Convex and Nonsmooth Optimization Final Project
Risk Minimization by Conditional Value-at-Risk

Charlie Chen

May 2022

1 Introduction

Evaluating risks and balance uncertainty in portfolio is one of the most crucial topics in mathematical finance. The appropriate evaluation and minimization of risks ensure that investors will not easily lose their money and therefore money regulator can earn the agency fee. In the field of risk, there are two important indicators: Conditional Value-at-Risk (CVaR) and Value-at-Risk (VaR). There has been many papers proposing different ways to minimize these two measurements. But one of the most classical work is *Optimization of Conditional Value-at-Risk* published in 2000 by R. Tyrrell Rockafellar and Stanislav Uryasev[3]. In the paper, the authors proposes an effective way to optimize Conditional Value-at-Risk and getting the value of Value-at-Risk simultaneously in a convex programming manner if the loss function is convex. Because of its nice formation, linear programming and nonsmooth optimizations can be utilized to optimize the loss.

This paper will first explain what those two conditional variables are, how we can understand and analyze them, and supplement readers with related financial knowledge and terminology. Next, it will discuss the main methods in the paper and its advantages compared to previous methods. Then, we will show different ways to optimize the loss function.

2 Essential Terminology and Background knowledge

2.1 Basic Model

Define vector $x \in \mathbb{R}^n$ to be the portfolio, where x_i is the proportion to ith instrument (investment). Let vector $y \in \mathbb{R}^m$ be the market factors, or uncertainties, which are uncontrollable but influence revenue. For example, y may be the price in the future. It is a random variable. In reality, we probably don't have the distribution curve for y, but only sample points. For the sake of simplicity, let's first assume $p(y)$ is known. As we expose the algorithm itself, we will see it is not necessary.

Then, we have our loss function $f(x, y)$, denoting the loss of holding such portfolio x with market factor y. f is defined as $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$.

A naive example may be this: let $x, y \in \mathbb{R}$, x_i be the quantities of bond i and y_i be the price difference of bond i next year compared to this year. The loss function can be defined as $f(x, y) = -x^T y$. We want to maximize our holding of the bond. Therefore, we define the loss as the negative value of our earning.

If we assume that the price difference y is rather deterministic and is known in prior. This reduces to a simple linear programming. The solution, intuitively, is to invest all the money in the instrument that is the most profitable.

If we assume that the price difference y is stochastic but the distribution is known, or assumed to be joint normal with x, we can still conduct stochastic programming on it, which is beyond the scope of this paper. Notice that in general, higher returns usually means higher risks. So to ensure that the risk is low, there is usually some compromise for the return.

However, recall that it is more realistic that y is stochastic and $p(y)$ is not known in prior. We can only infer $p(y)$ based on historical data.

2.2 More complicated concepts

Notice that because the definition for loss functions can be diverse. It is hard to manipulate them in a more general way. So, we want to "normalize" this loss into probability percentage.

Motivated by this, let's define

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

which is a cumulative distribution function. The value monotonically increasing from 0 to 1 when α is increasing and x is fixed. Ψ is the cumulative probability that the loss is lower than α.

This is very closed to what we mean by Value-at-Risk and Conditional Value-at-Risk. We will define them now.

Value-at-Risk: β-VaR is defined as

$$\alpha_\beta(x) = \min\{\alpha \in \mathbb{R} : \Psi(x, \alpha) \geq \beta\}$$

β-VaR is then, with probability β, the loss of which the portfolio will not exceed. If $\alpha_\beta(x) = A$, we can say that, with confidence β, the loss of portfolio x will not be greater than A. Ideally, we want β-VaR to be as small as possible, because small value of β-VaR guarantees small loss of the portfolio, with β confidence. Some common choice of β are $0.90, 0.95, 0.99$ [3].

β-VaR is quite a straightforward indicator for the loss, but it has some mathematical drawbacks, like lack of subaddictivity and convexity. It also relies on linear approximation of risks and normal distribution on market factors y. Without these assumptions, it may have several local maxima. So the portfolio may end up becoming sub-optimal. So another concept, though highly correlated, is proposed.

Conditional Value-at-Risk: β-CVaR is defined as

$$\phi_\beta(x) = (1-\beta)^{-1} \int_{f(x,y)\geq \alpha_\beta(x)} f(x,y)p(y)dy$$

This is relatively more complicated. It denotes the complement of β-VaR. It measures the expected loss when unfortunately the bad case happens, namely the loss exceeds β-VaR, which, though, has only probability $1-\beta$. The first term $(1-\beta)^{-1}$ is a normalization term, which can be discarded during optimization because it is unnecessary and may lead to numerical instability. This is called Conditional Value-at-Risk because it measure the loss on the condition that bad things happen. In fact, we can rewrite the definition as

$$\phi_\beta(x) = (1-\beta)^{-1}\mathbb{E}_{y\sim p(y)}[f(x,y)|f(x,y)\geq \alpha_\beta(x)]$$

Similarly, we want to minimize ϕ_β because small value of β-VaR guarantees that the expected loss it was bad luck.

CVaR was an emerging concept used in evaluating risks since it has better mathematical properties and minimization of CVaR is closely related to VaR. It was gaining more applications during the time the paper was written and several similar approaches were written by the time.

In the paper [3], Rockafeller and Uryasev proposed a way to optimizing CVaR and calculating the value of VaR at the same time. The technique is compatible with both linear and non-linear derivatives (financial products?) and leads to solving stochastic optimization problem, which has more a lot of literature and algorithms to employ.

3 Main Methods

3.1 Main Formula

The most important formula of the paper[3] is the following

$$F_\beta(x,\alpha) = \alpha + (1-\beta)^{-1}\int_{y\in\mathbb{R}^m} [f(x,y)-\alpha]^+ p(y)dy \tag{1}$$

where $[t]^+ = \max(0,t)$.

The authors[3] then claimed Theorem 1 that this $F_\beta(x,\alpha)$ is convex and continuously differentiable. And β-CVaR and β-VaR can be derived from this:

$$\phi_\beta(x) = \min_{\alpha \in \mathbb{R}} F_\beta(x, \alpha)$$
$$A_\beta(x) = \arg\min_{\alpha \in \mathbb{R}} F_\beta(x, \alpha)$$
$$\alpha_\beta(x) = \min A_\beta(x)$$

Note that $A_\beta(x)$ is a nonempty, closed, bounded interval. The proof can be found in the Appendix in [3].

The definition of F combine CVaR and VaR in one unified formula and allows optimization of CVaR without explicitly calculating VaR, which CVaR depends upon by definition. Also, it is remarkable that β-VaR can be directly evaluated by the F. Though, it is different from optimizing two indicators at the same time.

3.2 Discretization

In reality, we can only make approximations by sampling. So, the authors also proposed the discrete representations which depends upon a collection of vectors $y_1, \ldots, y_q$, instead of the density function $p(y)$. The formula is the following:

$$\tilde{F}_\beta(x, \alpha) = \alpha + \frac{1}{q(1-\beta)} \sum_{k=1}^{q} [f(x, y_k) - \alpha]^+$$

Since it is an approximation to a convex and differentiable function, it is not surprised that this function is convex and piecewise linear, which can be optimized by line search or linear programming.

3.3 Optimization

Theorem 2 of [3] claims that minimization of β-CVaR with respect to x is equivalent to minimize $F_\beta(x, \alpha)$ over (x, α). In other words,

$$\min_{x \in X} \phi_\beta(x) = \min_{(x,\alpha) \in X \times \mathbb{R}} F_\beta(x, \alpha)$$

In particular, if and only if we find x^* minimizes the left hand side, x^* and corresponding $\alpha^* \in A_\beta(x^*)$ minimizes the right hand side. Notice that F is convex with respect to α and also x if $f(x, y)$ is convex. This reduces the optimization to convex programming.

4 Experiments

The authors[3] gave two case studies in the paper: profiling and hedging. The profiling example has the assumptions that the distribution for y is normal

Table 1: Portfolio mean return: This is reproduced from [3]

Instrument	Mean return
S&P	0.0101110
Gov. bond	0.0043532
Small cap	0.0137058

Table 2: Portfolio Covariance Matrix: This is reproduced from [3]

	S&P	Gov. bond	Small cap
S&P	0.00324625	0.00022983	0.00420395
Gov. bond	0.00022983	0.00049937	0.00019247
Small cap	0.00420395	0.00019247	0.00764097

and the mean and variance are known. The hedging example is based on real statistics and doesn't assume that the distribution is normal. However, the dataset for the second is not public. In this section, we will re-implement the first example with different optimization methods and compare the results with the original paper.

4.1 Context Preparation: Profiling

In short, profiling means choosing different equities in appropriate and limited proportions to maximize the revenue and minimize the risks. Following the scenario in [3], we use x represents the proportion of financial instruments and y represents the return corresponding to those instruments. Both x and y are represented in percentage and x has the constraints that $x_j \geq 0, \sum_{j=1}^{n} x_j = 1$. The loss function is defined as $f(x, y) = -x^T y$. $f(x, y)$ is affine, so it is clearly convex, so by the Theorem 2 stated in Section 3.3, $F(x, \alpha)$ is convex with respect to both x and α. y is assumed to be joint normal and the means and covariance are shown in Table 1 and Table 2. Let $\mu(x)$ and $\sigma(x)$ define the mean and variance of some portfolio x. Therefore, we have:

$$\mu(x) = -x^T m \text{ and } \sigma^2(x) = x^T V x$$

Further, we use the discretization of $\tilde{F}(x, \alpha)$, which is piecewise linear in our case. The results in the original paper is given in Table 3.

Table 3: Results of the original paper [3](reproduced)

β	Sample #	S&P	Bond	Small Cap	VaR	VaR Diff(%)	CVaR	CVaR Diff(%)	Iterations	Time(min)
0.9	1000	0.35250	0.15382	0.49368	0.06795	0.154	0.09962	2.73	1157	0.0
0.9	3000	0.55726	0.07512	0.36762	0.06537	3.645	0.09511	1.92	636	0.0
0.9	5000	0.42914	0.12436	0.44649	0.06662	1.809	0.09824	1.30	860	0.1
0.9	10000	0.48215	0.10399	0.41386	0.06622	2.398	0.09503	2.00	2290	0.3
0.9	20000	0.45951	0.11269	0.42780	0.06629	2.299	0.09602	0.98	8704	1.5
0.95	1000	0.53717	0.08284	0.37999	0.09224	2.259	0.11516	0.64	156	0.0
0.95	3000	0.54875	0.07839	0.37286	0.09428	4.524	0.11888	2.56	652	0.0
0.95	5000	0.57986	0.06643	0.35371	0.09175	1.715	0.11659	0.59	388	0.1
0.95	10000	0.47102	0.10827	0.42072	0.08927	1.03	0.11467	1.00	1451	0.2
0.95	20000	0.49038	0.10082	0.40879	0.09136	1.284	0.11719	1.11	2643	0.7
0.99	1000	0.41844	0.12848	0.45308	0.13454	1.829	0.14513	5.12	340	0.0
0.99	3000	0.61960	0.05116	0.32924	0.12791	3.187	0.14855	2.89	1058	0.0
0.99	5000	0.63926	0.04360	0.31714	0.13176	0.278	0.15122	1.14	909	0.1
0.99	10000	0.45203	0.11556	0.43240	0.12881	2.51	0.14791	3.31	680	0.1
0.99	20000	0.45766	0.11340	0.42894	0.13153	0.451	0.15334	0.24	3083	0.9

4.2 Linear Programming

We can model the optimization of the problem to linear programming. In specific, following the construction of [3]:

$$\begin{aligned}
\min_{\alpha,u_k} \quad & \alpha + \frac{1}{q(1-\beta)}\sum_{k=1}^{q} u_k \\
\text{s.t.} \quad & u_k \geq 0 \\
& x^T y_k + \alpha + u_k \geq 0 \\
& x_k \geq 0 \\
& \sum_{k=1}^{r} x_k = 1 \\
& \mu(x) = -x^T m \leq -R
\end{aligned}$$

for $k = 1, \ldots, r$

y_k is sampled through the distribution according to the means and covariance in Table 1 and 2. The implementations can be found at `https://github.com/CharlleChen/sp22-convex-final`. The optimization results can be found in Table 4. We can see that the reproduced results is pretty similar to that from the original paper.

4.3 Nonsmooth Optimization 1

In the paper [3], the authors mentioned that instead of using linear programming, nonsmooth optimization can be faster. The nonsmooth optimization in the paper is proposed by Uryas'ev in 1991 [4]. It includes two simultaneous working algorithms: the first one is finding the subgradient, while the second one adjusts the subgradient by a matrix to better optimize the loss function.

Recall the loss function for optimize is

$$\tilde{F}_\beta(x,\alpha) = \alpha + \frac{1}{q(1-\beta)}\sum_{k=1}^{q}[f(x,y_k)-\alpha]^+$$

According to [3], the gradient is

$$(0,1) + \frac{1}{q(1-\beta)}\sum_{k=1}^{q}\lambda_k(m-y_k,-1)$$

$$\text{with } \lambda_k = \begin{cases} 1, & \text{if } x^T(m-y_k)-\alpha > 0. \\ \text{between } 0,1, & \text{if } x^T(m-y_k)-\alpha = 0. \\ 0, & \text{if } x^T(m-y_k)-\alpha < 0. \end{cases}$$

Table 4: Results of the cvxpy (total runtime is 130.67s

β	Sample #	S&P	Bond	Small Cap	VaR	VaR Diff(%)	CVaR	CVaR Diff(%)
0.9	1000	0.62476	0.04917	0.32606	0.06910	1.85	0.09756	0.60
0.9	3000	0.45822	0.11319	0.42860	0.06968	2.70	0.09700	0.03
0.9	5000	0.40932	0.13198	0.45870	0.06704	-1.19	0.09799	1.04
0.9	10000	0.42474	0.12606	0.44921	0.06892	1.58	0.09868	1.76
0.9	20000	0.44981	0.11642	0.43377	0.06800	0.22	0.09680	-0.18
0.95	1000	0.59578	0.06031	0.34390	0.08912	-1.19	0.11620	0.25
0.95	3000	0.43811	0.12092	0.44097	0.08910	-1.22	0.11613	0.19
0.95	5000	0.37799	0.14402	0.47799	0.09099	0.87	0.11823	2.00
0.95	10000	0.38920	0.13971	0.47108	0.09260	2.66	0.11766	1.51
0.95	20000	0.46776	0.10952	0.42272	0.08992	-0.31	0.11564	-0.23
0.99	1000	0.53253	0.08463	0.38285	0.13936	5.47	0.15257	-0.26
0.99	3000	0.51356	0.09192	0.39452	0.13209	-0.03	0.15502	1.34
0.99	5000	0.24360	0.19568	0.56072	0.13576	2.75	0.15844	3.57
0.99	10000	0.48861	0.10150	0.40988	0.13296	0.63	0.15469	1.12
0.99	20000	0.42892	0.12445	0.44663	0.13423	1.59	0.15576	1.82

Since the nonsmooth optimization algorithm is essentially a gradient descent algorithm, which is unconstrained. We need to impose our constraints on it. Our constraints are

$$\begin{aligned}\sum x_k &= 1\\ x_k &\geq 0\\ x^T m &\geq R\end{aligned}$$

For the nonnegativity of x_k, we impose absolute value on x and use chain rule to propagate the gradient. In other words, when calculating F, we have

$$\tilde{F}_\beta(x,\alpha) = \alpha + \frac{1}{q(1-\beta)}\sum_{k=1}^{q}[f(|x|, y_k) - \alpha]^+$$

When calculate the gradient, we first calculate the gradient respect to $|x|$ and then we times it by the sign. i.e.

$$\Delta\tilde{F}(|x|,\alpha) * sgn(x)$$

For the normalization condition, we normalize the vector by its 1-norm on each step iteration.

$$x_{i+1} = x_i - \rho \Delta F$$
$$x_{i+1} = x_{i+1} / ||x_{i+1}||_1$$

For the constraint $x^T m \geq R$, which requires the portfolio to be somewhat profitable. We add a penalty term to the loss function and update its loss function accordingly so that the gradient descent will optimize this boundary condition as well.

$$\tilde{F}_\beta(x, \alpha) = \alpha + \frac{1}{q(1-\beta)} \sum_{k=1}^{q} [f(|x|, y_k) - \alpha]^+ + \mathit{penalty} \cdot [-x^T m + R]^+ \tag{2}$$

So the gradient of the loss function also updates.

$$(0, 1) + \frac{1}{q(1-\beta)} \sum_{k=1}^{q} \lambda_k (m - y_k, -1) - \mathit{penalty} \cdot \mathbb{1}_{-x^T m + R >= 0} \cdot m \tag{3}$$

$$\text{with } \lambda_k = \begin{cases} 1, & \text{if } x^T(m - y_k) - \alpha > 0. \\ \text{between } 0, 1, & \text{if } x^T(m - y_k) - \alpha = 0. \\ 0, & \text{if } x^T(m - y_k) - \alpha < 0. \end{cases} \tag{4}$$

Since x will be normalized, adding penalty according to the proportion of m doesn't guarantee $x^T m$ becomes larger than R, so in the gradient, we swap m with m^l, which increases the gap between different instruments. Empirically, $l = 2$ works fine. The value for penalty is chosen to be 10^3. Better values may exist.

These conversions methods from constrained to unconstrained may not be the optimal, but they produce some applicable optimized portfolio results. The result is shown in Table 5. Notice that there are some abnormal VaR values. CVaR is comparable to the previous method by linear programming, but is generally worse. This may be due to that the conversion from constrained optimization problem to unconstrained is not optimal. The authors of [3] probably employs better methods.

4.4 BFGS on nonsmooth optimization

Another promising algorithm on nonsmooth optimization is Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. BFGS empirically performs well on nonsmooth problems, but there are no general convergence theorem. Lewis and Overton give convergence guarantees on some specific scenario and argue that the morbid non-convergence cases are due to the ill-conditioned Hessian matrix [2]. They also propose an inexact line search in accompany with BFGS algorithms [2]. The implementations can be found at `https://cs.nyu.edu/~overton/software/hanso/`. However, due to time limit, we only experiment the standard

Table 5: Results of the nonsmooth optimization algorithms [4] (total runtime is 214.52s)

β	Sample #	S&P	Bond	Small Cap	VaR	VaR Diff(%)	CVaR	CVaR Diff(%)
0.9	1000	0.28453	0.17743	0.53804	0.06420	-5.37	0.09997	3.09
0.9	3000	0.27846	0.18209	0.53945	0.07003	3.22	0.09762	0.66
0.9	5000	0.28123	0.17876	0.54001	0.07043	3.80	0.09867	1.75
0.9	10000	0.28044	0.17994	0.53962	0.06668	-1.72	0.09929	2.39
0.9	20000	0.28091	0.18133	0.53775	0.05817	-14.27	0.09879	1.87
0.95	1000	0.28496	0.17647	0.53857	0.09299	3.09	0.11846	2.20
0.95	3000	0.27975	0.18014	0.54011	0.08709	-3.45	0.11701	0.95
0.95	5000	0.28006	0.17995	0.53998	0.08094	-10.27	0.12032	3.81
0.95	10000	0.27907	0.18018	0.54075	0.09056	0.40	0.11832	2.08
0.95	20000	0.28042	0.18153	0.53805	0.10413	15.44	0.11944	3.05
0.99	1000	0.28232	0.17763	0.54005	0.13321	0.82	0.15481	1.20
0.99	3000	0.28183	0.17772	0.54045	0.13488	2.08	0.15711	2.70
0.99	5000	0.26968	0.18556	0.54475	0.13590	2.85	0.15848	3.60
0.99	10000	0.27826	0.18038	0.54136	0.13739	3.98	0.15616	2.08
0.99	20000	0.27895	0.17960	0.54145	0.13519	2.32	0.15674	2.46

BFGS algorithm. In particular, we utilize `scipy.optimize.minimize(method='BFGS')` to optimize the problem.

We continue use the formula 2 as our optimization objective because it encodes constraint requirements. Moreover, since we cannot normalize x on each step (since the quasi-Newton method is happening within the function), we normalize x in the function definition. We still use penalty $= 10^3$. The result is shown in Table 6.

Still, it performs worse than the linear programming, but has a similar metrics as that by the other nonsmooth optimization algorithm (probably something wrong with the modified loss function 2).

Also, we notice that CVaR blows up for $\beta = 0.99$. It may be some cumulative numerical error according to [2].

4.5 Experiment Summary

Based on the formula 1 proposed from [3], we employ linear programming to directly solve the problem. Then, we try to optimize the efficiency by trying metric variable algorithms. To convert constrained optimization to unconstrained optimization, we impose additional penalty terms to the loss function. Gradient descent methods return valid solutions, but not as optimal as those given by linear programming. We also experiment BFGS algorithm. It performs decently well, though it diverges for some β. There are a lot of room for improvement:

Table 6: Results of the BFGS optimization algorithms (total runtime is 128.018s)

β	Sample #	S&P	Bond	Small Cap	VaR	VaR Diff(%)	CVaR	CVaR Diff(%)
0.9	1000	0.19313	0.21485	0.59201	0.07144	5.30	0.09943	2.53
0.9	3000	0.19309	0.21258	0.59433	0.07545	11.21	0.09887	1.95
0.9	5000	0.19265	0.21526	0.59209	0.06362	-6.22	0.09906	2.15
0.9	10000	0.19295	0.21515	0.59190	0.07075	4.28	0.09946	2.56
0.9	20000	0.19318	0.21506	0.59176	0.07145	5.31	0.09782	0.87
0.95	1000	0.18784	0.21711	0.59505	0.09573	6.13	0.11908	2.74
0.95	3000	0.18832	0.21693	0.59475	0.10071	11.65	0.11850	2.23
0.95	5000	0.18827	0.21695	0.59478	0.10229	13.41	0.12053	3.98
0.95	10000	0.18891	0.21670	0.59439	0.10257	13.71	0.11966	3.24
0.95	20000	0.18820	0.21697	0.59483	0.08422	-6.63	0.11769	1.54
0.99	1000	0.18278	0.21906	0.59816	0.59335	349.07	0.59335	287.87
0.99	3000	0.18430	0.21847	0.59723	0.58546	343.10	0.58546	282.71
0.99	5000	0.18312	0.21892	0.59795	0.56729	329.35	0.56729	270.83
0.99	10000	0.18251	0.21916	0.59833	0.57338	333.96	0.57338	274.81
0.99	20000	0.18312	0.21893	0.59796	0.57528	335.39	0.57528	276.06

our modified version 2 is clearly not the best conversion; also we can employ inexact line search [2] to BFGS algorithm.

Another potential improvement is on the formulation of the original 1. In this formulation, we bound the revenue as the constraint and minimize the risk. However, the other direction is more intuitive: within the given risk, we want to maximize the revenue. Since risk is almost always in contradiction with profit, these two statement should be essentially the same. And this is exactly the improvement proposed by [1], in which they reformulate the loss function so that we can bound risk and maximize revenue.

5 Discussion

Optimizing uncertainty and trade-off between risks and benefits are not within the field of finance, but almost everywhere. An example is recommendation system. When recommending content to users, the system can either choose "explore" by recommend some novel content to the users, which may or may not be liked by the user, but it can collect more information about the users; on the other hand, it can choose "exploit" known preference of users. This option is likely to be beneficial in the short term, but not necessary in the long term.

6 Summary

In this paper, we go through the fundamentals in risk evaluation and measurement, and then explore the capstone paper [3] on financial mathematics. In particular, it contributes to the optimization of risk management, including techniques like hedging. Then, we modify the formulation and run experiments with three different numerical optimization algorithms, which showcase different characteristics. The formulation is proved to powerful.

References

[1] Pavlo Krokhmal, Jonas Palmquist, and Stanislav Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of risk*, 4:43–68, 2002.

[2] Adrian S Lewis and Michael L Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1):135–163, 2013.

[3] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[4] SP Uryas' ev. New variable-metric algorithms for nondifferentiable optimization problems. *Journal of Optimization Theory and Applications*, 71(2):359–388, 1991.