Performance Evaluation of Iterative Methods for Solving Robust Convex Quadratic Problems

Christian Kroer ckroer@cs.cmu.edu
Computer Science Department, Carnegie Mellon University
Nam Ho-Nguyen hnh@andrew.cmu.edu
Tepper School of Business, Carnegie Mellon University
George Lu gclu@andrew.cmu.edu
Computer Science Department, Carnegie Mellon University
Fatma Kılınç-Karzan fkilinc@andrew.cmu.edu
Tepper School of Business, Carnegie Mellon University

Abstract

We investigate the practical performance of several recent iterative methods for solving robust convex optimization. In particular, we compare the performance of simple constraint generation to that of the more recent advances on incorporating regret minimization. We investigate the performance of these techniques on two robust convex quadratic problems: robust portfolio optimization and robust support vector machines (SVMs). For these problems, we identify how these recent iterative methods can be coupled with recent advances on solving trust region subproblem to achieve superior performance.

1 Introduction

Consider the general form of a robust convex optimization (RCO) problem:

$$\min_x \left\{ f(x) : \sup_{u^i \in U^i} f^i(x, u^i) \le 0,\ i \in [m], \quad x \in X \right\}, \tag{1}$$

where $X \subseteq \mathbb{R}^n$ is a convex domain, $U^1, \ldots, U^m$ are uncertainty sets, and $f, f^1, \ldots, f^m$ are all convex functions of $x \in X$ and $[m] = \{1, \ldots, m\}$. In traditional nonrobust optimization, the input data is a singleton u^i for each U^i and thus there is no supremum. Often, the data defining the nonrobust problem are uncertain or misspecified (only an approximation of the true data is available). In many applications, optimization with poorly instantiated data can have a large negative effect on performance. For example, in portfolio optimization, the covariance matrix is difficult to estimate, and the mean-variance model is known to be notoriously sensitive to these errors Goldfarb and Iyengar [2003]. To address this, several methodologies have been developed to handle the data uncertainty. Robust optimization (RO) addresses data uncertainty by seeking a solution $x \in X$ that is feasible for all data realizations u^i from the uncertainty sets U^i. RO has been extensively studied in the literature, and we refer the reader to the paper by Ben-Tal and Nemirovski Ben-Tal and Nemirovski [1998], the book by Ben-Tal et al. Ben-Tal et al. [2009] and surveys Ben-Tal and Nemirovski [2002, 2008], Bertsimas et al. [2011], Caramanis et al. [2012] for a detailed account of RO theory and its numerous applications.

The traditional solution method for RCO is based on reformulating it first into an equivalent deterministic robust counterpart problem via duality theory, and then solving the robust counterpart as a deterministic convex optimization problem. However, the robust counterpart approach often leads to larger and much less scalable problems than the associated nominal problem of (1) where the uncertain data (noise) $[u^1; \ldots; u^m]$ is fixed to a given value. For example, it is well-known that the robust counterpart of a convex quadratic program (QP) with ellipsoidal uncertainty is a semidefinite program Ben-Tal et al. [2009]. To bypass the robust counterpart approach and the challenges associated with it, recently several interesting *iterative methods* have been suggested for solving RO problems; see e.g., Mutapcic and Boyd [2009], Ben-Tal et al. [2015], Bertsimas et al. [2016], Ho-Nguyen and Kılınç-Karzan [2016]. These methods solve (1) by iteratively updating the solution x and the noise $[u^1; \ldots; u^m]$ to approximate their optimum values.

The iterative RCO methods of Mutapcic and Boyd [2009], Ben-Tal et al. [2015], Ho-Nguyen and Kılınç-Karzan [2016] build a solution $\bar{x}$ in similar ways by iteratively generating a *solution* sequence x_t and a *data* sequence u_t (for $t \ge 1$) that approximate the 'ideal' solution and data points respectively. A key feature

OPTML 2017: 10th NIPS Workshop on *Optimization for Machine Learning* (NIPS 2017).

in these approaches is that generating the next *solution* point x_t uses information from the *data* sequence $u_1, \ldots, u_{t-1}$ up to iteration $t-1$, and vice versa. Both Mutapcic and Boyd [2009] and Ben-Tal et al. [2015] in each iteration call a *nominal solution oracle* that involves solving full optimization problems in the same class as the underlying deterministic optimization problem of (1), whereas Ho-Nguyen and Kılınç-Karzan [2016] replaces these solution oracles via some intricate first-order updates. Mutapcic and Boyd [2009] and Ben-Tal et al. [2015] differ in terms of how the noise updates are carried out: Mutapcic and Boyd [2009] relies on a pessimization oracle whereas Ben-Tal et al. [2015] performs regret minimization (online learning) for this. While Ben-Tal et al. [2015], Ho-Nguyen and Kılınç-Karzan [2016] are shown to have a worst-case bound of $O(\frac{1}{\sqrt{T}})$ on their error after T steps, the cutting-set method of Mutapcic and Boyd [2009] has an exponential worst-case convergence rate. In spite of its theoretical convergence rate, Mutapcic and Boyd [2009] demonstrated that their cutting-set method requires few iterations in practice for random robust linear programs and two classes of robust convex QPs equipped with domain-specific exact, approximate, or heuristic pessimization oracles.

In this paper we perform detailed numerical experiments on the performance of the iterative algorithms from Mutapcic and Boyd [2009] and Ben-Tal et al. [2015]. In particular, we investigate a cutting-set algorithm equipped with a pessimization oracle like in Mutapcic and Boyd [2009] and a regret minimization approach as suggested in Ben-Tal et al. [2015]. Our focus is on solving robust convex QPs with ellipsoidal uncertainty sets arising in two different applications: robust portfolio optimization and robust SVMs. In this setting, given a feasible solution x, the pessimization problem of finding the worst-case uncertainty instantiation u is equivalent to the *trust-region subproblem* (TRS). We show that the cutting-set method combined with a TRS solver leads to fast practical performance; few cutting sets are required across all problems. We find that the regret minimization approach has much higher variance in its performance: for some problems it is fast; but occasionally it requires prohibitively many iterations and encounters numerical issues.

2 Iterative Algorithms for RCO

All algorithms considered in this paper are based on iteratively calling nominal solution oracles to solve variants of (1) where, instead of the supremum over U^i, we solve for a fixed finite set of constraints $\tilde{U}^i \subset U^i$. The exact form of $\tilde{U}^i$ changes across iterations of these algorithms. For the cutting-set method equipped with a pessimization solver, $\tilde{U}^i$ is an expanding set containing all worst-case instantiations of noise, with a new noise instantiation being added for every iteration that constraint i is violated by the current solution x_t. For the regret minimization approach $\tilde{U}^i$ consists of a singleton current noise instantiation chosen according to some online regret minimization scheme.

The pseudocode for the cutting-set (pessimization) method is given in Algorithm (1). The algorithm starts out with some initial set of uncertainty instantiations for each $i \in [m]$, usually a singleton set for each i. A loop then alternatingly 1) computes $\tilde{x}$, a solution to the current instantiation of (1), and 2) expands the uncertainty sets with worst-case noise instantiations generated by the pessimization solver against the current $\tilde{x}$. Following a standard analysis of cutting-plane methods, Mutapcic and Boyd [2009] prove that their approach is guaranteed to converge to a solution, although after an exponential number of steps. On the other hand, they also show that in practice far fewer iterations are needed, as is often the case with cutting-plane methods.

Algorithm 1 Cutting-set (pessimization) approach

```
input: Initial sets $\tilde{U}_0^i$, feasibility tolerance $\epsilon$
Repeat:
Set $\tilde{x}$ equal to a solution to (1) for $\tilde{U}_{t-1}$
If (1) is infeasible, then return "infeasible"
for $i = 1, \ldots, m$ do
    if $\max_{u^i \in U^i} f^i(\tilde{x}, u^i) > 0$ then
        $\tilde{U}_t^i = \tilde{U}_{t-1}^i \cup \{\arg\max_{u^i \in U^i} f^i(\tilde{x}, u^i)\}$
If $\max_{i, u^i \in U^i} f^i(\tilde{x}, u^i) < \epsilon$, then return $\tilde{x}$
```

Algorithm 2 Regret minimization approach

```
input: Initial sets $\tilde{U}_0^i$ , #iterations T, regret minimizer $RM$, initial solution $x_0$
for $t = 1, \ldots, T$ do
    for $i = 1, \ldots, m$ do
        $\tilde{U}_t^i = \{RM(x_{t-1}, u_{t-1})\}$
    Set $x_t$ equal to a solution to (1) for $\tilde{U}_t$
    If (1) is infeasible, then return "infeasible"
return $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$
```

The pseudocode for the regret minimization method is shown in Algorithm (2). The algorithm keeps each $\tilde{U}_t^i$ as a singleton throughout all iterations t. At iteration t, $\tilde{U}_t^i$ is updated to the regret-minimizer update returned by $RM(x_{t-1}, u_{t-1})$, and then x_t is updated to be the solution of (1) for $\tilde{U}_t^i$. Ben-Tal et al. [2015] show that, as long as the updates $u_t = RM(x_{t-1}, u_{t-1})$ satisfy the regret bounds $\max_{u^i \in U^i} \frac{1}{T}\sum_{t=1}^{T} f^i(x_t, u^i) - \frac{1}{T}\sum_{t=1}^{T} f^i(x_t, u_t^i) \leq \epsilon$, then the final solution $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$ is a 2ϵ-approximate solution to (1).

3 Numerical Results and Discussion

In our case, for the case of cutting-set approach, we modeled the resulting pessimization oracle via transforming the associated TRS into a convex QP as described in Appendix A.

For the regret minimization approach, we always check feasibility of both the average of the solutions $\tilde{x}$ obtained so far, as well as the current iterate $\tilde{x}$. Which is better depends on the instance class: for synthetic portfolio optimization and SVM the current iterate was almost always better, but for portfolio optimization on real stock data, the averaged iterate was better the majority of the time.

For the regret-minimizer RM in the regret minimization approach, we focus on the Follow the Approximate Perturbed Leader (FTAPL) algorithm. Note that FTAPL does not necessarily need the uncertainty set U^i to be convex. FTAPL requires the constraint functions to be linear in u_t^i, i.e., $f^i(x_t, u_t^i) = g^i(x_t)^\top u_t^i$, and updates u_t^i according to

$$\text{Find} \quad u_t^i \quad \text{s.t.} \quad \left(\sum_{s=1}^{t-1} g^i(x_s) + p_t\right)^\top u_t^i \geq \max_{u^i \in U^i} \left(\sum_{s=1}^{t-1} g^i(x_s) + p_t\right)^\top u^i - \epsilon,$$

where $p_t \in [0, 1/\eta]^{\dim(u^i)}$ is chosen uniformly at random. Ben-Tal et al. [2015] suggests a way to transform robust quadratic constraints into the correct form required by FTAPL. In the updates above, the parameter η is chosen depending on the structure of the problem; we refer to Ben-Tal et al. [2015] for details about choosing this as well as the regret bounds.

The experiments were run on a Macbook Pro with a 2.6Ghz Intel Core i7 processor and 16GB of RAM. All nominal convex QPs as well as all convexified TRSs were solved with Gurobi version 7.5.1. A timeout of 5 minutes was used. In our experiments, we consider two classes of robust convex QPs: one originating from a robust portfolio optimization and the other from a robust SVM.

Our robust portfolio optimization problem is based on the Markowitz mean-variance portfolio model with short sales and is given by the following problem formulation:

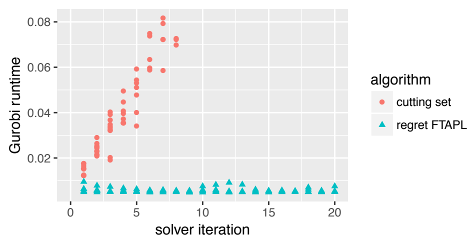
$$\min_x \left\{ \max_{\tilde{V} \in U_V} x^\top (\tilde{V}^\top F \tilde{V}) x + x^\top D x - \lambda \min_{\tilde{\mu} \in U_\mu} \tilde{\mu}^\top x : \ \mathbf{1}_n^\top x = 1 \right\},$$

where the uncertainty sets are defined as $U_V := \left\{ V_0 + \sum_{j \in [k]} P_j u_j : \ u \in \mathbb{R}^k, \ \|u\|_2 \leq 1 \right\}$ with fixed matrices $V_0, P_j \in \mathbb{R}^{m \times n}$. The first two terms represent risk: F is a covariance matrix specifying factors that affect returns and V is a matrix describing how these factors affect returns, D is a diagonal matrix specifying expected residual error. The third term represents the expected return of the portfolio. The parameter $\lambda > 0$ specifies the tradeoff between maximizing returns and minimizing risk. We discuss the derivation of this robust portfolio optimization model and the details of the associated instance generation in Appendix B. For robust portfolio optimization, we consider two types of instances, one based on synthetic data and one derived from real stock returns. To generate instances based on real data, we utilize data on 482 stocks that remained in the S&P 500 index for all 1258 trading days between 1 January 2003 and 31 December 2007.

Table 1 shows the synthetic robust portfolio optimization results for both algorithmic approaches across a number of different instances that vary the number of assets and factors in the model. Across all instance sizes the cutting-set approach performs very well: it never needs more than 8 iterations before converging to a solution. In contrast to this, the regret minimization approach frequently ends up requiring hundreds of iterations, and does not solve all instances for any of the problem sizes.

				# iterations			runtime (seconds)		
algorithm	assets (n)	factors	# solved	mean	min	max	mean	min	max
cutting set	20	8	10	3.70	2	5	0.02	0.01	0.04
cutting set	40	16	10	4.10	2	7	0.06	0.02	0.15
cutting set	80	32	10	5.80	3	8	0.37	0.12	0.62
regret FTAPL	20	8	8	89.00	12	330	1.02	0.16	3.55
regret FTAPL	40	16	9	100.89	4	199	3.18	0.12	6.26
regret FTAPL	80	32	6	121.33	80	204	12.03	8.16	20.98

Table 1: Summary statistics for synthetic robust portfolio optimization instances.



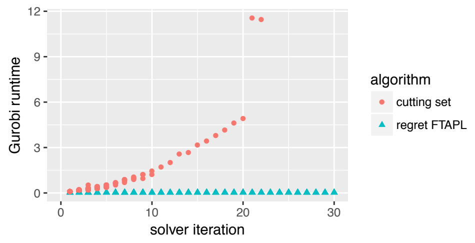


Figure 1: Runtimes for solution oracles. Left: portfolio optimization, Right: SVM.

Next we run experiments on real stock-market data. The results are shown in Table 2. The cutting-set solver solves all instances in 3 iterations or less, whereas the regret minimization approach requires significantly more.

				# iterations			runtime (seconds)		
algorithm	period length	assets (n)	# solved	mean	min	max	mean	min	max
cutting set	20	200	10	2.00	1	3	0.08	0.02	0.15
cutting set	20	482	20	1.55	1	2	0.27	0.06	0.46
cutting set	50	482	40	1.38	1	2	0.21	0.06	0.49
regret FTAPL	20	200	10	2.70	1	11	0.14	0.05	0.57
regret FTAPL	20	482	19	9.89	1	66	2.78	0.26	18.16
regret FTAPL	50	482	40	4.67	1	36	1.39	0.28	10.68

Table 2: Summary statistics for robust portfolio optimization on S&P500 stock data.

We next perform experiments on randomly generated robust SVM instances which leads to the following problem formulation:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \max_{u:\, \|u\|_2 \le 1} \alpha^\top Y^\top \Big(X_0 + \sum_{j \in [k]} P_j u_j \Big)^\top \Big(X_0 + \sum_{j \in [k]} P_j u_j \Big) Y - \sum_{i \in [m]} \alpha_i \\ \text{s.t.} \quad & \sum_{i \in [m]} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C \quad \forall i \in [m] \end{aligned}$$

We discuss the derivation of this robust SVM model and the details of the associated instance generation in Appendix C.

Table 3 shows summary statistics for solving robust SVM instances. As with the case of robust portfolio optimization, we find that the cutting-set approach solves all instances in a small number of iterations. We do find that the hardest instances for this approach require 20 to 40 iterations, as opposed to less than 10 for the robust portfolio optimization case. Similarly, we find that the robust SVM instances are significantly harder for the regret minimization approach: it can solve only about half of the smaller instances, and only a quarter of the largest instances.

				# iterations			runtime (seconds)		
algorithm	n	m	# solved	mean	min	max	mean	min	max
cutting set	10	30	40	6.22	2	19	0.09	0.01	0.60
cutting set	20	60	40	6.12	2	42	0.35	0.03	4.60
cutting set	80	240	40	4.22	2	22	3.47	0.53	67.56
regret FTAPL	10	30	21	378.10	1	3059	4.85	0.02	39.05
regret FTAPL	20	60	21	201.05	2	897	6.68	0.06	28.81
regret FTAPL	80	240	11	126.91	10	326	107.02	7.48	271.23

Table 3: Summary statistics for robust SVM instances.

Finally, we examine across iterations the per-solve runtime for the nominal solution oracles involved in these approaches. The results for both synthetic robust portfolio optimization (left) and robust SVM (right) are shown in Figure 1. As one might expect, the per-solve runtime of the nominal solution oracle for the cutting-set approach grows approximately linearly with the number of cuts added to the formulation of its nominal solution oracle. Contrary to this, the regret minimization approach has the same runtime for its nominal solution oracle across all iterations.

Acknowledgments. This research is supported in part by NSF grant CMMI 1454548. Christian Kroer is sponsored by a Facebook Fellowship.

References

A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4): 769–805, 1998. doi: 10.1287/moor.23.4.769. URL http://dx.doi.org/10.1287/moor.23.4.769.

A. Ben-Tal and A. Nemirovski. Robust optimization – methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.

A. Ben-Tal and A. Nemirovski. Selected topics in robust convex optimization. *Mathematical Programming*, 112(1):125–158, 2008.

A. Ben-Tal, L. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009. ISBN 9781400831050. URL https://books.google.com/books?id=DttjR7IpjUEC.

A. Ben-Tal, E. Hazan, T. Koren, and S. Mannor. Oracle-based robust optimization via online learning. *Operations Research*, 63(3):628–638, 2015. doi: 10.1287/opre.2015.1374. URL http://dx.doi.org/10.1287/opre.2015.1374.

D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011. doi: 10.1137/080734510. URL http://dx.doi.org/10.1137/080734510.

D. Bertsimas, I. Dunning, and M. Lubin. Reformulation versus cutting-planes for robust optimization. *Computational Management Science*, 13(2):195–217, Apr 2016. ISSN 1619-6988. doi: 10.1007/s10287-015-0236-z. URL https://doi.org/10.1007/s10287-015-0236-z.

C. Caramanis, S. Mannor, and H. Xu. Robust optimization in machine learning. In S. Sra, S. Nowozin, and S. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2012.

D. Goldfarb and G. Iyengar. Robust portfolio selection problems. *Mathematics of Operations Research*, 28 (1):1–38, 2003. doi: 10.1287/moor.28.1.1.14260.

N. Ho-Nguyen and F. Kılınç-Karzan. Online first-order framework for robust convex optimization. Technical report, July 2016. http://www.optimization-online.org/DB_HTML/2016/07/5555.html.

N. Ho-Nguyen and F. Kılınç-Karzan. A second-order cone based approach for solving the trust-region subproblem and its variants. *SIAM Journal on Optimization*, 27(3):1485–1512, 2017. doi: 10.1137/16M1065197.

A. Mutapcic and S. Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods and Software*, 24(3):381–406, June 2009. ISSN 1055-6788. doi: 10.1080/10556780802712889. URL http://dx.doi.org/10.1080/10556780802712889.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.