CSE 599: Interplay between Convex Optimization and Geometry **Winter 2018**

Lecture 2: Ellipsoid Method and Reductions Between Convex Oracles

Lecturer: Yin Tat Lee

Disclaimer: *Please tell me any mistake you noticed.*

2.1 Cutting Plane Methods

Given a continuously differentiable convex function f. Suppose that we computed $\nabla f(x)$. As we taught in last lecture (Theorem 1.4.6), we know that

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \text{ for all } y. \tag{2.1}$$

Let x^* be any minimizers of f. Putting $y = x^*$, we have that

$$f(x) \geq f(x^*) \geq f(x) + \langle \nabla f(x), x^* - x \rangle .$$

Therefore, we know that $\langle \nabla f(x), x^* - x \rangle \leq 0$. Namely, x^* lies on a half space with its normal $\nabla f(x)$. Therefore, each gradient computation cuts the set of possible solution into half. In $\mathbb{R}$, this allows us to do a binary search to minimize convex functions.

It turns out that in $\mathbb{R}^n$, such binary search still works. In this course, we will cover several ways to do this binary search. All of them follow the same framework, called cutting plane methods.

In this framework, we maintain a convex set $E^{(k)}$ that contains the minimizer x^* of f. Each iteration, we compute the gradient of f at a certain point $x^{(k)}$ depending on $E^{(k)}$. The convexity of f shows that any minimizers x^* lies in the half space

$$H^{(k)} \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n \text{ such that } \nabla f(x^{(k)})^T (x - x^{(k)}) \leq 0\} \tag{2.2}$$

and hence $x^* \in H^{(k)} \cap E^{(k)}$. The algorithm continues by choosing $E^{(k+1)}$ which contains $H^{(k)} \cap E^{(k)}$. The main questions are

1. What $E^{(k+1)}$ and $x^{(k)}$ does we choose?
2. How we measure the progress?
3. How fast is the convergent?
4. How expensive is each step?

Getting an algorithm that is fast in both theory and practice is still an active research area.

We first start by explaining the simplest algorithm, ellipsoid method.

Problem. Suppose we take $E^{(k+1)} = H^{(k)} \cap E^{(k)}$ every step and measure the progress by the volume of $E^{(k)}$. Then, we can define the "greedy method" which finds a point that decreases the volume as much as possible. For other purposes, [6] showed that this point can be approximated in polynomial time via sampling. Naturally, one may ask if this greedy method for convex optimization practical? More general, can we think this as a two player game and design some fast algorithms.

Year	$E^{(k)}$ and $x^{(k)}$	Rate	Cost/iter
1965 [15, 16]	Center of gravity	$1-\frac{1}{e}$	n^n
1979 [19, 17, 12]	Center of ellipsoid	$1-\frac{1}{2n}$	n^2
1988 [11]	Center of John ellipsoid	0.87	$n^{2.878}$
1989 [18]	Volumetric center	0.9999999	$n^{2.378}$
1995 [5]	Analytic center	0.9999999	$n^{2.378}$
2004 [22]	Center of gravity	$1-\frac{1}{e}$	n^6
2015 [21]	Hybrid center	$1-10^{-27}$	n^2 (amortized)

Table 2.1: Different Cutting Plane Methods

2.2 Ellipsoid Method

In ellipsoid method, each step we maintain an ellipsoid $E^{(k)} \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : (x-x^{(k)})^\top (A^{(k)})^{-1}(x-x^{(k)}) \le 1\}$ that contains all minimizers of f. After we compute $\nabla f(x^{(k)})$ and $H^{(k)}$ via (2.2), we define $E^{(k+1)}$ be the smallest volume ellipsoid containing $E^{(k)} \cap H^{(k)}$. The key observation is that the volume of the ellipsoid $E^{(k)}$ decrease by roughly $1-\frac{1}{2n}$ every iteration.

Lemma 2.2.1. *We have that*

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \frac{1}{n+1} A^{(k)} \widehat{g}, \\ A^{(k+1)} &= \frac{n^2}{n^2-1}\left(A^{(k)} - \frac{2}{n+1} A^{(k)} \widehat{g}\widehat{g}^\top A^{(k)}\right), \\ \widehat{g} &= \frac{1}{\sqrt{\nabla f(x^{(k)})^\top A^{(k)} \nabla f(x^{(k)})}} \nabla f(x^{(k)}). \end{aligned}$$

Moreover, we have that $\mathrm{vol} E^{(k+1)} < e^{-\frac{1}{2n}} \mathrm{vol} E^{(k)}$.

Proof. Since volume of all ellipsoids changed by same factor under any affine transformation, we can assume the $A^{(k)} = I$, $x^{(k)} = 0$ and $\widehat{g} = e_1$. The rest is a calculus exercise. □

Problem. Note that the ellipsoid method looks exactly like quasi-Newton method with the Hessian A_k^{-1}. Does this suggest a natural combination of quasi-Newton method and ellipsoid method that is polynomial time in worst case and efficient in practice?

Lemma 2.2.1 shows that the volume decreases by a constant factor every n steps. It in turns implies that we can minimize any convex function with ε additive error in $O(n^2 \log(1/\varepsilon))$ iterations. To make the statement more general, we note that ellipsoid method can be used in non-differentiable functions.

Theorem 2.2.2. *Let f be a convex function on $\mathbb{R}^n$, $E^{(0)}$ be any initial ellipsoid and $\Omega \subset E^{(0)}$ be any convex set. Suppose that for any $x \in E^{(0)}$, we can find, in time $\mathcal{T}$, a direction $g(x)$ such that*

$$f(y) \ge f(x) \text{ for any } y \text{ such that } g(x)^\top (y-x) \ge 0.$$

Let $M = \mathrm{vol}(E^{(0)})/\mathrm{vol}(\Omega)$ and $x^{(i)}$ be the sequence of points produced by ellipsoid method. Then, we have

$$\min_{i=1,2,\cdots k} f(x^{(i)}) - \min_{y\in\Omega} f(y) \le M^{\frac{1}{n}} \exp(-\frac{k}{2n^2}) \left(\max_{z\in\Omega} f(z) - \min_{x\in\Omega} f(x)\right) .$$

Furthermore, each iteration takes $O(n^2 + \mathcal{T})$ time.

Remark. If f is continuously differentiable, then $g(x) = \nabla f(x)$.

Proof. We run ellipsoid method (with the direction $g(x)$ instead of the gradient $\nabla f(x)$).

Let x^* be any minimizer of f over Ω. Note that after k iteration, $\mathrm{vol}(E^{(k)}) < \exp(-\frac{k}{2n})\mathrm{vol}(E^{(0)})$. Let $\alpha = M^{\frac{1}{n}}\exp(-\frac{k}{2n^2})$ and $S = x^* + \alpha(\Omega - x^*)$, namely, $S = \{x^* + \alpha(z - x^*) : z \in \Omega\}$. Then, we have

$$\mathrm{vol}(S) = \alpha^n \mathrm{vol}(\Omega) = \exp(-\frac{k}{2n})M \cdot \mathrm{vol}(\Omega) = \exp(-\frac{k}{2n})\mathrm{vol}(E^{(0)}) > \mathrm{vol}(E^{(k)}).$$

Therefore, S is not a subset of $E^{(k)}$ and hence there is a point $y \in S \backslash E^{(k)}$. Therefore, for some $i \le k$, we have

$$g(x^{(i)})^T(y - x^{(i)}) > 0$$

and by the assumption of g, we have

$$f(y) \ge f(x^{(i)}).$$

Since $y \in S$, we have $y = (1-\alpha)x^* + \alpha z$ for some $z \in \Omega$. Thus, the convexity of f implies that

$$f(y) \le (1-\alpha)f(x^*) + \alpha f(z).$$

Therefore, we have

$$\min_{i=1,2,\cdots k} f(x^{(i)}) - \min_{x\in\Omega} f(x) \le f(y) - f(x^*) \le \alpha\left(\max_{z\in\Omega} f(z) - \min_{x\in\Omega} f(x)\right).$$

□

This theorem can be used to solve many problems in polynomial time. As an illustration, we show how to solve linear programs in polynomial time here. For the linear program $\min_{Ax\ge b} c^\top x$, the function we want to minimize is

$$L(x) = c^\top x + \ell_{Ax\ge b}. \tag{2.1}$$

For the function L (2.1), we can use $g(x) = c$ if $Ax \ge b$ and $g(x) = a_i$ if $a_i^\top x < b$.

Let $\Omega = \{Ax \ge b\}$ and R be the "diameter" of Ω defined by $R \stackrel{\mathrm{def}}{=} \max_{x\in\Omega}\|x\|^2$. We can simply pick $E^{(0)}$ be a unit ball centered at 0 with radius R and apply Theorem 2.2.2. to find x such that

$$f(x) - \min_{x\in\Omega} f(x) \le \varepsilon(\max_{x\in\Omega} c^\top x - \min_{x\in\Omega} c^\top x)$$

in time $O(n^2(n^2 + \mathrm{nnz}(A))\log(\frac{R}{\mathrm{vol}(\Omega)^{1/n}\varepsilon}))$. To get the solution exactly, i.e., $\varepsilon = 0$, we need to assume the linear program is integral and the running time depends on the bound of the number in the matrix A and the vectors b and c. It is still open how to solve linear program with time polynomial only to the number of variables. We call such running time strongly polynomial.

Problem. How to solve linear programs in strongly polynomial time?

2.3 Convex Conjugate

Computing the gradient of a function is one way to access the function. However, it is not always a simple chain rule exercise. For example, separation for the convex hull of points is pretty difficult to compute; it involves solving linear programs.

Definition 2.3.1. $\mathrm{conv}(X)$ is a convex hull of X if $\mathrm{conv}(X) = \{\sum \alpha_i x_i : \ \alpha_i \ge 0, \sum \alpha_i = 1, x_i \in X\}$.

In this case, computing gradient may not be the ideal way to access the function. However, we note that the following problem is very easy

$$h(\theta) \stackrel{\text{def}}{=} \max_{x \in \text{conv}(\{a_i\})} \theta^\top x.$$

This can be computed by checking which $a_i^\top x$ is the largest. From the last lecture (Corollary 1.3.3), we see that we can reconstruct the convex set from $h(\theta)$. Therefore, this is a feasible way to access the convex set. This can be generalized to convex functions as follows:

Definition 2.3.2. For any convex function f, we define the convex conjugate

$$f^*(\theta) \stackrel{\text{def}}{=} \max_x \theta^\top x - f(x).$$

The following lemma shows that indeed one can recover f from f^*.

Lemma 2.3.3 (Involution property)**.** *For any convex function f, we have that $f^{**} = f$.*

Proof. As we know (Corollary 1.3.3), every convex set is the intersection of half spaces. By the reduction via epigraph, every convex function is the supremum of affine functions. Note that "$f(x) \geq \theta^\top x - b$ for all x" is same as "$b \geq \theta^\top x - f(x)$ for all x". Therefore, for a fixed θ, the supremum of these affine functions is $\theta^\top x - f^*(\theta)$. Therefore, we have that

$$f(x) = \max_\theta \theta^\top x - f^*(\theta).$$

□

Just to give some intuition. Recall that for any linear space X, we have that $X^{**} = X$. Therefore, there are two natural "coordinate systems" to record a convex function, the primal space X and the dual space X^*. Under these "coordinate systems", we have f and f^*.

Exercise 2.3.4 (Order reversion)**.** $f \geq g \iff f^* \leq g^*$.

Interestingly, convex conjugate is the unique transformation on convex functions that satisfies involution, and orders reversion.

Theorem 2.3.5 ([4])**.** *Given a transformation $\mathcal{T}$ that maps lower-semi-continuous convex functions onto itself such that $\mathcal{T}\mathcal{T}\phi = \phi$ and $\phi \leq \psi \implies \mathcal{T}\phi \geq \mathcal{T}\psi$ for all lower-semi-continuous convex functions ϕ and ψ. Then, $\mathcal{T}$ is essentially convex conjugate, namely, there is an invertible symmetric linear transformation B, a vector v_0 and a constant C_0 such that*

$$(\mathcal{T}\phi)(x) = \phi^*(Bx + v_0) + v_0^\top x + C_0.$$

After defining the dual, the first question again is to compute its gradient.

Lemma 2.3.6. *For any continuously differentiable function f, we have that $\nabla f^*(\theta) = \arg\sup_x \theta^\top x - f(x)$.*

Proof. Let $x^* = \arg\sup_x \theta^\top x - f(x)$. By definition, we have that $f^*(\theta) = \theta^\top x^* - f(x^*)$ and that $f^*(\eta) \geq \eta^\top x^* - f^*(x^*)$ for all η. Therefore, we have that

$$f^*(\eta) \geq f^*(\theta) + {x^*}^\top(\eta - \theta) \text{ for all } \eta.$$

Therefore, we have that $\nabla f^*(\theta) = x^*$. □

This lemma says that we can compute the gradient of f^* by solving the convex optimization problem $\min_x f(x) - \theta^\top x$. Therefore, Theorem 2.2.2 basically says that we can compute ∇f^* by computing n many ∇f. Due to the involution (Lemma 2.3.3), computing ∇f can also be reduced to computing ∇f^*.

Going back to the example about $\mathrm{conv}(\{a_i\})$, since we know how to compute $\max_{x\in\mathrm{conv}(\{a_i\})} \theta^\top x = \ell^*_{\mathrm{conv}(\{a_i\})}$, this reduction gives us a way to do separation on $\mathrm{conv}(\{a_i\})$, equivalently, to compute the subgradient of $\ell^*_{\mathrm{conv}(\{a_i\})}$. In combinatorial optimization, many convex sets are given by the convex hull for some discrete objects. In many cases, the only known way to do the separation is via such reductions. Now, we will discuss the implication of such reduction, or other similar reduction for convex sets more.

2.4 4 oracles for convex sets

In the seminar work by Grötschel, Lovasz and Schrijver [10], they introduced four different ways to access convex sets, showed they are equivalent and used it to get the first polynomial time method for many combinatorial problems.

Instead of introducing that four oracles for convex set directly, it is easier to first introduce two oracles for convex functions. The first oracle is what we have been talking about all the time, just more formally.

Definition 2.4.1 (Subgradient Oracle (GRAD))**.** Queried with a vector y with $\|y\|_2 \le 1$ and real numbers $\delta > 0$, the oracle outputs an extended real number α satisfying (2.2) and a vector $c \in \mathbb{R}^n$ such that

$$\alpha + c^T(x - y) < \max_{z\in B(x,\delta)} f(z) + \delta \text{ for all } x \in \mathbb{R}^n \tag{2.1}$$

The restriction $\|y\|_2 \le 1$ is arbitrary and 1 can be any number. However, without such formal restriction, we cannot even compute the gradient of x^2 with finite number of precision. The another oracle is just a strictly weaker oracle:

Definition 2.4.2 (Evaluation Oracle (EVAL))**.** Queried with a vector y with $\|y\|_2 \le 1$ and real number $\delta > 0$ the oracle finds an extended real number α such that

$$\min_{x\in B(y,\delta)} f(x) - \delta \le \alpha \le \max_{x\in B(y,\delta)} f(x) + \delta. \tag{2.2}$$

For any convex set K, we have the following four oracles, GRAD(ℓ_K), EVAL(ℓ_K), GRAD(ℓ^*_K), EVAL(ℓ^*_K). Roughly speaking, they corresponding to finding a separating hyperplane, checking membership, finding an extreme point and computing the width with respect to some direction.

In this course, we will not talk about convex optimization under noisy oracle. However, I want to emphasize that the question about minimizing a convex function under noisy evaluation oracle is an active research area and the gaps between the lower bound and upper bound for many problems are still huge [7, 9].

Problem. Given an approximately convex function F on unit ball such that $\max_{\|x\|_2\le 1} |f(x) - F(x)| \le \varepsilon/n$ for some convex function f. How fast can we find x on the unit ball such that $F(x) \le \min_{\|x\|_2\le 1} F(x) + O(\varepsilon)$? The current fastest algorithm takes $O(n^{4.5}\log^{O(1)}(n/\varepsilon))$ oracle call to F.

However, if we can compute evaluation oracle accurately, then we can compute the gradient via $O(n)$ many calls to evaluation oracle. When the function is smooth, we can simply approximate the gradient of a function by calculating a finite difference

$$\frac{\partial f}{\partial x_i} = \frac{f(x + he_i) - f(x)}{h} + O(h)$$

which only takes $n + 1$ calls to the evaluation oracle. The only issue is that convex function may not be differentiable. However, any convex Lipschitz function are twice differentiable almost everywhere (see the proof at the end). Therefore, we simply need to perturb x by a random noise, then apply a finite difference.

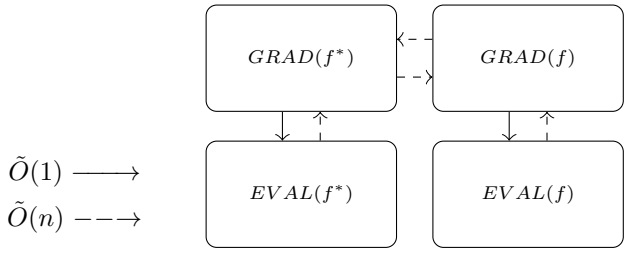

Figure 2.1: This illustrates the relationships of oracles for a convex function f and its convex conjugate f^*.

Figure 2.1 shows the current best reduction between these four oracles. It is easy to show that you need at least n calls to evaluation oracle to compute one gradient and that you need n calls to gradient oracle of f to compute one gradient of f^*. However, the following problem is still open:

Problem. Prove that it takes $\Omega(n^2)$ calls to the evaluation oracle of f to compute the gradient of f^*.

Lemma 2.4.3 ([13])**.** *Let $B_\infty(x,r) = \{y : \|x-y\|_\infty \le r\}$. For any $0 < r_2 \le r_1$ and any convex function f defined on $B_\infty(x, r_1+r_2)$ with $\|\nabla f(z)\|_\infty \le L$ for any $z \in B_\infty(x, r_1+r_2)$ we have*

$$\mathbb{E}_{y\in B_\infty(x,r_1)}\mathbb{E}_{z\in B_\infty(y,r_2)} \|\nabla f(z) - g(y)\|_1 \le n^{3/2}\frac{r_2}{r_1}L$$

where $g(y)$ is the average of ∇f over $B_\infty(y,r_2)$.

Proof. Let $\omega_i(z) = \langle \nabla f(z) - g(y), e_i\rangle$ for all $i \in [n]$. Then, we have that

$$\int_{B_\infty(y,r_2)} \|\nabla f(z) - g(y)\|_1 \, dz \le \sum_i \int_{B_\infty(y,r_2)} |\omega_i(z)| \, dz.$$

Since $\int_{B_\infty(y,r_2)} \omega_i(z)dz = 0$, the Poincare inequality for a box shows that

$$\begin{aligned}\int_{B_\infty(y,r_2)} |\omega_i(z)| \, dz &\le r_2 \int_{B_\infty(y,r_2)} \|\nabla\omega_i(z)\|_2 \, dz \\ &= r_2 \int_{B_\infty(y,r_2)} \left\|\nabla^2 f(z)e_i\right\|_2 dz \\ &= \sqrt{n}r_2 \int_{B_\infty(y,r_2)} \left\|\nabla^2 f(z)\right\|_F dz\end{aligned}$$

Now, using f is convex, we have that $\left\|\nabla^2 f(z)\right\|_F \le \mathrm{tr}\nabla^2 f(z) = \Delta f(z)$. Therefore, we have

$$\begin{aligned}\mathbb{E}_{z\in B_\infty(y,r_2)} \|\nabla f(z) - g(y)\|_1 \, dz &\le \sqrt{n}r_2\mathbb{E}_{z\in B_\infty(y,r_2)}\Delta f(z)dz \\ &= \sqrt{n}r_2\Delta h(y)\end{aligned}$$

where $h = \frac{1}{(2r_2)^n} f * \chi_{B_\infty(0,r_2)}$ where $\chi_{B_\infty(0,r_2)}$ is 1 on the set $B_\infty(0,r_2)$ and 0 on outside.

Integrating by parts, we have that

$$\int_{B_\infty(x,r_1)} \Delta h(y)dy = \int_{\partial B_\infty(x,r_1)} \langle \nabla h(y), n(y)\rangle \, dy$$

where $\Delta h(y) = \sum_i \frac{d^2h}{dx_i^2}(y)$ and $n(y)$ is the normal vector on $\partial B_\infty(x,r_1)$ the boundary of the box $B_\infty(x,r_1)$, i.e. standard basis vectors. Since f is L-Lipschitz with respect to $\|\cdot\|_\infty$ so is h, i.e. $\|\nabla h(z)\|_\infty \le L$. Hence, we have that

$$\mathbb{E}_{y\in B_\infty(x,r_1)}\Delta h(y) \le \frac{1}{(2r_1)^n}\int_{\partial B_\infty(x,r_1)} \|\nabla h(y)\|_\infty \|n(y)\|_1 \, dy \le \frac{1}{(2r_1)^n}\cdot 2n(2r_1)^{n-1}\cdot L = \frac{nL}{r_1}.$$

Therefore, we have that

$$\mathbb{E}_{y\in B_\infty(x,r_1)}\mathbb{E}_{z\in B_\infty(y,r_2)}\left\|\nabla f(z)-g(y)\right\|_1 dz \le n^{3/2}\frac{r_2}{r_1}L.$$

□

References

[4] Shiri Artstein-Avidan and Vitali Milman. The concept of duality in convex analysis, and the characterization of the legendre transform. *Annals of mathematics*, pages 661–674, 2009.

[5] David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

[6] Amitabh Basu and Timm Oertel. Centerpoints: A link between optimization and convex geometry. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 14–25. Springer, 2016.

[7] Alexandre Belloni, Tengyuan Liang, Hariharan Narayanan, and Alexander Rakhlin. Escaping the local minima via simulated annealing: Optimization of approximately convex functions. In *Conference on Learning Theory*, pages 240–265, 2015.

[8] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM (JACM)*, 51(4):540–556, 2004.

[9] Sébastien Bubeck, Ronen Eldan, and Yin Tat Lee. Kernel-based methods for bandit convex optimization. *arXiv preprint arXiv:1607.03084*, 2016.

[10] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Algorithms and Combinatorics, 1988.

[11] L Khachiyan, S Tarasov, and E Erlich. The inscribed ellipsoid method. In *Soviet Math. Dokl*, volume 298, 1988.

[12] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[13] Yin Tat Lee, Aaron Sidford, and Santosh S Vempala. Efficient convex optimization with membership oracles. *arXiv preprint arXiv:1706.07357*, 2017.

[14] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1049–1065. IEEE, 2015.

[15] A Yu Levin. On an algorithm for the minimization of convex functions. In *Soviet Mathematics Doklady*, volume 160, pages 1244–1247, 1965.

[16] Donald J Newman. Location of the maximum on unimodal surfaces. *Journal of the ACM (JACM)*, 12(3):395–398, 1965.

[17] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, 13(1):94–96, 1977.

[18] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 338–343. IEEE, 1989.

[19] David B Yudin and Arkadii S Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12:128–142, 1976.