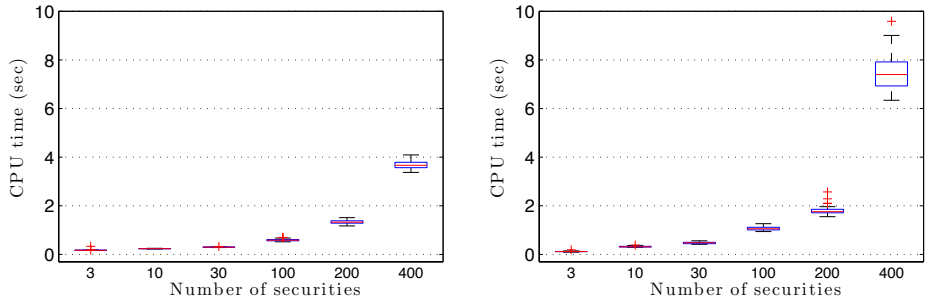

Figure 2.5: The CPU times of Rockafellar's (left) and Iyengar's (right) methods needed for CVaR optimization with 3000 scenarios. The number of assets is ranging from 3 up to 400.

However, as we explained in the Section 2.2, the fast gradient descent method will see an increase in CPU time when we increase the number of assets. An experiment for testing this was setup using the historical data in 2001 where we generated 3000 scenarios for a given number of assets. In Fig. 2.5, the experiment shows that the CPU time using the fast gradient descent method (on the right) grows much faster than that of the LP-approach as the number of assets increases. So for the CVaR optimization problem, on the one hand, a large number of scenarios are needed in order to accurately reproduce the density distribution, which will favor the gradient descent method. On the other hand, a large pool of investment products will make the LP-method a better approach for optimizing CVaR.

Discrepancy of Optimization given by Rockafellar's and Iyengar's

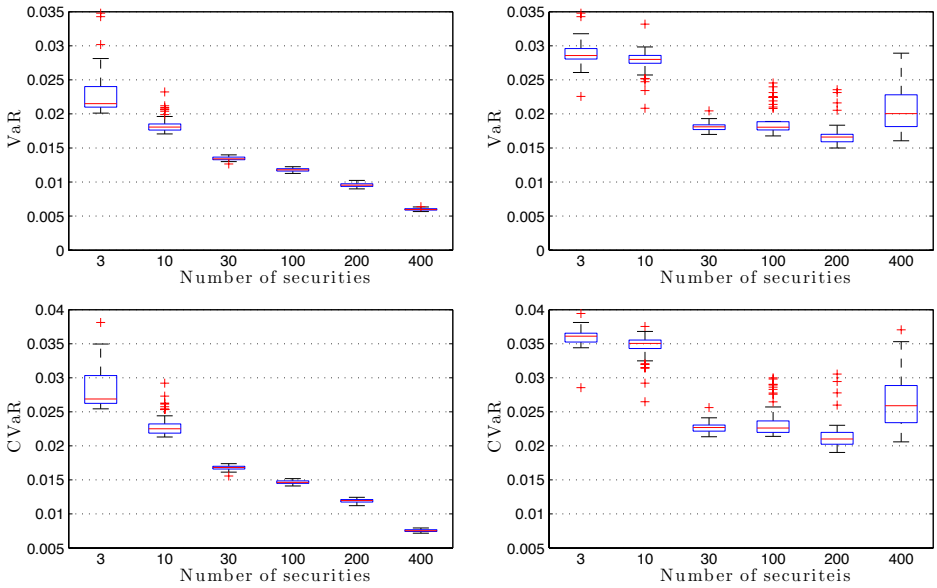

Figure 2.6: Optimized values of VaR (top row) and CVaR (bottom row) given by Rockafellar's (left) and Iyengar's (right) methods with different numbers of assets.

An interesting phenomenon is the discrepancy between the optimal VaR and CVaR estimated in the LP-approach and by the fast gradient descent method. Since the latter one estimates the optimal CVaR with a certain error, it is worth testing how accurate the method is compared with the LP-approach as we increase the number of assets. As depicted in Fig. 2.6, when the number of assets is over 200, the estimated optimal CVaR by the fast gradient descent method is one order of magnitude larger than the one given by the LP-approach. So with the increasing number of assets, not only does the fast gradient descent method get slower, but it is also more prone to large error.

Chapter 3

Fitting Historical Data to Multivariate Distributions

One of the first things needed in portfolio optimization is modelling the probability distribution for the underlying assets. Since the market is non-stationary, we can not track back too much into the history, otherwise the time window of data won't be able to reflect the return and risk in the prevailing market. In our case, we only have 121 days of returns (scenarios), which are not enough to work with the Monte Carlo optimization algorithm. So we have to fit the historical daily returns of 1269 securities to a certain multivariate density distribution with the help of Expectation Maximization (EM) algorithm. Thereafter, with the parameters of the distribution, tens of thousands of faked scenarios are generated to realize CVaR optimization using different optimization algorithms.

In this chapter, we will talk about the several types of multivariate density distributions - normal distribution, student-t distribution, skewed student-t distribution and normal inverse Gaussian distribution - which we will use to fit the historical returns. The EM algorithm and the way to generate the random vectors of number obeying the multivariate density distribution are provided.

3.1 Normal Distribution

The multivariate normal distribution (MVN) is possibly the easiest multivariate distribution to implement, where the parameters needed to be calculated are given by

$$\hat{\mu} := \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i, \qquad \hat{\Sigma} := \frac{1}{n-1}\sum_{i=1}^{n}\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)^{\mathrm{T}}$$

Where, $\mathbf{X}_i$ is the vector of daily returns of a pool of securities; $\bar{\mathbf{X}}$ is the expected returns. There are several problems when modelling stock portfolios in the real world. Since the market is non-stationary, one cannot rely too

heavily on historical data that is too old. However, in order to accurately determine the distribution of a portfolio, one needs a large number of observations. Consequently, the error in the sample covariance matrix can be quite significant [7].

Ledoit and Wolf in 2004 [7] proposed a modification to the sample covariance matrix by using a shrinkage method.

$$\hat{\Sigma}_{\text{shrink}} = \hat{\delta}^* F + \left(1 - \hat{\delta}^*\right) \hat{\Sigma} \tag{3.1}$$

As shown in (3.1), $\hat{\Sigma}$ is the sample covariance matrix for the MVN; F is the so-called shrinkage target; $\hat{\delta}^*$ is the shrinkage intensity. The method assumes all pairwise correlations are identical and averages all the sample correlations as the estimator for the common constant correlation. Together with the estimated variance, the shrinkage target (F) is constructed as

$$f_{ii} = s_{ii} \text{ and } f_{ij} = \bar{r}\sqrt{s_{ii}s_{jj}}$$
$$\bar{r} = \frac{2}{(N-1)\,N} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} r_{ij}, \quad r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}}$$

Shrinkage estimation methods often depend on the invertiblility of a covariance matrix, which need not exist when the number of observations is too low. The estimator proposed by Ledoit and Wolf [7] is defined as the matrix of from (3.1) that minimizes the expected value of the following loss function L:

$$L\left(\delta\right) = ||\delta F + (1-\delta)\, S - \Sigma||^2_{\text{Fro}}\,.$$

Here $\|\cdot\|_{Fro}$ is the Frobenius norm. Asymptotically, the optimal value δ^* behaves as a constant (κ) over the number of securities. Since $\hat{\delta}^*$ is a constant in the region of [0, 1], the optimal estimation is

$$\hat{\delta}^* = \max\left\{0,\, \min\left\{\frac{\hat{\kappa}}{T},\, 1\right\}\right\},$$

where T is the number of observations.

Even with the polishing of the shrinkage method, the MVN still has a long way to go before it can be used in the real-world. In Fig. 3.1, the 126 historical returns of the first security in the 2001 data set is plotted against the sampled data according to the normal distribution with the same mean and variance. The big divergence from the normal distribution is clear in the tails of the distribution. So we need to turn to alternative candidates of the joint density distribution for returns on stock portfolios.

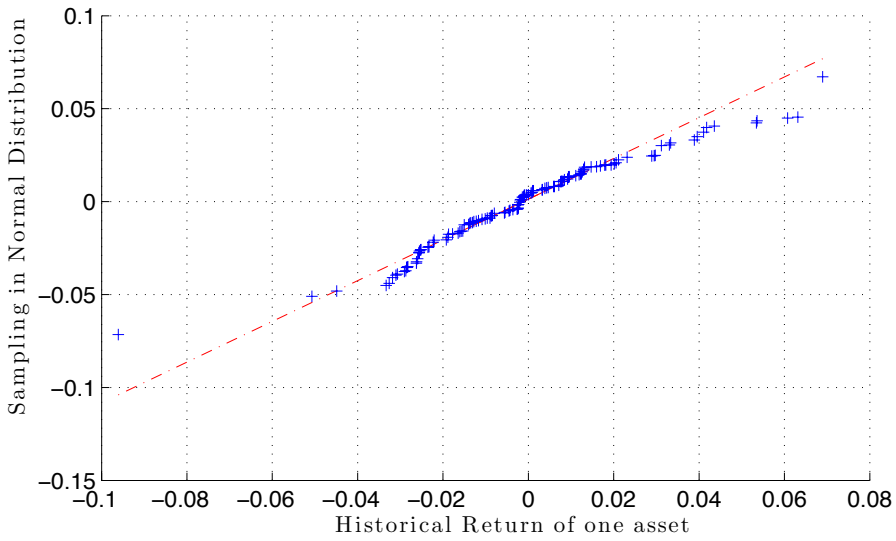

Figure 3.1: QQ-plot of the normal distribution ($\sim \mathrm{N}(\mu, \sigma^2)$, where μ and σ are the mean and standard error of the historical return of the corresponding asset.) against the historical return of one of the assets.

3.2 Student-t Distribution

Joint normal distribution is popular because the covariance matrix (the 2^{nd} moments) is good enough to proxy the multivariate dependence and easy to fit the historical data. However, it is widely acknowledged that the historical data is not normally distributed. As an example shown in the Fig. 3.1, the return of the security with biggest market capital is compared with the sampled normal distribution, the fat tail behaviour is clearly visible. So a more general multivariate distribution with the ability to control the tail effect is required. Here, we look at the multivariate student-t distribution, which has heavy-tails and non-zero tail dependence as stated by [4]. The definition of the multivariate student-t distribution and the EM-algorithm for calibrating the multivariate student-t distribution are given below.

Definition 3.2.1 (Multivariate student-t distribution)**.** Let $\mu \in \mathbb{R}^d$, $\Sigma \in \mathbb{R}^{d\times d}$ a real positive semi-definite matrix, and $\nu > 2$. The d dimensional jointly student-t distributed random vector $\mathbf{X}$, denoted by

$$\mathbf{X} \sim \mathrm{StudentT}_d\left(\nu,\, \mu,\, \Sigma\right),$$

is a multivariate normal mean-variance mixture variable with distribution given by

$$\mathbf{X} \overset{\mathrm{d}}{=} \mu + \sqrt{W}\mathbf{Z}$$

where

1) $\mathbf{Z} \sim \mathbb{N}(\mathbf{0}, \Sigma)$, the multivariate normal distribution with mean of $\mathbf{0}$ and covariance Σ;

2) $W \sim \mathbb{IG}(\nu/2, \nu/2)$, the inverse gamma distribution, and is independent of $\mathbf{Z}$.

Let $\mathbf{X} \in \mathbb{R}^d$ be jointly student-t distributed. Then the joint Probability Density Function (PDF) of $\mathbf{X}$ is denoted as

$$f(\mathbf{x}\,|\nu,\, \mu,\, \Sigma) = \frac{\Gamma\left(\frac{\nu+d}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)(\pi\nu)^{\frac{d}{2}}\,|\Sigma|^{\frac{1}{2}}}\left(1+\frac{\rho(\mathbf{x})}{\nu}\right)^{-\frac{\nu+d}{2}}, \tag{3.2}$$

where $\rho(\mathbf{x}) = (\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)$. The mean and covariance of the student-t distributed random vector $\mathbf{X}$ are

$$\begin{aligned}\mathbb{E}(\mathbf{X}) =& \mu\\ \mathbf{COV}(\mathbf{X}) =& \frac{\nu}{\nu-2}\Sigma.\end{aligned}$$

Algorithm 3.2.2 (EM-algorithm[2] for calibrating the multivariate student-t distribution)**.** Steps are:

1) Set the iteration counter $k = 1$. Select staring values for $\mu^{[1]}$, $\gamma^{[1]}$, $\mu^{[1]}$ and $\Sigma^{[1]}$. Reasonable starting value for mean and dispersion matrix are the sample mean and sample covariance matrix.

2) Calculate $\theta_i^{[k]}$, $\eta_i^{[k]}$ and $\xi_i^{[k]}$ and their average $\bar{\theta}$, $\bar{\eta}$ and $\bar{\xi}$. where

$$\theta_i^{[k]} = \frac{\nu^{[k]}+d}{\rho_i^{[k]}+\nu^{[k]}} \tag{3.3}$$

$$\eta_i^{[k]} = \frac{\rho_i^{[k]}+\nu^{[k]}}{\nu^{[k]}+d-2} \tag{3.4}$$

$$\xi_i^{[k]} = \log\left(\frac{\rho_i^{[k]}+\nu^{[k]}}{2}\right) - \psi\left(\frac{d+\nu^{[k]}}{2}\right) \tag{3.5}$$

$$\psi(x) = \frac{\mathrm{d}\log(\Gamma(x))}{\mathrm{d}x},\ \bar{\theta} = \frac{1}{n}\sum_1^n \theta_i,\ \bar{\eta} = \frac{1}{n}\sum_1^n \eta_i,\ \bar{\xi} = \frac{1}{n}\sum_1^n \xi_i \tag{3.6}$$

[2]EM-algorithm is the abbreviation of Expectation-Maximization algorithm.

3) Update γ, μ and Σ according to

$$\gamma^{[k+1]} = \frac{n^{-1}\sum_{i=1}^{n}\theta_i^{[k]}}{\bar{\theta}^{[k]}\bar{\eta}^{[k]} - 1} \tag{3.7}$$

$$\mu^{[k+1]} = \frac{n^{-1}\sum_{i=1}^{n}\theta_i^{[k]}\mathbf{x}_i - \gamma^{[k+1]}}{\bar{\theta}^{[k]}} \tag{3.8}$$

$$\Sigma^{[k+1]} = \frac{1}{n}\sum_{i=1}^{n}\theta_i^{[k]}\left(\mathbf{x}_i - \mu^{[k+1]}\right)\left(\mathbf{x}_i - \mu^{[k+1]}\right)' - \bar{\eta}^{[k]}\gamma^{[k+1]}\gamma^{[k+1]\prime} \tag{3.9}$$

4) Compute $\nu^{[k+1]}$ by numerically solving the equation

$$-\psi\left(\frac{\nu}{2}\right) + \log\left(\nu/2\right) + 1 - \bar{\xi}^{[k]} - \bar{\theta}^{[k]} = 0 \tag{3.10}$$

5) Set counter $k \to k+1$ and go back to step 2 unless the relative increment of log likelihood is small and in this case, we terminate the iteration.

The result of this algorithm is an estimate of the maximum likelihood parameter values for the given data.

3.3 Skewed Student-t Distribution

Compared with the student-t distribution, the skewed student-t distribution gives one more control on the possible asymmetric on the positive and negative sides of return. It should provide a better fitting to the historical data. However, the calibration algorithm will be less efficient. The definition and the EM-algorithm for calibration, given in the book of Quantitative Risk Management: Concepts, Techniques, and Tools [10], can be found below.

Definition 3.3.1 (Multivariate skewed student-t distribution)**.** Let $\gamma \in \mathbb{R}^d$, $\mu \in \mathbb{R}^d$, $\Sigma \in \mathbb{R}^{d\times d}$ a real positive semi-definite matrix, and $\nu > 2$. The d dimensional jointly skewed student-t distributed random vector $\mathbf{X}$, denoted by

$$\mathbf{X} \sim \text{SkewedT}_d\left(\nu,\, \mu,\, \Sigma,\, \gamma\right),$$

is a multivariate normal mean-variance mixture variable with distribution given by

$$\mathbf{X} \overset{\text{d}}{=} \mu + W\gamma + \sqrt{W}\mathbf{Z}$$

where

1) $\mathbf{Z} \sim \mathbb{N}(\mathbf{0},\, \Sigma)$, the multivariate normal distribution with mean of $\mathbf{0}$ and covariance Σ;

2) $W \sim \mathbb{IG}(\nu/2,\, \nu/2)$, the inverse gamma distribution, and is independent of $\mathbf{Z}$.

Let $\mathbf{X} \in \mathbb{R}^d$ be jointly skewed student-t distributed. Then the joint Probability Density Function (PDF) of $\mathbf{X}$ is denoted as

$$f\left(\mathbf{x}\,|\nu,\, \mu,\, \Sigma,\, \gamma\right) = c\frac{K_{\frac{\nu+d}{2}}\left(\sqrt{\left(\nu+\rho\left(\mathbf{x}\right)\right)\left(\gamma'\Sigma^{-1}\gamma\right)}\right)\exp\left[\left(\mathbf{x}-\mu\right)'\Sigma^{-1}\gamma\right]}{\left(\sqrt{\left(\nu+\rho\left(\mathbf{x}\right)\right)\left(\gamma'\Sigma^{-1}\gamma\right)}\right)^{-\frac{\nu+d}{2}}\left(1+\frac{\rho(\mathbf{x})}{\nu}\right)^{\frac{\nu+d}{2}}} \tag{3.11}$$

$$c = \frac{2^{1-\frac{\nu+d}{2}}}{\Gamma\left(\frac{\nu}{2}\right)(\pi\nu)^{\frac{d}{2}}\left|\Sigma\right|^{\frac{1}{2}}}$$

where $\rho(\mathbf{x}) = (\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)$. The mean and covariance of the student-t distributed random vector $\mathbf{X}$ are

$$\mathbb{E}\left(\mathbf{X}\right) = \mu + \gamma + \frac{\nu}{\nu-2}$$

$$\mathbf{COV}\left(\mathbf{X}\right) = \frac{\nu}{\nu-2}\Sigma + \gamma\gamma'\frac{2\nu^2}{\left(\nu-2\right)^2\left(\nu-4\right)}.$$

Algorithm 3.3.2 (EM-algorithm for calibrating the multivariate skewed t distribution)**.** For the skewed t distribution, it follows exactly the same steps of the EM-algorithm as the one for student-t distribution. It just has a set of more complicated formulas for the auxiliary variables $\theta_i^{[k]}$, $\eta_i^{[k]}$, and $\xi_i^{[k]}$ in the step 2 of the EM-algorithm for student-t distribution. The detailed

formation of them are as below,

$$\theta_i^{[k]} = \left(\frac{\rho_i^{[k]} + \mu^{[k]}}{\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]}} \right)^{-\frac{1}{2}} \frac{K_{\frac{\mu+d+2}{2}} \left(\sqrt{\left(\rho_i^{[k]} + \mu^{[k]} \right) \left(\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]} \right)} \right)}{K_{\frac{\mu+d}{2}} \left(\sqrt{\left(\rho_i^{[k]} + \mu^{[k]} \right) \left(\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]} \right)} \right)} \tag{3.12}$$

$$\eta_i^{[k]} = \left(\frac{\rho_i^{[k]} + \mu^{[k]}}{\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]}} \right)^{\frac{1}{2}} \frac{K_{\frac{\mu+d-2}{2}} \left(\sqrt{\left(\rho_i^{[k]} + \mu^{[k]} \right) \left(\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]} \right)} \right)}{K_{\frac{\mu+d}{2}} \left(\sqrt{\left(\rho_i^{[k]} + \mu^{[k]} \right) \left(\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]} \right)} \right)} \tag{3.13}$$

$$\xi_i^{[k]} = \frac{1}{2} \log \left(\frac{\rho_i^{[k]} + \mu^{[k]}}{\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]}} \right) + \frac{\partial K_{-\frac{\mu+d}{2}+\alpha} \left(\sqrt{\left(\rho_i^{[k]} + \mu^{[k]} \right) \left(\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]} \right)} \right) / \partial \alpha \Big|_{\alpha=0}}{K_{\frac{\mu+d}{2}} \left(\sqrt{\left(\rho_i^{[k]} + \mu^{[k]} \right) \left(\gamma^{[k]\prime} \Sigma^{[k]^{-1}} \gamma^{[k]} \right)} \right)} \tag{3.14}$$

3.4 Normal Inverse Gaussian Distribution

As defined below, normal inverse Gaussian (NIG) distribution is a commonly used distribution. Same as the skewed student-t distribution, It also has the advantage of being able to represent the asymmetry of the historical data. Since the historical data show somewhat skewness toward the positive side in the QQ-plot of the Fig. 3.1, so the optimization result given the NIG distribution is expected to have a better return/risk ratio than the normal distribution, which can be viewed in the Fig. 4.11. The EM-algorithm for calibrating the multivariate NIG distribution [13] is shown below together with the definition.

Definition 3.4.1 (Multivariate normal inverse Gaussian distribution)**.** Let $\alpha > 0$, $\beta \in \mathbb{R}^d$, $\delta > 0$, $\mu \in \mathbb{R}^d$, and $\mathbf{\Gamma} \in \mathbb{R}^{d \times d}$ a real positive semi-definite matrix. The d dimensional jointly normal inverse gaussian distributed random vector $\mathbf{X}$ denoted by,

$$\mathbf{X} \sim \mathrm{MNIG}_d \left(\alpha,\, \beta,\, \delta,\, \mu,\, \mathbf{\Gamma} \right),$$

is a multivariate normal mean-variance mixture variable with distribution given by

$$\mathbf{X} \overset{\mathrm{d}}{=} \mu + Z \mathbf{\Gamma} \beta + \sqrt{Z} \mathbf{\Gamma}^{1/2} \mathbf{Y}$$

where

1) $\mathbf{Y} \sim \mathbb{N}_d(\mathbf{0}, \mathbf{I})$, the multivariate normal distribution with mean of $\mathbf{0}$ and the unit covariance;

2) $Z \sim \mathbb{IG}(\delta^2, \alpha^2 - \beta'\mathbf{\Gamma}\beta)$, $\mathbb{IG}(\chi, \psi)$, $\chi, \psi > 0$ [3], the inverse gamma distribution, and is independent of $\mathbf{Y}$.

3) The PDF of the inverse gaussian distribution has the form

$$f_{\mathbf{Z}}(z) = \left(\frac{\chi}{2\pi z^3}\right)^{\frac{1}{2}} e^{\sqrt{\chi\psi}} \exp\left[-\frac{1}{2}\left(\chi z^{-1} + \psi z\right)\right], \quad z > 0$$

Let $\mathbf{X} \in \mathbb{R}^d$ be jointly normal inverse gaussian distributed. Then the joint Probability Density Function (PDF) of $\mathbf{X}$ is denoted as

$$f_{\mathbf{X}}(\mathbf{x}\,|\alpha, \beta, \delta, \mu, \mathbf{\Gamma}) = \int f_{\mathbf{X}|Z}(\mathbf{x}\,|z)\, f_Z(z)\, \mathrm{d}z \tag{3.15}$$

$$= \frac{\delta}{2^{\frac{d-1}{2}}} \left[\frac{\alpha}{\pi q(\mathbf{x})}\right]^{\frac{d+1}{2}} \exp\left[p(\mathbf{x})\right] K_{\frac{d+1}{2}}\left[\alpha q(\mathbf{x})\right] \tag{3.16}$$

where $p(\mathbf{x}) = \delta\sqrt{\alpha^2 - \beta'\mathbf{\Gamma}\beta} + \beta'(\mathbf{x} - \mu)$ and $q(\mathbf{x}) = \sqrt{\delta^2 + [(\mathbf{x} - \mu)\mathbf{\Gamma}^{-1}(\mathbf{x} - \mu)]}$. The mean and covariance of the student-t distributed random vector $\mathbf{X}$ are

$$\mathbb{E}(\mathbf{X}) = \mu + \frac{\delta\mathbf{\Gamma}\beta}{\sqrt{\alpha^2 - \beta'\mathbf{\Gamma}\beta}}$$

$$\mathbf{COV}(\mathbf{X}) = \delta\left(\alpha^2 - \beta'\mathbf{\Gamma}\beta\right)^{-\frac{1}{2}}\left[\mathbf{\Gamma} + \left(\alpha^2 - \beta'\mathbf{\Gamma}\beta\right)^{-1}\mathbf{\Gamma}\beta\beta'\mathbf{\Gamma}\right].$$

Algorithm 3.4.2 (EM-algorithm for calibrating the multivariate normal inverse Gaussian distribution)**.** The densities belonging to the exponential family. The E-steps calculates the conditional expectations of the sufficient

[3] compared to the definition on Wikipedia of inverse gaussian distribution, which is in terms of μ and λ,

$$\mathbb{IG}(\mu, \lambda) = \mathbb{IG}\left(\sqrt{\frac{\chi}{\psi}}, \chi\right) = \mathbb{IG}\left(\frac{\delta}{\sqrt{\alpha^2 - \beta'\mathbf{\Gamma}\beta}}, \delta^2\right).$$

Then the PDF on Wikipedia is defined as

$$f(x;\, \mu, \lambda) = \left(\frac{\lambda}{2\pi x^3}\right)^{\frac{1}{2}} \exp\left[\frac{-\lambda(x - \mu)^2}{2\mu^2 x}\right]$$

statistics of the unobserved data. Hence, at the E-step one needs to calculate the conditional expectation of the sufficient statistics for the inverse Gaussian distribution. Theses are $\sum_i Z_i$ and $\sum_i Z_i^{-1}$. Then the M-step updates the parameters using the expectations of the sufficient statistics found at the E-step.

1) In the E-step, we need the first-order moment and the inverse first moment of $Z|\,\mathbf{X}$. These are given by,

$$\varsigma_i = \mathbb{E}\left(Z_i\,|\mathbf{X}_i = \mathbf{x}_i\right) = \frac{q\left(\mathbf{x}_i\right)}{\alpha}\frac{K_{(d-1)/2}\left[\alpha q\left(\mathbf{x}_i\right)\right]}{K_{(d+1)/2}\left[\alpha q\left(\mathbf{x}_i\right)\right]} \tag{3.17}$$

$$\varphi_i = \mathbb{E}\left(Z_i^{-1}\,|\mathbf{X}_i = \mathbf{x}_i\right) = \frac{\alpha}{q\left(\mathbf{x}_i\right)}\frac{K_{(d+3)/2}\left[\alpha q\left(\mathbf{x}_i\right)\right]}{K_{(d+1)/2}\left[\alpha q\left(\mathbf{x}_i\right)\right]} \tag{3.18}$$

for $i = 1, \ldots, N$.

2) The M-step updates the parameters using the pseudo values calculated in the E-step. Compute $\zeta = \sum_{i=1}^{N} \varsigma_i / N$ and $\vartheta = N(\sum_{i=1}^{N}(\varphi_i - \zeta^{-1}))^{-1}$. Then update the parameters as follows:

$$\hat{\delta} = \sqrt{\vartheta} \tag{3.19}$$

$$\hat{\beta} = \hat{\mathbf{\Gamma}}^{-1}\left(\frac{\sum_i \mathbf{x}_i\varphi_i - \bar{\mathbf{x}}\sum_i \varphi_i}{n - \zeta\sum_i \varphi_i}\right) \tag{3.20}$$

$$\hat{\mu} = \bar{\mathbf{x}} - \zeta\hat{\mathbf{\Gamma}}\hat{\beta} \tag{3.21}$$

$$\hat{\alpha} = \sqrt{\left(\hat{\delta}/\zeta\right)^2 + \hat{\beta}^{\mathrm{T}}\hat{\mathbf{\Gamma}}\hat{\beta}} \tag{3.22}$$

where $\bar{\mathbf{x}} = \sum_i \mathbf{x}_i / n$. The $\mathbf{\Gamma}$-matrix can be estimated as follows:

$$\hat{\mathbf{\Gamma}} = \frac{\mathbf{R}\left(\mathbf{B} + \mathbf{I}\right)^{-1}}{\left\{\det\left[\mathbf{R}\left(\mathbf{B} + \mathbf{I}\right)^{-1}\right]\right\}^{1/d}} \tag{3.23}$$

where

$$\mathbf{B} = \frac{1}{2}\left[-\mathbf{I} + \mathbf{W}\left(\mathbf{I} + 4\mathbf{D}_{\mathrm{AR}}\right)^{1/2}\mathbf{W}^{-1}\right] \tag{3.24}$$

$$\mathbf{R} = \frac{1}{N}\sum_i \left(\mathbf{x}_i - \hat{\mu}\right)\left(\mathbf{x}_i - \hat{\mu}\right)^{\mathrm{T}}\varphi_i \tag{3.25}$$

$$\mathbf{D}_{\mathrm{AR}} = \mathbf{W}^{-1}\mathbf{A}\mathbf{R}\mathbf{W}, \quad \mathbf{A} = \xi\hat{\beta}\hat{\beta}^{\mathrm{T}} \tag{3.26}$$

where $\mathbf{D}_{\mathrm{AR}}$ is a diagonal matrix of eigenvalues, and $\mathbf{W}$ is a matrix whose columns are the corresponding eigenvectors so that $\mathbf{ARW} = \mathbf{WD}_{\mathrm{AR}}$.

The algorithm is iterated between these two steps until a reasonable convergence criterion has been reached.

3.5 Goodness-of-Fit Test

The goodness-of-fit as a statistical model is often used in statistical hypothesis testing to identify if outcome frequencies follow a specific distributions. The following tests and their underlying measure of fits can be used: Anderson-Darling test [3], Pearson χ^2 test [12] and Kolmogorov-Smirnov test [9] and Cramer-von Mises group [6]. However, most of the goodness-of-fit tests have been developed for univariate distributions, except for the case of multivariate normality. The problem is the probability distribution of these multivariate statistics are not distribution-free as in the univariate case [6]. Even more, Not only do we want to test the goodness-of-fit to a single multivariate distribution, but also want to compare the goodness-of-fit against each other among several multivariate distributions. In Justel's paper [6], an algorithm was developed for calculation of the bivariate case of goodness-of-fit. The computation of it of a portfolio with such a large number of variables (1269 assets) itself is a problem. Therefore, as seen in the chapter 4, we adopt the fast and straightforward ex-post test of portfolio returns and efficient frontier to identify the preferability in-between these multivariate distribution applied in our work.

Chapter 4

Computation Results

4.1 Portfolio Performance

In this chapter, we are going to show the computation results of ex-post test with our 121 days of daily returns of a pool of 1269 securities in 2001. In Chapter 2, we talk about the VaR and CVaR as another two risk metrics besides the variance. The objective of CVaR optimization problem is to minimize the value of CVaR with constraints on the expected return of the portfolio and maximum allocation rate for each of the securities in the pool.[4] The ex-post test is carried out with both Rockafellar's [14] and Iyengar's [5] algorithms for the normal distribution. For other distributions, smaller number of securities is used due to the technical problem explained later and the results are compared among different distributions. The efficient frontiers with different distribution are also given.

4.1.1 Constraint w/ $\sum w_i = 1$

The CVaR is optimized for the portfolio with all the 1269 assets available in the data set of 2001 with the constraint

$$\mathcal{W} = \left\{ w \in \mathbb{R}^n \middle| \sum_{i=1}^{n} w_i = 1,\ -0.05 \leq w \leq 0.05,\ \mathbb{E}[-f(w,\cdot)] \geq 2\bar{r} \right\}. \quad (4.1)$$

where $\bar{r}$ represents the average return for the portfolio of assets. The optimal allocations shown in Fig. 4.1 were solved using Rockafellar's linear programming method **??**, and in Fig. 4.2 solved using Iyengar's fast gradient descent method [5]. Both of them assume that the historical data obey the multivariate normal distribution. Because of the long-short constraint, both algorithms allocated most assets towards the extreme of allocation limits in $\mathcal{W}$. Because of the reason explained in the Section 2.3, let's focus on the result given by the LP-approach. Assets with positive returns had long optimal allocation, while assets with negative returns had short optimal allocations.

[4] The constraints may include the limitation on allocation rate for both short and long.

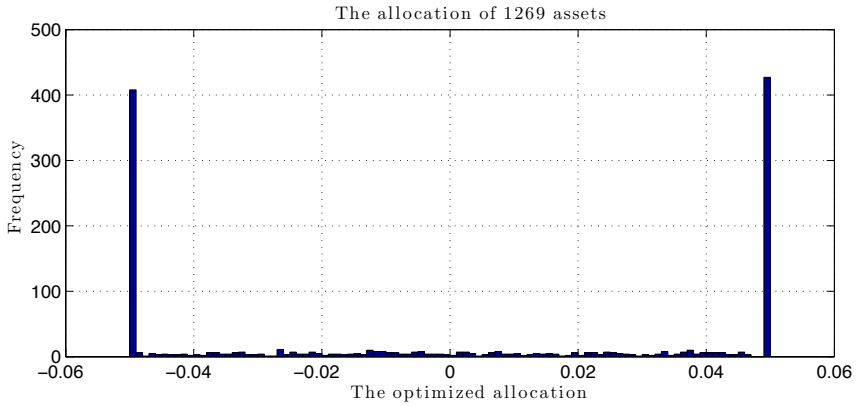

Figure 4.1: The histogram of the optimal allocation of the assets calculated with Rockafellar's method with the constraints of $\mathcal{W} = \{w \in \mathbb{R}^n | \sum_{i=1}^n w_i = 1, -0.05 \leq w \leq 0.05, \mathbb{E}[-f(w,\cdot)] \geq 2\bar{r}\}$. The x-axis shows the allocation rate for all of the securities. The total allocation is 1. The negative allocation represents short-selling. The y-axis is the number of securities whose allocations fall into the corresponding bin. 20000 scenarios of the return are generated with the estimated multivariate normal distribution.

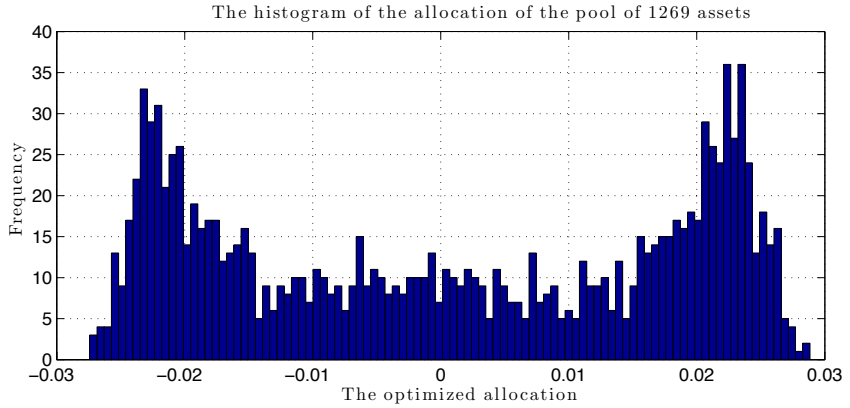

Figure 4.2: The histogram of the optimal allocation of the assets calculated with Iyengar's method with the constraints of $\mathcal{W} = \{w \in \mathbb{R}^n | \sum_{i=1}^n w_i = 1, -0.05 \leq w \leq 0.05, \mathbb{E}[-f(w,\cdot)] \geq 2\bar{r}\}$. The x-axis shows the allocation rate for all of the securities. The total allocation is 1. The negative allocation represents short-selling. The y-axis is the number of securities whose allocations fall into the corresponding bin. 20000 scenarios of the return are generated with the estimated multivariate normal distribution.

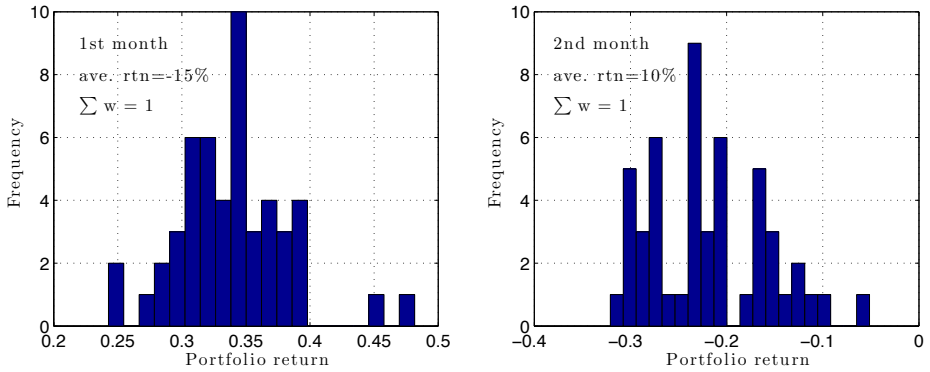

Figure 4.3: The distribution of the ex-post performance of the optimal portfolio in terms of CVaR in the first (left) and the second (right) future months with the constraint $\mathcal{W} = \{w \in \mathbb{R}^n | \sum_{i=1}^n w_i = 1, -0.05 \leq w \leq 0.05, \mathbb{E}[-f(w, \cdot)] \geq 2\bar{r}\}$. The x-axis is the monthly rate of return. The y-axis shows the number of times the rate of return falls in the corresponding bin. The distributions of the future return are obtained by running the steps of sampling and optimization 50 times with the number of scenarios equal to 20000. The horizontal axis is the percentage monthly return. The vertical axis is the number of times a stock with that monthly return falls into the corresponding bin.

As depicted in Fig. 4.3, in the first month following optimization, when the average return of the market is approximately -15%, our optimal portfolio has an average return of about 34% with a minimum of 24% and maximum of 48%. Even though the return of our optimal portfolio is much better than the market average, this portfolio is highly impractical. Since the constraint set for this portfolio just requires the net allocation to be one without adding any restrictions on the margin requirements for short positions. The optimal allocation was close to 2700% for long positions and -2600% for short positions. In the real world, where we have transaction costs and other limitations, high allocation rates will increase the cost of management and degrade the performance of the portfolio.

In the second following optimization month, if we follow the strategy of buy-and-hold, the market turns around with an average return of close to 10%. However, our portfolio performs very poorly with a loss ranging from 6% to 31% and an average of approximately -22%. This is a great example of the non-stationary market and shows the importance of dynamically managing ones portfolio, especially in a volatile market.

In the Fig. 4.4 and the Fig. 4.5, the CVaR optimization results with assuming that the historical data obey different multivariate distributions - the normal, the student-t and the NIG - are compared side by side. Due to the technical problem of singularity of matrices when calibrating the statistical models with a large number of securities, only the 80 biggest securities (1269 securities in total) in terms of the market value are used. The CVaR minimization is carried out with the Rockafellar's linear programming algorithm [14] using the trial version of the optimization toolbox for MATLAB released by MOSEK [1]. The constraints applied are,

$$\mathcal{W} = \left\{ w \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} w_i = 1,\ -0.1 < w < 0.1,\ \mathbb{E}[-f(w,\cdot)] \geq \max(2\bar{r},\ 1.1r^+) \right\}$$

where the total number of assets is $n = 80$; $w_i (i = 1, 2, \cdots, 80)$ are the individual allocation percentage while the total allocation is 1, which means the total assets; $f(w,\cdot)$ is the loss function for the portfolio of 80 assets. The absolute value[5] of the allocation for each asset is not allowed to be bigger than 10% of the total asset and the expected return of the portfolio is required to be bigger than twice of the average return of all the 80 assets in the portfolio and the 1.1 times of the average return of all the assets with positive returns. Since the optimization algorithm is basically a Monte

[5]Positive allocation means buy, while negative one means short.

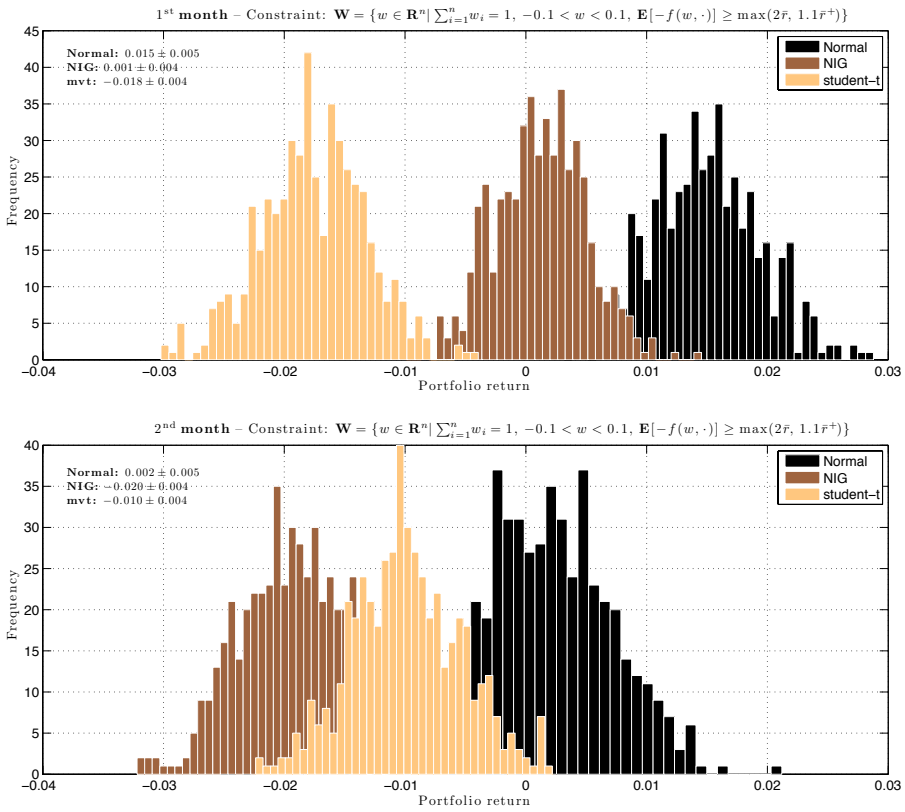

Figure 4.4: The distribution of the ex-post returns of the optimal portfolios of 80 biggest securities in 2001 fitted with three different multivariate distributions of normal, NIG and student-t in the 1st and the 2nd-month for the constraint of $\mathbf{W} = \{w \in \mathbf{R}^n | \sum_{i=1}^n w_i = 1, -0.1 < w < 0.1, \mathbf{E}[-f(w,\cdot)] \geq \max(2\bar{r}, 1.1\bar{r}^+)\}$. For each distribution, the optimal portfolio was found 500 times by MC method. The top figure shows the returns in the 1st future month, and the bottom one shows the returns in the 2nd future month.

Carlo simulation, which is stochastic, the optimization is executed 500 times for each distribution to find a distribution of the optimal returns. In the Fig. 4.4 and the Fig. 4.5, results with respect to normal, student-t and NIG distributions are represented in the color of black, brown and khaki, respectively.

For this constraint, the optimal returns are statistically distinguishable. Ranked in the descending order, it is normal, NIG and student-t. The 2^{nd} month returns are shuffled, which are not quite related to the ones in the 1^{st} month. Again, it shows the instability of the market.

In the Fig. 4.5, the average allocation percentages among the 500 runs of the optimization procedure for three distributions are consistent. Their long positions for the whole portfolio are 2.14 ± 0.029, 2.36 ± 0.028 and 2.43 ± 0.035 for the normal, NIG and student-t distribution, respectively.

4.1.2 Constraint w/ $\sum w_i = 0$

With the same historical data and a different constraint set

$$\mathcal{W} = \left\{ w \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} w_i^+ = 1,\ \sum_{i=1}^{n} w_i^- = -1,\ -0.05 \le w \le 0.05,\ \mathbb{E}[-f(w,\cdot)] \ge 2\bar{r} \right\}, \tag{4.2}$$

where w_i^+ and w_i^- denote the positive and negative allocation for the i-th asset respectively. The optimal allocation of our assets given by the LP-approach and the fast gradient descent method are depicted in Fig. 4.6 and Fig. 4.7, respectively. Due to the size of the problem and our observations, we only consider results given by the LP method.

The optimal allocation of our securities is not at the extremes because of the margin requirement on the short position and requirement on the expectation of the return. The majority of the securities have zero allocation, which are removed from the histogram in Fig. 4.6. Securities which have little contribution to decreasing the portfolio risk and have low absolute value of return are eliminated from consideration as part of the optimal portfolio.

The result of the ex-post test of the performance of the optimal allocation for the first and the second months are shown in the Fig. 4.8. In the first month following optimization, when the market return is -15%, our optimal portfolio has a return ranging from 0.2% to 3.5% and with a mean of about 2%. As we have seen in the previous constraint set, the market turns around in the second month with an average return of 10%. This time our portfolio has a loss ranging from 0.8% to 5.8% and a mean of about -3.5%.

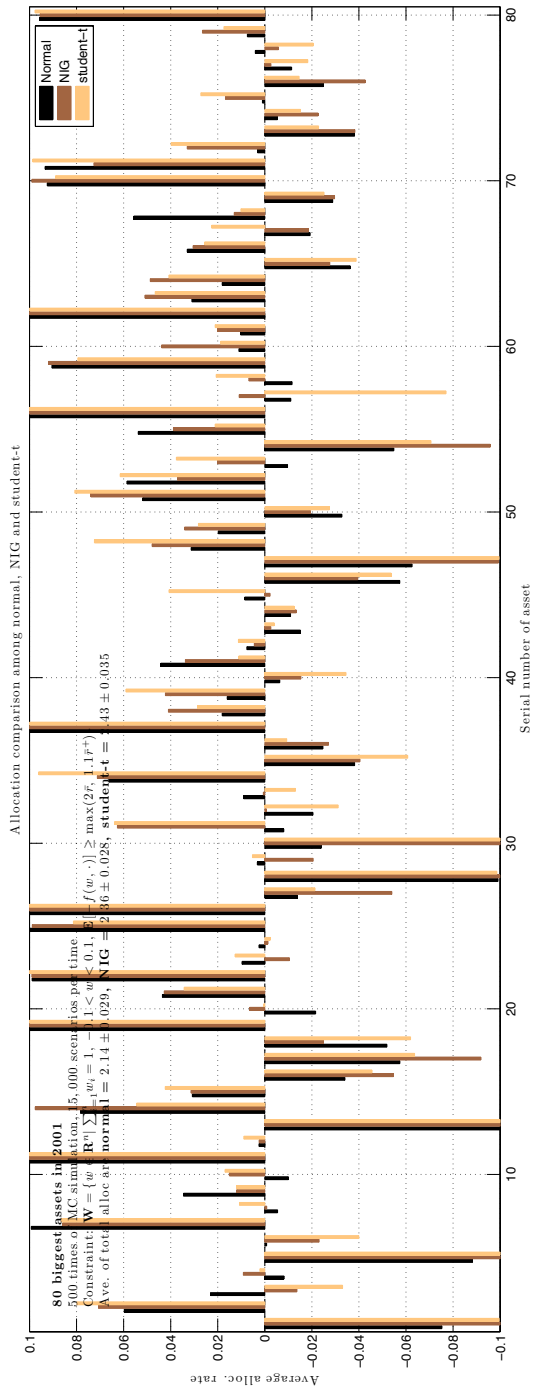

Figure 4.5: The averaged optimal allocations for the portfolio consisting of 80 biggest securities in 2001 fitted with three different multivariate distributions of normal, NIG and student-t, respectively. The constraint for optimization is 2^{nd}-month for the constraint of $\mathbf{W} = \{w \in \mathbf{R}^n | \sum_{i=1}^n w_i = 1, -0.1 < w < 0.1, \mathbf{E}[-f(w,\cdot)] \geq \max(2\bar{r}, 1.1\bar{r}^+)\}$. The average total allocations are printed at the top-left corners in the ascending order of normal, NIG and student-t.

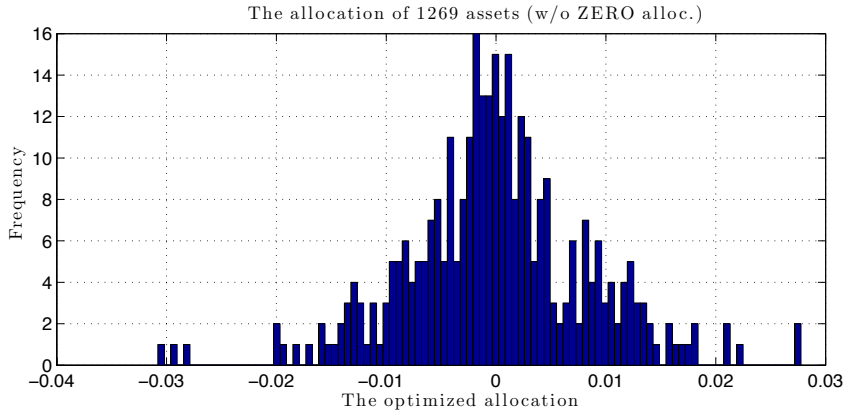

Figure 4.6: The optimized allocation of the assets calculated with Rockafellar's method with the constraints of $\mathcal{W} = \{w \in \mathbb{R}^n | \sum_{i=1}^n w_i^+ = 1, \sum_{i=1}^n w_i^- = -1, -0.05 \leq w \leq 0.05, \mathbb{E}[-f(w,\cdot)] \geq 2\bar{r}\}$. The x-axis shows the allocation rate for all of the securities. The total allocation is 1. The negative allocation represents short-selling. The y-axis is the number of securities whose allocations fall into the corresponding bin. We generated 20000 scenarios using the estimated MVN. Because of the large number of securities having nearly zero allocation, the securities with zero allocation are not shown in this figure.

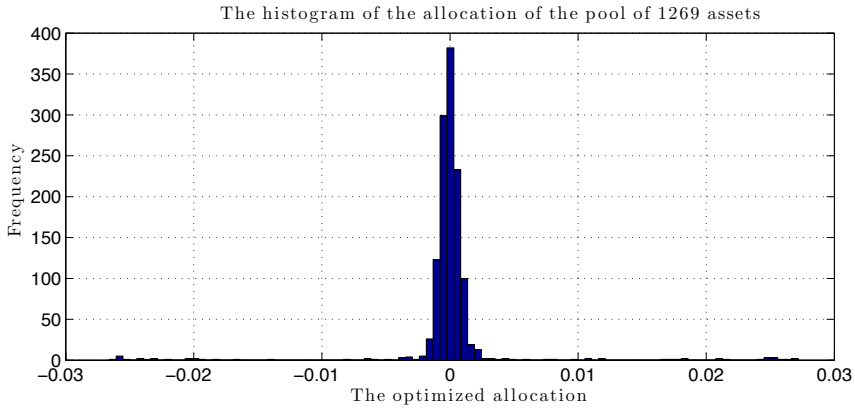

Figure 4.7: The optimized allocation of the assets calculated with Iyengar's method with the constraints of $\mathcal{W} = \{w \in \mathbb{R}^n | \sum_{i=1}^n w_i^+ = 1, \sum_{i=1}^n w_i^- = -1, -0.05 \leq w \leq 0.05, \mathbb{E}[-f(w,\cdot)] \geq 2\bar{r}\}$. The x-axis shows the allocation rate for all of the securities. The total allocation is 1. The negative allocation represents short-selling. The y-axis is the number of securities whose allocations fall into the corresponding bin. We generated 20000 scenarios using the estimated MVN.

In the Fig. 4.9 and the Fig. 4.10, same as in the Section 4.1.1, the CVaR optimization results with assuming that the historical data obey different multivariate distributions - the normal, the student-t and the NIG - are compared side by side. The constraints applied are,

$$\mathcal{W} = \left\{ w \in \mathbb{R}^n \left| \sum_{i=1}^n w_i^+ = 1, \sum_{i=1}^n w_i^- = -1, -0.1 < w < 0.1, \mathbb{E}[-f(w,\cdot)] \geq \max\left(2\bar{r}, 1.1\bar{r}^+\right) \right. \right\}$$

where the total number of assets is $n = 80$; $w_i (i = 1, 2, \cdots, 80)$ are the individual allocation percentage while the total allocation is 1, which means the total assets; $f(w,\cdot)$ is the loss function for the portfolio of 80 assets. The absolute value of the allocation for each asset is not allowed to be bigger than 10% of the total asset and the expected return of the portfolio is required to be bigger than twice of the average return of all the 80 assets in the portfolio and the 1.1 times of the average return of all the assets with positive returns. Since the optimization algorithm is basically a Monte Carlo simulation, which is stochastic, the optimization is executed 500 times for each distribution to find a distribution of the optimal returns. In the Fig. 4.9 and the Fig. 4.10, results with respect to normal, student-t and

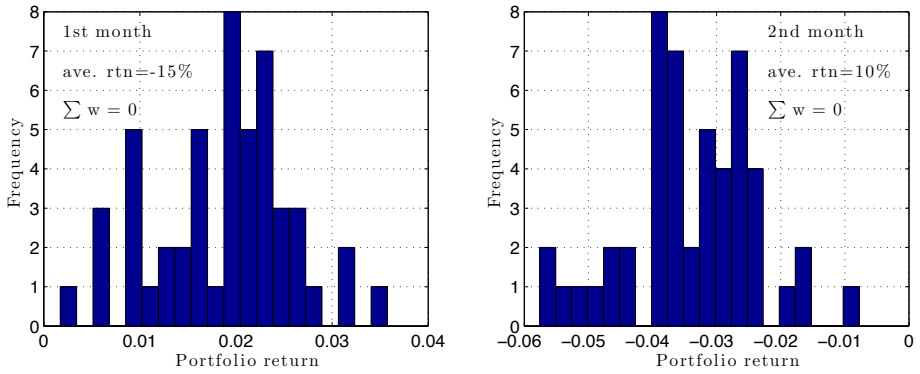

Figure 4.8: The distribution of the ex-post performance of the optimal portfolio in terms of CVaR in the first (left) and the second (right) future months with the constraint $\mathcal{W} = \{w \in \mathbb{R}^n | \sum_{i=1}^n w_i^+ = 1, \sum_{i=1}^n w_i^- = -1, -0.05 \leq w \leq 0.05, \mathbb{E}[-f(w,\cdot)] \geq 2\bar{r}\}$. The x-axis is the monthly rate of return. The y-axis shows the number of times the rate of return falls in the corresponding bin. The distributions of the future return are obtained by running the steps of sampling and optimization 50 times with the number of scenarios equal to 20000. The horizontal axis is the percentage monthly return. The vertical axis is the number of times that the return of the month falls into the corresponding bin.

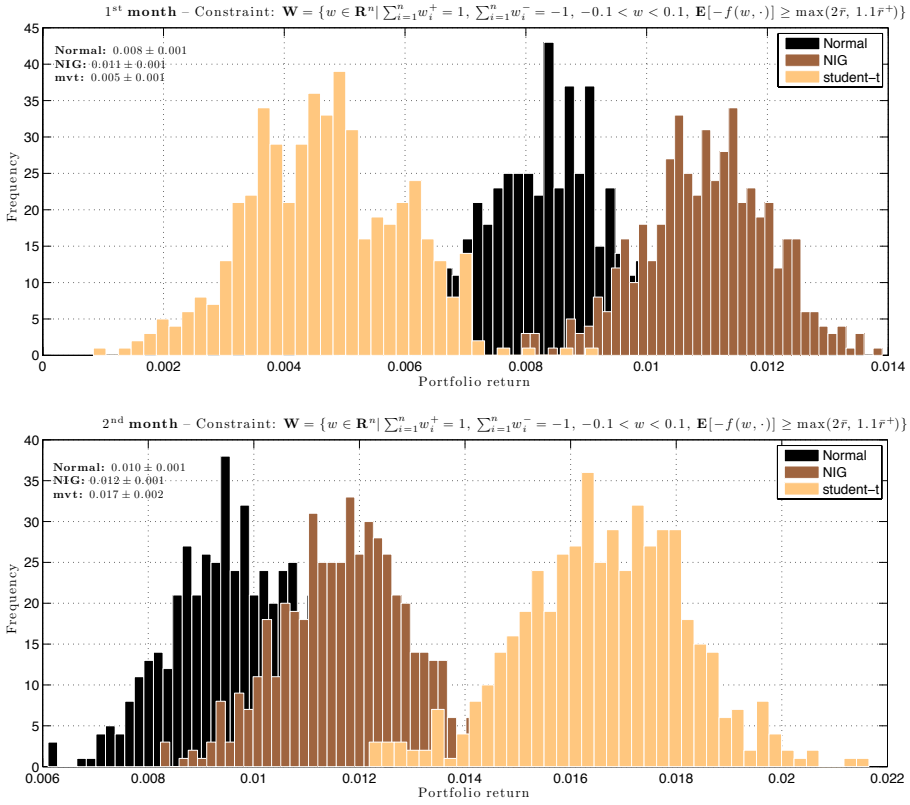

Figure 4.9: The distribution of the ex-post returns of the optimal portfolios of 80 biggest securities in 2001 fitted with three different multivariate distributions of normal, NIG and student-t in the 1st and the 2nd-month for the constraint of $\mathbf{W} = \{w \in \mathbf{R}^n | \sum_{i=1}^n w_i^+ = 1, \sum_{i=1}^n w_i^- = -1, -0.1 < w < 0.1, \mathbf{E}[-f(w,\cdot)] \geq \max(2\bar{r}, 1.1\bar{r}^+)\}$. For each distribution, the optimal portfolio was found 500 times by MC method. The top figure shows the returns in the 1st future month, and the bottom one shows the returns in the 2nd future month.

NIG distributions are represented in the color of black, brown and khaki, respectively. This order of optimal returns - NIG, normal and student-t distributions, in the descending order of returns - is in consistent with what we would expect based on the efficient frontiers given by different density distribution fittings, which we will see in the section 4.2. However in the second month after rebalancing the portfolio based on the CVaR minimization, the ranking of the density distributions is shuffled and irrelevant to the optimal allocation given by the CVaR minimization with using historical data. Again, it shows the instability of the market and supports the necessity of using a narrow window of historical data in the near past and expanding the number scenarios by fitting the limited number of scenarios to a multivariate density distributions.

In the Fig. 4.10, the average allocation percentages among the 500 runs of the optimization procedure for three distributions are consistent. Their long positions for the whole portfolio are 0.438 ± 0.010, 0.578 ± 0.013 and 0.655 ± 0.013 for the normal, NIG and student-t distribution, respectively. For each distribution, the allocation rate is more than 40%, which is efficient. Not as in the previous constraints, the relative small percentage of allocation is practical and decreases the transaction fee induced in real life.

4.2 Portfolio Efficient Frontiers

An interesting and very intuitive plot that we can make is the efficient frontier for our portfolio. With a certain level of total asset, without holding any risk-free asset, any combination of the risky assets can be dotted on the return-risk space, where the vertical axis is the return of the holding portfolio, while the horizontal axis is a kind of risk metric. A collection of all such possible portfolios defines a region in the return-risk space, whose upper edge of the left boundary is the efficient frontier in the absence of a risk-free asset. In Markowitz's work [8], the risk is represented by the standard deviation assuming the return obeys the normal distribution. Fig. 4.11 herein uses the CVaR as the risk metric and shows the maximum return at a certain level of risk (CVaR in our case), or in other words, the minimum risk at a certain level of return for the portfolio. Our investment pool was composed of the 50 largest securities (in terms of market capitalization) from our 2001 data set. The constraint set is

$$\mathcal{W} = \left\{ w \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} w_i = 1,\ 0 \le w \le 0.1,\ \mathbb{E}[-f(w,\cdot)] \ge 2\bar{r} \right\}, \tag{4.3}$$

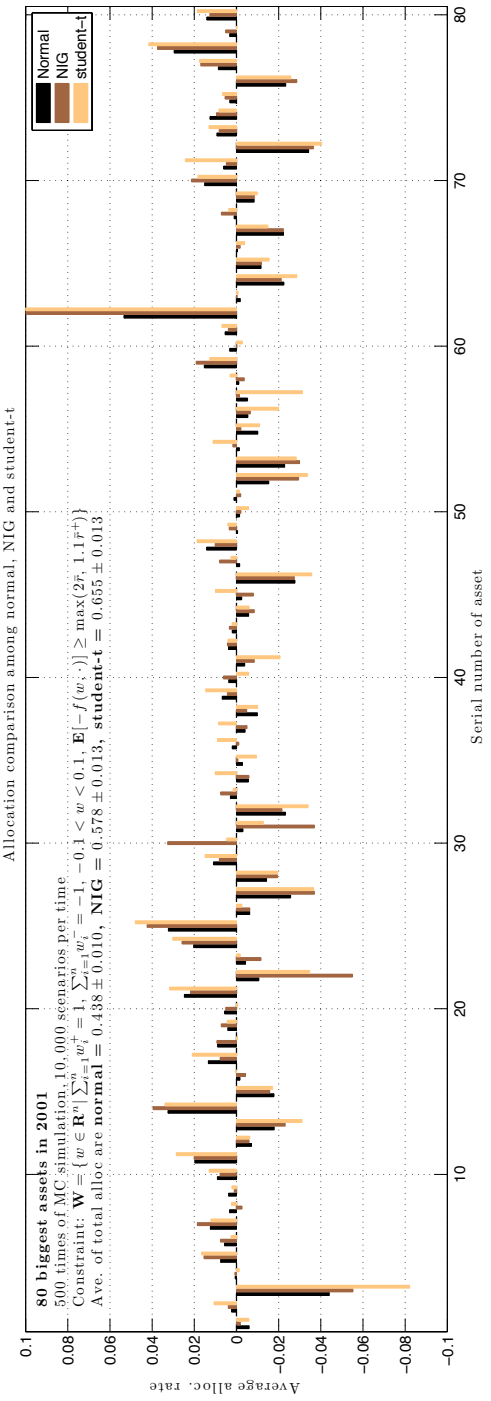

Figure 4.10: The averaged optimal allocations for the portfolio consisting of 80 biggest securities in 2001 fitted with three different multivariate distributions of normal, NIG and student-t, respectively. The constraint for optimization is 2^{nd}-month for the constraint of $\mathbf{W} = \{w \in \mathbf{R}^n | \sum_{i=1}^n w_i^+ = 1, \sum_{i=1}^n w_i^- = -1, -0.1 < w < 0.1, \mathbf{E}[-f(w,\cdot)] \geq \max(2\bar{r}, 1.1\bar{r}^+)\}$. The average total allocations are printed at the top-left corners in the ascending order of normal, NIG and student-t.

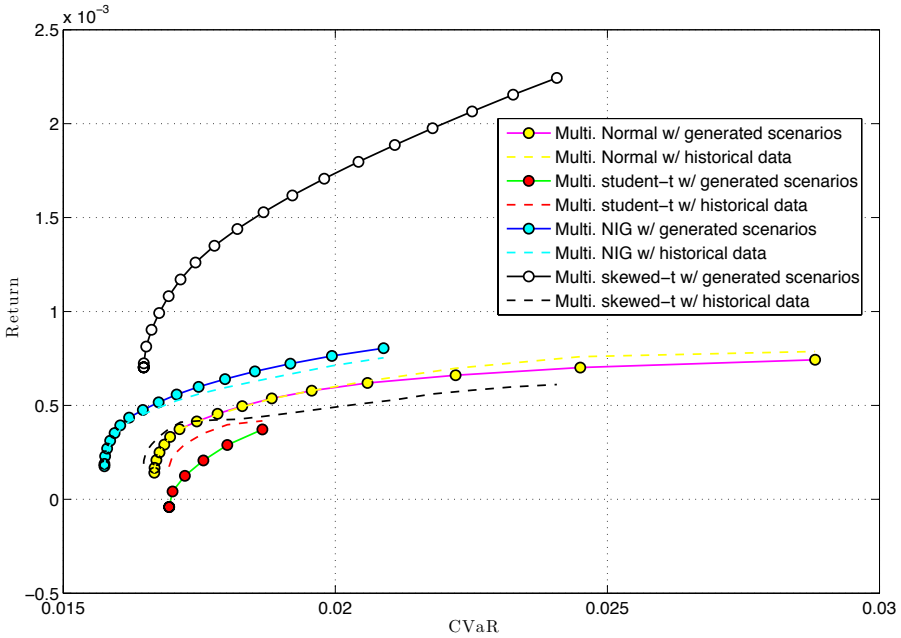

Figure 4.11: The portfolio efficient frontier for the 50 biggest securities in the data set of 2001 in terms of the market capital. The units on the axes are given as rates of return. The optimal allocations for different expected returns are estimated with 5000 scenarios each by the LP-approach. The constraint is $\mathcal{W} = \{w \in \mathbb{R}^n | \sum_{i=1}^{n} w_i = 1,\ 0 \leq w \leq 0.1,\ \mathbb{E}[-f(w,\cdot)] \geq 2\bar{r}\}$.

where short positions are forbidden. The portfolio efficient frontier is plotted by changing the value of expected return r listed as one of the constraints and solve for the minimum values of CVaR. We used the multivariate normal, student-t, and normal inverse gaussian (NIG) distributions during our test. Since the allocation for each security is not allowed to be greater than 10% of the total value of the portfolio, the efficient frontier is plotted by changing the value of r from the average of the 10 securities with the lowest returns to that of the 10 securities with the highest returns. As we see in the Fig. 3.1, historical returns show a characteristic of having fat tails, which represents a higher risk than the one estimated by the normal distribution. Therefore, using other symmetric multivariate distributions with fatter tails (e.g. multivariate t-distribution) is expected to give the efficient frontier below the one given by the multivariate normal distribution, which under-

estimates the risks exposing to the investor. As shown in the Fig. 4.11, the student-t distribution does give an efficient frontier below the one given by the normal. However, even though the NIG distribution has a fatter tail, it gives an efficient frontier above the one given by the normal distribution, which means, with the same amount of expected return, NIG distribution gives smaller amount of risk. It is true with both scenarios generated with respective joint distribution model (solid lines marked with filled circle in the Fig. 4.11) and the historical data (dashed lines). It is because the returns of securities are skewed toward the positive side[6], in other words, fat tail only on the positive side. Unlike the normal and student-t distributions, NIG distribution is able to fit asymmetric data, so a better fit on the positive side of the fat tail given by the NIG distribution should give a higher return than the one given by the normal distribution for a same certain level of risk. As for the skewed student-t distribution, the efficient frontiers, plotted by applying the optimal weight to the generated scenarios and the historical data respectively, have a huge discrepancy, which shows the possible flaws in the EM algorithm for the distribution model fitting.

[6]It makes sense. Otherwise what is the point of running a business, loosing money?

Chapter 5

Conclusion

In the CVaR optimization problem, we studied the LP-approach proposed by Rockafellar [14] and the fast gradient descent method brought forward by Iyengar [5]. We studied the algorithm performance of these two methods by testing their speed in terms of CPU time for both the cases of different number of scenarios and different number securities. We believe that Rockafellar's algorithm [14] will outperform Iyengar's one [5] in the reality, because of its accuracy and good performance with large number of securities, especially with the assist of the fast commercial optimization solver MOSEK.

As for the multivariate joint density distribution, we fit the data with four different types of multivariate joint density distribution: normal distribution, student-t distribution, skewed student-t distribution and normal inverse Gaussian distribution. we improved the estimated covariance matrix of multivariate normal distribution with using the method of shrinkage [7], which restrains the large errors which happens when the number of securities is larger than the number of observations.[7] For the other three distributions, when the number of securities is too large, we will face the problem of singularity when try to estimate the distribution parameters with using the EM algorithm. Therefore, a1 number of smaller-than-observation securities is used and the CVaR optimization is carried out using the 80 biggest securities in terms of market value.

The ex-post tests were carried out for testing the performance of our optimal allocations of the assets. For the multivariate normal distribution, with both constraints we applied, the 1^{st} future month performance is outstanding and were outperforming the market by a margin of 49% and 17% on average, respectively. However, the poor performance of our portfolio shows the non-stationary market and the importance to dynamically managing the portfolio. All of our portfolio performance tests were done with MVN distribution, which we believed underestimated the risk because of the fat tail shown by the QQ-plot in the Fig. 3.1 in the historical data

[7] In our case, we are considering a pool of 1269 assets with 121 observations

when compared with the multivariate normal distribution. So we expected a more conservative and better performance of our portfolios in terms of risk control with using other multivariate joint density distributions. Therefore, the efficient frontiers with 50 biggest securities in terms of market value are plotted for four different density distributions in the Fig. 4.11, where CVaR is adopted as the risk measure. The NIG distribution is spotted to have a better performance in terms of return per unit risk. It means the NIG distribution should have a best fitting to the historical data among these four distributions.

In the future, it is worth to spend time in circumventing the problem of singularity of matrices in EM algorithms when dealing with large number of securities. It will make the optimization method more practical when adopting the density distribution other than the normal one. The power law or mixed density distributions are another thing to try for a better fitting to the historical data.

Appendix A

Codes

A.1 Rockafellar $\sum = 1$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CVaR optimization with LP algorithm;
% Rockafellar: Optimization of Conditional Value-at-Risk(2000)
%
% intrinsic constraint is the total_weight == 1
%
% alpha     : optimal value of VaR
% x         : optimal allocation (1 by N, N is the # of assets)
% obj       : optimal value of CVaR
% LPtime    : CPU time of linear programming in seconds
% flag      : indicator for linprog function
%
% y         : scenarios of returns (q by N, qi is the # of scenarios)
% beta      : confidence level
% R         : expected minimum return in the constraint set
% A_g, b_g  : A_g*x >= b_g (inequality constraints)
% Aeq_g, beq_g: Aeq_g*x == beq_g (equality constraints)
% lb_g_x    : extra lower bound (constraint)
% ub_g_x    : extra upper bound (constraint)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [alpha, x, obj, LPtime, flag] = cvar_rock_lbub(y,beta,R,A_g,b_g,...
        Aeq_g,beq_g,lb_g_x,ub_g_x)
    %% init.
    [q,N] = size(y);

    if nargin~=3 && nargin~=5 && nargin~=7 && nargin~=9
        fprintf( Wrong number of arguments! );
        return;
    end

    if nargin<=7
        lb_g_x = -inf*ones(N,1);
        ub_g_x =  inf*ones(N,1);
        if nargin<=5
            Aeq_g = [];
            beq_g = [];
            if nargin==3
                A_g = [];
                b_g = [];
            end
        end
    end
```

```
43
44      if ((~isempty(A_g))&&(size(A_g,2)~=N+q+1))||...
45              ((~isempty(b_g))&&(size(b_g,2)~=1))||...
46              ((~isempty(Aeq_g))&&(size(Aeq_g,2)~=N+q+1))||...
47              ((~isempty(beq_g))&&(size(beq_g,2)~=1))||...
48              (size(A_g,1)~=size(b_g,1))||...
49              (size(Aeq_g,1)~=size(beq_g,1))
50          fprintf( Matrix dimension mismatch or wrong! );
51          return;
52      end
53
54      %% construct constraint matrices
55      f = [q*(1-beta); zeros(N,1); ones(q,1)];
56      A = zeros(q+1, N+q+1);
57      m = mean(y,1);
58      b = [-R; zeros(q,1)];
59      Aeq = [0, ones(1,N), zeros(1,q)];
60      beq = 1;
61      LB = [-inf;lb_g_x;zeros(q,1)];
62      UB = [ inf;ub_g_x;inf*ones(q,1)];
63
64      A(1,2:N+1) = -m;
65      A(2:end,1) = -1;
66      A(2:end,2:N+1) = -y;
67      A(2:end,N+2:end) = -diag(ones(q,1));
68
69      A = [A;A_g];
70      b = [b;b_g];
71      Aeq = [Aeq;Aeq_g];
72      beq = [beq;beq_g];
73
74      %% linear prog. (cal. CPU time simultaneously)
75      tstart = tic;
76  %     fprintf( I really start linear programming! Good luck!\n );
77      [alpha_x_u, fval, flag] = linprog(f,A,b,Aeq,beq,LB,UB);
78
79      %% construct outputs
80      LPtime = toc(tstart);
81      alpha = alpha_x_u(1);
82      x = alpha_x_u(2:N+1);
83      obj = fval/q/(1-beta);
84  end
```

A.2 Rockafellar $\sum = 0$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CVaR optimization with LP algorithm;
% Rockafellar: Optimization of Conditional Value-at-Risk(2000)
%
% intrinsic constraint is the total_weight == 0
% sum(x+) = 1, sum(x-) = -1, x = (x+) + (x-)
% # of variables: 2N+q+1 (x+, x-, VaR, q extra var. for
%   converting to LP problem)
%
% alpha     : optimal value of VaR
% x         : optimal allocation (1 by 2N, N is the # of assets)
% obj       : optimal value of CVaR
% LPtime    : CPU time of linear programming in seconds
% flag      : indicator for linprog function
%
% y         : scenarios of returns (q by N, qi is the # of scenarios)
% beta      : confidence level
% R         : expected minimum return in the constraint set
% A_g, b_g  : A_g*x >= b_g (inequality constraints)
% Aeq_g, beq_g: Aeq_g*x == beq_g (equality constraints)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [alpha, x, obj, LPtime, flag] = cvar_rock_pm(y,beta,R,A_g,b_g,...
        Aeq_g,beq_g)
    %% init.
    [q,N] = size(y);

    if nargin~=3 && nargin~=5 && nargin~=7 && nargin~=9
        fprintf( Wrong number of arguments! );
        return;
    end


    if nargin<=5
        Aeq_g = [];
        beq_g = [];
        if nargin==3
            A_g = [];
            b_g = [];
        end
    end

    if ((~isempty(A_g))&&(size(A_g,2)~=2*N+q+1))||...
            ((~isempty(b_g))&&(size(b_g,2)~=1))||...
            ((~isempty(Aeq_g))&&(size(Aeq_g,2)~=2*N+q+1))||...
            ((~isempty(beq_g))&&(size(beq_g,2)~=1))||...
            (size(A_g,1)~=size(b_g,1))||...
            (size(Aeq_g,1)~=size(beq_g,1))
        fprintf( Matrix dimension mismatch or wrong! );
        return;
    end

    %% construct constraint matrices
    f = [q*(1-beta); zeros(2*N,1); ones(q,1)];
```

```
54      A = zeros(q+1, 2*N+q+1);
55      m = mean(y,1);
56      b = [-R; zeros(q,1)];
57      Aeq = [0,ones(1,N),zeros(1,N+q); ...
58             0,zeros(1,N),ones(1,N),zeros(1,q)];
59      beq = [1; -1];
60      LB = [-inf;zeros(N,1);-inf*ones(N,1);zeros(q,1)];
61      UB = [inf*ones(N+1,1);zeros(N,1);inf*ones(q,1)];
62
63      A(1,2:N+1) = -m;
64      A(1,N+2:2*N+1) = -m;
65      A(2:end,1) = -1;
66      A(2:end,2:N+1) = -y;
67      A(2:end,N+2:2*N+1) = -y;
68      A(2:end,2*N+2:end) = -diag(ones(q,1));
69
70      A = [A;A_g];
71      b = [b;b_g];
72      Aeq = [Aeq;Aeq_g];
73      beq = [beq;beq_g];
74
75      %% linear prog. (cal. CPU time simultaneously)
76      tstart = tic;
77  %     fprintf( I start linear programming! Good luck!\n );
78      [alpha_x_u, fval, flag] = linprog(f,A,b,Aeq,beq,LB,UB);
79
80      %% construct outputs
81      LPtime = toc(tstart);
82      alpha = alpha_x_u(1);
83      x = alpha_x_u(2:2*N+1);
84      obj = fval/q/(1-beta);
85  end
```

A.3 Iyengar $\sum = 1$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CVaR optimization with Fast Gradient Descent Method
% Iyengar: Fast gradient descent method for mean-CVaR optimization (2009)
%
% intrinsic constraint is the total_weight == 1
%
% input:
%   y           return matrix of N by n
%   beta        confidence level
%   M           weight of short < 1/M; M->inf means no short
%   R           return constraint
%   p           N by 1 vector, probability of each senario
%   epsilon     accuracy (should set to 1e-3)
%   A_g         the extra inequality constriant LHS
%   b_g         the extra inequality constriant RHS
%   Aeq_g       the extra equality constriant LHS
%   beq_g       the extra equality constriant RHS
%
% output:
%   var         optimized VaR
%   cvar        optimized CVaR
%   x           the weights for assets
%   LPtime      CPU time
%
% others:
%   N           the number of senarios
%   n           the number of assets
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [var, x, cvar, LPtime, PRtime] = cvar_iyen(y,p,beta,R,M,...
        epsilon,A_g,b_g,Aeq_g,beq_g,lb_g,ub_g)
    [N,n] = size(y); % find the senario # and asset # from rtn mtx

%% argument check
    if nargin==9 || nargin==7
        fprintf( Wrong number of arguments! );
        return;
    end

    if nargin<=10
        lb_g = [];
        ub_g = [];
        if nargin<=8
            Aeq_g = [];
            beq_g = [];
            if nargin<=6
                A_g = [];
                b_g = [];
            end
        end
    end

    if ((~isempty(A_g))&&(size(A_g,2)~=n))||...
            ((~isempty(b_g))&&(size(b_g,2)~=1))||...
```

```
54              ((~isempty(Aeq_g))&&(size(Aeq_g,2)~=n))||...
55              ((~isempty(beq_g))&&(size(beq_g,2)~=1))||...
56              (size(A_g,1)~=size(b_g,1))||...
57              (size(Aeq_g,1)~=size(beq_g,1))
58          fprintf( Matrix dimension mismatch or wrong! );
59          return;
60      end
61
62  %% init.
63      D1 = (1+2/M)^2/2;
64      D2 = -log(beta) - (1-beta)/beta*log(1-beta);
65      sigma2 = 1/(1-beta);
66      Omega = max(sum(y.*y, 2));
67      K = ceil(sqrt(Omega*D1*D2/sigma2)/epsilon);
68      mu = epsilon/2/D2;
69      w = ones(n,1)/n;
70
71      Q = zeros(N, K+1);
72      W = zeros(n, K+2);
73      Y = zeros(n, K+1);
74      Z = zeros(n, K+1);
75
76      W(:,1) = w;
77
78  %% constraint matrices
79      m = mean(y,1);
80      A = -m;
81      b = -R;
82      Aeq = ones(1,n);
83      beq = 1;
84
85      A = [A;A_g];
86      b = [b;b_g];
87      Aeq = [Aeq;Aeq_g];
88      beq = [beq;beq_g];
89      lb = lb_g;
90      ub = ub_g;
91
92      H = Omega/mu/sigma2*eye(n);
93      option = optimset( LargeScale , off , Display , off );
94
95  %% main
96  PRtime = zeros(K,1);
97  fprintf( The total # of iterations is: %d \n , K);
98  tstart = tic;
99  for i = 0:K
100     tprint = tic;
101     fprintf( %d -th iteration \n , i);
102     PRtime(i+1) = toc(tprint);
103     %% solve for q
104     [a,exp_alphabymu] = newton(y,W(:,i+1),p,beta,mu);
105     Q(:,i+1) = p/beta./(1+a*exp_alphabymu);
106
107     %% solve for y
108     fy = (-Q(:,i+1) *y - Omega/mu/sigma2*W(:,i+1) ) ;
109     Y(:,i+1) = quadprog(H,fy,A,b,Aeq,beq,lb,ub,[],option);
```

```
110
111         %% solve for z
112         fz = zeros(1,n);
113         for t = 0:i
114             f = -(t+1)/2*Q(:,t+1) *y;
115             fz = fz + f;
116         end
117         fz = fz ;
118         Z(:,i+1) = quadprog(H,fz,A,b,Aeq,beq,lb,ub,[],option);
119
120         %% update w
121         W(:,i+2) = Z(:,i+1)/(i+3)+Y(:,i+1)*(i+1)/(i+3);
122     end
123     LPtime = toc(tstart);
124
125
126     x = Y(:,K+1);
127     [var, cvar] = var_cvar(y,x,beta);
```

A.4 Iyengar $\sum = 0$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CVaR optimization with Fast Gradient Descent Method
% Iyengar: Fast gradient descent method for mean-CVaR optimization (2009)
%
% intrinsic constraint is the total_weight == 0
% sum(x+) = 1, sum(x-) = -1, x = (x+) + (x-)
%
% input:
%   y           return matrix of N by n
%   beta        confidence level
%   M           weight of short < 1/M; M->inf means no short
%   R           return constraint
%   p           N by 1 vector, probability of each senario
%   epsilon     accuracy (should set to 1e-3)
%   A_g         the extra inequality constriant LHS
%   b_g         the extra inequality constriant RHS
%   Aeq_g       the extra equality constriant LHS
%   beq_g       the extra equality constriant RHS
%
% output:
%   var         optimized VaR
%   cvar        optimized CVaR
%   x           the weights for assets
%   LPtime      CPU time
%
% others:
%   N           the number of senarios
%   n           the number of assets
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [var, x, cvar, LPtime, PRtime] = cvar_iyen_pm(y,p,beta,R,M,...
        epsilon,A_g,b_g,Aeq_g,beq_g,lb_g,ub_g)
    [N,n] = size(y); % find the senario # and asset # from rtn mtx
    y = [y,y];

%% argument check
    if nargin==9 || nargin==7
        fprintf( Wrong number of arguments! );
        return;
    end

    if nargin<=10
        lb_g = [];
        ub_g = [];
        if nargin<=8
            Aeq_g = [];
            beq_g = [];
            if nargin<=6
                A_g = [];
                b_g = [];
            end
        end
    end

```

```
54      if ((~isempty(A_g))&&(size(A_g,2)~=2*n))||...
55              ((~isempty(b_g))&&(size(b_g,2)~=1))||...
56              ((~isempty(Aeq_g))&&(size(Aeq_g,2)~=2*n))||...
57              ((~isempty(beq_g))&&(size(beq_g,2)~=1))||...
58              (size(A_g,1)~=size(b_g,1))||...
59              (size(Aeq_g,1)~=size(beq_g,1))
60          fprintf( Matrix dimension mismatch or wrong! );
61          return;
62      end
63
64  %% init.
65      D1 = (1+2/M)^2/2;
66      D2 = -log(beta) - (1-beta)/beta*log(1-beta);
67      sigma2 = 1/(1-beta);
68      Omega = max(sum(y.*y, 2));
69      K = ceil(sqrt(Omega*D1*D2/sigma2)/epsilon);
70      mu = epsilon/2/D2;
71      w = [ones(n,1);-ones(n,1)]/n;
72
73      Q = zeros(N, K+1);
74      W = zeros(2*n, K+2);
75      Y = zeros(2*n, K+1);
76      Z = zeros(2*n, K+1);
77
78      W(:,1) = w;
79
80  %% constraint matrices
81      m = mean(y,1);
82      A = -m;
83      b = -R;
84      Aeq = [[ones(1,n),zeros(1,n)];[zeros(1,n),ones(1,n)]];
85      beq = [1;-1];
86      lb = [zeros(1,n),-ones(1,n)*Inf];
87      ub = [ones(1,n)*Inf,zeros(1,n)];
88
89      A = [A;A_g];
90      b = [b;b_g];
91      Aeq = [Aeq;Aeq_g];
92      beq = [beq;beq_g];
93      lb = [lb,lb_g];
94      ub = [ub,ub_g];
95
96      H = Omega/mu/sigma2*eye(2*n);
97      option = optimset( LargeScale , off , Display , off );
98
99  %% main
100 PRtime = zeros(K,1);
101 fprintf( The total # of iterations is: %d \n , K);
102 tstart = tic;
103 for i = 0:K
104     tprint = tic;
105     fprintf( %d -th iteration \n , i);
106     PRtime(i+1) = toc(tprint);
107     %% solve for q
108     [a,exp_alphabymu] = newton(y,W(:,i+1),p,beta,mu);
109     Q(:,i+1) = p/beta./(1+a*exp_alphabymu);
```

```
110
111       %% solve for y
112       fy = (-Q(:,i+1) *y - Omega/mu/sigma2*W(:,i+1) ) ;
113       Y(:,i+1) = quadprog(H,fy,A,b,Aeq,beq,lb,ub,[],option);
114
115       %% solve for z
116       fz = zeros(1,2*n);
117       for t = 0:i
118           f = -(t+1)/2*Q(:,t+1) *y;
119           fz = fz + f;
120       end
121       fz = fz ;
122       Z(:,i+1) = quadprog(H,fz,A,b,Aeq,beq,lb,ub,[],option);
123
124       %% update w
125       W(:,i+2) = Z(:,i+1)/(i+3)+Y(:,i+1)*(i+1)/(i+3);
126   end
127   LPtime = toc(tstart);
128
129
130   x = Y(:,K+1);
131   [var, cvar] = var_cvar(y,x,beta);
```

A.5 Multivariate normal random number generator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generating multivariate normal random sets
%
% vec(y) = vec(ave) + chol(cov)*vec(Z)
% where ave is the vector of means
%       y satisfies MN(mu,sigma)
%       cov is the covariance matrix
%       Z is a vector of indpt. normal random numbers
%
% http://www.columbia.edu/~mh2078/MCS04/MCS_framework_FEegs.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = MN_chol(mu,sigma,N)
    C = chol(sigma);
    Z = randn(length(mu),N);
    y = ((C )*Z) + ones(N,1)*mu;
end
```

A.6 Multivariate student-t random number generator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generating multivariate student-t random sets
%
% vec(x) = vec(ave) + vec(y)*sqrt(n/u)
% where ave is the vector of means
%       y satisfies N(0,cova)
%       u satisfies chi-squared(n)
%       n is the DF
%
%       N # of scenarios generated
%
% MVN is generated by Choleski decomp.
%
% ref: http://dx.doi.org/10.1016/S0731-9053(08)22003-7
%      <The skewed t distribution for portfolio credit risk>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = mvt_chol(ave, cova, n, N)
C = chol(cova);
Z = randn(length(ave), N);
u = chi2rnd(n, [N,1]);
y = ones(N,1)*ave + (C * Z) .*repmat(sqrt(n./u),[1,length(ave)]);
end
```

A.7 Multivariate skewed student-t random number generator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generating multivariate skewed student-t random sets
%
% vec(x) = vec(ave) + vec(g)*(n/u) + vec(y)*sqrt(n/u)
% where ave is the vector of means
%       y satisfies N(0,cova)
%       u satisfies chi-squared(n)
%       g is the vector of skewness for all variables
%       n is the DF
%
%       N # of scenarios generated
%
% MVN is generated by Choleski decomp.
%
% ref: http://dx.doi.org/10.1016/S0731-9053(08)22003-7
%      <The skewed t distribution for portfolio credit risk>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = mvskewt_chol(ave, cova, nu, g, N)
C = chol(cova);
Z = randn(length(ave), N);
u = chi2rnd(nu, [N,1]);
y = ones(N,1)*ave + (ones(N,1)*g ) .*repmat(nu./u,[1,length(ave)])...
    + (C * Z) .*repmat(sqrt(nu./u),[1,length(ave)]);
end
```

A.8 Multivariate NIG random number generator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generating multivariate NIG random sets
%
% vec(mnig) = vec(ave) + ig*Sigma*beta + sqrt(ig)*MVN(0,Sigma)
%
% alpha represents the size of the fat tails
% beta  represents the skewness
% ig    satisfies the univariate IG distribution
%
% wikipedia       Oigard s paper
% IG(mu,lambda) = IG(sqrt(chi/psi),chi)
%               = IG(delta/sqrt(alpha^2-beta*Gamma*beta),delta^2)
%
% IG(mu,lambda) random number generated from gaussian random number
% http://en.wikipedia.org/wiki/Inverse_Gaussian_distribution
%
% DOI: 10.1016/j.sigpro.2005.03.005
% <EM-estimation and modeling of heavy-tailed processes
% with the multivariate normal inverse Gaussian distribution>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function mnig = nig_chol(alpha_,beta_,delta_,mu_,Sigma_,N)
    D = length(mu_);

    %% generate IG(p1,p2) [w/ notation in Wikipedia]
    p1 = delta_/sqrt(alpha_^2 - beta_ * Sigma_*beta_);
    p2 = delta_^2;

    nu = randn(N,1);
    nu = nu.^2;

    x = p1 + p1^2*nu/2/p2 - p1/2/p2*sqrt(4*p1*p2*nu+p1^2*nu.^2);

    temp = [x,p1^2./x];
    z = (rand(N,1)>p1./(p1+x));
    ig = temp(z*N+(1:N) );

    %% generate MNIG
    mnig = repmat(mu_,1,N) + Sigma_*beta_*ig  ...
        + chol(Sigma_) *randn(D,N).*repmat(sqrt(ig) ,D,1);
    mnig = mnig ;
end
```

A.9 Empirical VaR and CVaR

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate the empirical VaR and CVaR
%
% y: q by N matrix, N assets with q scenarios
% x: allocation for N assets
% beta: confidence level
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [var, cvar] = var_cvar(y, x, beta)
    q = size(y,1);

    rtn = y*x;
    rtn = sort(rtn);
    serialN_real = (1-beta)*q;
    serialN_inte = floor(serialN_real);

    var = -rtn(serialN_inte)*(serialN_inte+1-serialN_real) - ...
        rtn(serialN_inte+1)*(serialN_real-serialN_inte);
    cvar = -mean([rtn(1:serialN_inte-1);-var]);
end
```

A.10 Newton-Raphson method

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Newton method for solving the explicit p in Iyengar
%
%   y: the return matirx of the size of N by n, where
%       N ~ # of senarios, n ~ # of assets
%   w: the weights of assets of n by 1
%   p: the prob. of senarios of N by 1
%   beta: the confidence level
%   mu: a constant defined in Iyengar (epsilon/2/D2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [a,exp_alphabymu] = newton(y, w, p, beta, mu)
    a = exp(y*w/mu);

    % initial guess of the root
    x0 = (1-beta)/beta/mean(a);

    delta = 1; epsilon = 1e-5;
    while delta > epsilon
    fx  = sum(p./(1+a*x0)) - beta;
    fxp = -sum(p.*a./(1+a*x0).^2);
    x1  = x0 - fx/fxp;
    delta = abs(x1 - x0);
    x0 = x1;
    end

    exp_alphabymu = x1;
end
```

A.11 Shrinkage for small number of scenarios

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% shrink the covariance matrix
%
% http://www.iijournals.com/doi/abs/10.3905/jpm.2004.110
% Ledoit(2004): <Honey, I Shrunk the Sample Covariance Matrix>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [meanx,sigma,shrinkage]=cov1para(x,shrink)

% function sigma=cov1para(x)
% x (t*n): t iid observations on n random variables
% sigma (n*n): invertible covariance matrix estimator
%
% Shrinks towards one-parameter matrix:
%    all variances are the same
%    all covariances are zero
% if shrink is specified, then this const. is used for shrinkage

% de-mean returns
[t,n]=size(x);
meanx=mean(x,1);
x=x-meanx(ones(t,1),:);

% compute sample covariance matrix
sample=(1/t).*(x'*x);

% compute prior
core = sqrt(diag(sample)*diag(sample)');
r = sample./core;
r_ave = (sum(sum(r)) - sum(diag(r)))*2/n/(n-1);
prior = r_ave*(core-diag(diag(sample)))+diag(diag(sample));

if (nargin < 2 || shrink == -1) % compute shrinkage parameters

  % ppi
  ppi = ((x .^2)'*(x.^2)-2*sample.*(x'*x))/t + sample.^2;

  % theta
  theta12 = ((x .^3)'*x-(diag(sample)*ones(1,n)).*(x'*x)...
      -(sum(x .^2,2)*ones(1,n)).*sample...
      +((diag(sample)*ones(1,n)).*sample))/t;

  theta21 = (x'*(x.^3)-(ones(n,1)*diag(sample)').*(x'*x)...
      -(ones(n,1)*sum(x.^2,1)).*sample...
      +(ones(n,1)*(diag(sample)').*sample))/t;

  % rho
  sample_i = diag(sample)*ones(1,n);
  sample_j = ones(n,1)*diag(sample)';
  rho_2nd = sqrt(sample_j./sample_i).*theta12...
      +sqrt(sample_i./sample_j).*theta21;
  rho = sum(diag(ppi)) + r_ave/2*sum(sum(rho_2nd-diag(diag(rho_2nd))));

```

```
54    % gamma
55    gamma=norm(sample-prior, fro )^2;
56
57    % compute shrinkage constant
58    kappa=(sum(sum(ppi))-rho)/gamma;
59    shrinkage=max(0,min(1,kappa/t));
60
61  else % use specified number
62    shrinkage=shrink;
63  end
64
65  % compute shrinkage estimator
66  sigma=shrinkage*prior+(1-shrinkage)*sample;
```

A.12 Multivariate student-t estimator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EM algorithm for fitting multivariate t to the data
%
% ref: http://dx.doi.org/10.1016/S0731-9053(08)22003-7
%      <The skewed t distribution for portfolio credit risk>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [ave, cova, nu] = mvt_EM(data)
    [N,D] = size(data);
    tol = 1e-7;
    LLF_prev = -1000;
    LLF = -5;
    nu_init = 15;

    options  =  optimset( fzero );
    options  =  optimset(options ,  Display      ,  off );
    options  =  optimset(options ,  MaxFunEvals  , 5000);
    options  =  optimset(options ,  MaxIter      , 5000);
    options  =  optimset(options ,  TolFun       , 1e-12);
    options  =  optimset(options ,  TolX         , 1e-12);

    %% init.
    ave = mean(data,1) ;
    cova = cov(data);
    nu = nu_init;

    %% EM algorithm
    while abs((LLF - LLF_prev)/LLF_prev) > tol
        %     fprintf( LLF = %6.4f \n , LLF);
        %% aux. variables
        data_m0 = data - repmat(ave ,N,1);
        rho = dot(data_m0/cova, data_m0, 2);
        theta = (nu + D)./(rho + nu);
        %     eta = (rho + nu)/(nu + d - 2);
        zzeta = log(.5*(rho + nu)) - psi(.5*(D + nu));

        %% means
        theta_m = mean(theta);
        %     eta_m = mean(eta);
        zzeta_m = mean(zzeta);

        %% update ave, cova, nu
        ave = mean(repmat(theta,1,D).*data) /theta_m;
        data_m0 = data - repmat(ave ,N,1);
        cova = (repmat(theta,1,D).*data_m0) *data_m0/N;

        %     fprintf( solve\n ) ;
        nu = fsolve(@(x) -psi(.5*x)+log(.5*x)+1-zzeta_m-theta_m, nu, options);
        %     fprintf( nu = %6.4f,\t LLF = %6.4f\n , nu, LLF);

        %% LLF update
        LLF_prev = LLF;
```

```
54          const = gamma((nu+D)*.5)/(gamma(.5*nu)*(pi*nu)^(.5*D)*sqrt(det(cova)));
55          LLF = N*log(const) - sum(log(1+rho/nu))*(nu+D)/2;
56      end
57      ave = ave ;
58
59  end
```

A.13 Multivariate skewed student-t estimator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EM algorithm for fitting multivariate t to the data
%
% ref: http://dx.doi.org/10.1016/S0731-9053(08)22003-7
%      <The skewed t distribution for portfolio credit risk>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [ave, cova, nu, g] = mvskewt_EM(data)
    [N,D] = size(data);
    tol = 1e-7;
    LLF_prev = -1000;
    LLF = -5;
    nu_init = 15;

    options  =  optimset( fzero );
    options  =  optimset(options ,  Display      ,  off );
    options  =  optimset(options ,  MaxFunEvals  , 5000);
    options  =  optimset(options ,  MaxIter      , 5000);
    options  =  optimset(options ,  TolFun       , 1e-12);
    options  =  optimset(options ,  TolX         , 1e-12);

    %% init.
    ave = mean(data,1) ;
    ave_data = ave;
    cova = cov(data);
    nu = nu_init;
    g =[1;-1];
    g = repmat(g,length(ave)/2,1);

    %% EM algorithm
    data_m0 = data - repmat(ave ,N,1);
    rho = dot(data_m0/cova, data_m0, 2);
    gSg = g / cova*g;
    term1 = (rho + nu)/gSg;
    term2 = sqrt((rho + nu)*gSg);
    bessel_p1 = besselk(.5*(nu+D)+1,term2);
    bessel_00 = besselk(.5*(nu+D)  ,term2);
    bessel_m1 = besselk(.5*(nu+D)-1,term2);
%     bessel_00_p = besselk(.5*(nu+D)*(1+1e-12), term2);
%     bessel_00_m = besselk(.5*(nu+D)*(1-1e-12), term2);
%     bessel_00_d = (bessel_00_p - bessel_00_m)/2e-12/(nu+D);

    n = 0;
    while abs((LLF - LLF_prev)/LLF_prev) > tol
        n = n + 1;
        fprintf( n = %d, LLF = %6.4f \n , n, LLF);
        %% E-step: aux. variables
        theta = 1./sqrt(term1).*bessel_p1./bessel_00;
        eta   = sqrt(term1).*bessel_m1./bessel_00;

        zzeta = zeros(N,1);
        for  ii = 1:N
```

```
54              zzeta(ii) = quadgk(@(y) 0.5.*(y.^(-.5*(nu+D)-1)).*log(y).*exp(-.5.*term2(ii).*(y+1./y)),0,
55          end
56          zzeta = .5*log(term1) + zzeta./bessel_00;
57  %         zzeta = .5*log(term1) + bessel_00_d./bessel_00;
58
59          theta = real(theta);
60          eta = real(eta);
61          zzeta = real(zzeta);
62
63          %% means
64          theta_m = mean(theta);
65          eta_m = mean(eta);
66          zzeta_m = mean(zzeta);
67
68          %% M-step: update g, ave, cova, nu
69          data_m0_data = data - repmat(ave_data ,N,1);
70          g = -mean(repmat(theta,1,D).*data_m0_data) / (theta_m*eta_m - 1);
71          ave = (mean(repmat(theta,1,D).*data)  - g)/theta_m;
72          cova = (repmat(theta,1,D).*data_m0) *data_m0/N - g*g *eta_m;
73
74          %     fprintf( solve\n );
75          nu = fsolve(@(x) -psi(.5*x)+log(.5*x)+1-zzeta_m-theta_m, nu, options);
76          %     fprintf( nu = %6.4f,\t LLF = %6.4f\n ,  nu, LLF);
77
78          g = real(g);
79          ave = real(ave);
80          cova = real(cova);
81          nu = real(nu);
82
83          %% LLF update
84          LLF_prev = LLF;
85          data_m0 = data - repmat(ave ,N,1);
86          rho = dot(data_m0/cova, data_m0, 2);
87          gSg = g / cova*g;
88          term1 = (rho + nu)/gSg;
89          term2 = sqrt((rho + nu)*gSg);
90          bessel_p1 = besselk(.5*(nu+D)+1,term2);
91          bessel_00 = besselk(.5*(nu+D)  ,term2);
92          bessel_m1 = besselk(.5*(nu+D)-1,term2);
93  %         bessel_00_p = besselk(.5*(nu+D)*1.005, term2);
94  %         bessel_00_m = besselk(.5*(nu+D)* .995, term2);
95  %         bessel_00_d = (bessel_00_p - bessel_00_m)/(.01*(nu+D));
96
97          const = 2^(1-.5*(nu+D))/(gamma(.5*nu)*(pi*nu)^(.5*D)*sqrt(det(cova)));
98          LLF = N*log(const) - sum(log(1+rho/nu))*(nu+D)/2 ...
99              + sum(log(bessel_00)) + sum(log(term2))*(nu+D)/2....
100             + sum(dot(data_m0/cova,repmat(g ,N,1)));
101     end
102     ave = ave ;
103
104 end
```

A.14 Multivariate NIG estimator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EM algorithm for fitting multivariate NIG to the data
%
% ref: http://dx.doi.org/10.1016/j.sigpro.2005.03.005
%      <EM-estimation and modeling of heavy-tailed processes
%       with the multivariate normal inverse Gaussian distribution>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [alpha_,beta_,delta_,mu_,Sigma_, n] = nig_EM(data)
    [N,D] = size(data);
    tol = 1e-5;
    LLF_prev = -1000;
    LLF = -5;

    %% init.
    alpha_ = 1;
    beta_  = zeros(D,1);
    delta_ = 1;
    mu_    = mean(data,1) ;
    Sigma_ = cov(data);
    Sigma_ = Sigma_/det(Sigma_)^(1/D);

    ave = mu_;

    %% EM algorithm
    n = 0;
    q_x = q(data ,delta_,mu_,Sigma_);
    while abs((LLF - LLF_prev)/LLF_prev) > tol
        %% E-step, aux. variables
        bessel_p3 = besselk(.5*(D+3),alpha_*q_x);
        bessel_p1 = besselk(.5*(D+1),alpha_*q_x);
        bessel_m1 = besselk(.5*(D-1),alpha_*q_x);

        stigma = q_x/alpha_.*bessel_m1./bessel_p1;
        phi = alpha_./q_x.*bessel_p3./bessel_p1;
        stigma_m = mean(stigma);
        theta_m = 1/mean(phi-1/stigma_m);

        %% M-step, delta, beta, mu, alpha, Sigma
        delta_ = sqrt(theta_m);

        data0 = data  - repmat(mu_,1,N);
        R = (data0.*repmat(phi ,D,1))*data0 / N;
        A = stigma_m*(beta_*beta_ );
        [W,D_ar] = eig(A*R);
        B = .5*(-eye(D)+W*sqrtm(eye(D)+4*D_ar)/W);
        Sigma_ = R/(B+eye(D));
        Sigma_ = Sigma_/det(Sigma_)^(1/D);

        beta_ = Sigma_\((data *phi - ave*sum(phi))/(N-stigma_m*sum(phi)));
        mu_ = ave - stigma_m*Sigma_*beta_;
        alpha_= sqrt((delta_/stigma_m)^2 + beta_ * Sigma_*beta_);

```

```
54          %% real part only
55          delta_ = real(delta_);
56          alpha_ = real(alpha_);
57          mu_ = real(mu_);
58          beta_ = real(beta_);
59          Sigma_ = real(Sigma_);
60
61          %% LLF update
62          LLF_prev = LLF;
63          const_log = .5*log(stigma_m)-log(besselk(-.5,delta_^2./stigma_m));
64          q_x = q(data ,delta_,mu_,Sigma_);
65          LLF = N*const_log + sum(log(besselk(.5*(D+1),alpha_*q_x)))...
66              - .5*(D+1)*(sum(log(q_x))-N*log(alpha_))...
67              + sum(beta_*(data  - repmat(mu_,1,N)));
68          n = n + 1;
69          fprintf( n = %d, LLF = %6.4f\n , n, LLF);
70  %         if n == 746
71  %             fprintf( hello, i am here! );
72  %         end
73      end
74
75  end
76
77  function val = q(x,delta,mu,Sigma)
78      N = size(x,2);
79      if N ~= 1
80          mu = repmat(mu,1,N);
81      end
82      val = sqrt(delta^2 + dot((x-mu) /Sigma,(x-mu) ,2));
83  end
```

Bibliography

[1] Mosek aps - fruebjergvej 3, box 16 - 2100 copenhagen o - denmark, 2011.

[2] S. Alexander, T.F. Coleman, Y. Li, Minimizing var and cvar for a portfolio of derivatives, Journal of Banking and Finance 30 (2006) 583–605.

[3] T.W. Anderson, D.A. Darling, A test of goodness of fit, Journal of the American Statistical Association 49 (1954) pp. 765–769.

[4] W. Hu, A.N. Kercheval, The skewed-t distribution for portfolio credit risk (2008).

[5] G. Iyengar, A. Ma, Fast gradient descent method for mean-cvar optimization, preprint (2009).

[6] A. Justel, D. Peña, R. Zamar, A multivariate kolmogorov-smirnov test of goodness of fit, Statistics and Probability Letters 35 (1997) 251 – 259.

[7] O. Ledoit, M. Wolf, Honey, i shrunk the sample covariance matrix, The Journal of Portfolio Management 30 (2004) 110 – 119.

[8] H. Markowitz, Portfolio selection, The Journal of Finance 7 (1952) 77–91.

[9] J. Massey, Frank J., The kolmogorov-smirnov test for goodness of fit, Journal of the American Statistical Association 46 (1951) pp. 68–78.

[10] A. McNeil, R. Frey, P. Embrechts, Quantitative Risk Management: Concepts, Techniques, and Tools, Princeton University Press, 2006.

[11] Y. Nesterov, Smooth minimization of non-smooth functions, Mathematical Programming 103 (2005) 127–152. 10.1007/s10107-004-0552-5.

[12] P.C. O'Brien, T.R. Fleming, A multiple testing procedure for clinical trials, Biometrics 35 (1979) pp. 549–556.

[13] T.A. Øigård, A. Hanssen, R.E. Hansen, F. Godtliebsen, Em-estimation and modeling of heavy-tailed processes with the multivariate normal inverse gaussian distribution, Signal Processing 85 (2005) 1655 – 1673.

[14] R. Rockafellar, S. Uryasev, Optimization of conditional value-at-risk, The Journal of Risk 2 (2000) 21–41.

[15] W.F. Sharpe, Capital asset prices: A theory of market equilibrium under conditions of risk, The Journal of Finance 19 (1964) pp. 425–442.