Localization and Cutting-plane Methods

S. Boyd and L. Vandenberghe

September 18, 2003

In this chapter we describe a class of methods for solving general convex and quasiconvex optimization problems, based on the use of *cutting-planes*, which are hyperplanes that separate the current point from the optimal points. These methods, called *cutting-plane methods* or *localization methods*, are quite different from interior-point methods, such as the barrier method or primal-dual interior-point method described in chapter 11 of Boyd and Vandenberghe. Cutting-plane methods are usually less efficient for problems to which interior-point methods apply, but they have a number of advantages that can make them an attractive choice in certain situations.

- Cutting-plane methods do not require differentiability of the objective and constraint functions, and can directly handle quasiconvex as well as convex problems.
- Cutting-plane methods can exploit certain types of structure in large and complex problems. A cutting-plane method that exploits structure can be faster than a general-purpose interior-point method for the same problem.
- Cutting-plane methods do not require evaluation of the objective and all the constraint functions at each iteration. (In contrast, interior-point methods require evaluating all the objective and constraint functions, as well as their first and second derivatives.) This can make cutting-plane methods useful for problems with a very large number of constraints.
- Cutting-plane methods can be used to decompose problems into smaller problems that can be solved sequentially or in parallel.

1 Cutting-planes

1.1 Cutting-plane oracle

The goal of cutting-plane and localization methods is to find a point in a convex set $X \subseteq \mathbf{R}^n$, which we call the *target set*, or to determine that X is empty. In an optimization problem, the target set X can be taken as the set of optimal (or ϵ-suboptimal) points for the problem, and our goal is find an optimal (or ϵ-suboptimal) point for the optimization problem.

We do not have direct access to any description of the target set X (such as the objective and constraint functions in an underlying optimization problem) except through an *oracle*.

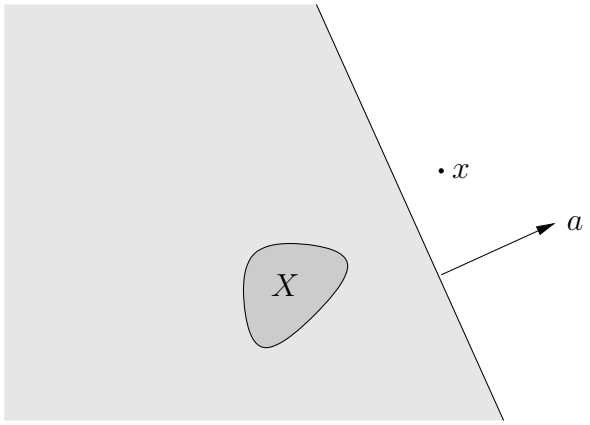

Figure 1: The inequality $a^T z \leq b$ defines a cutting-plane at the query point x, for the target set X, shown shaded. To find a point in the target set X we need only search in the lightly shaded halfspace; the unshaded halfspace $\{z \mid a^T z > b\}$ cannot contain any point in the target set.

When we query the oracle at a point $x \in \mathbf{R}^n$, the oracle returns the following information to us: it either tells us that $x \in X$ (in which case we are done), or it returns a separating hyperplane between x and X, *i.e.*, $a \neq 0$ and b such that

$$a^T z \leq b \text{ for } z \in X, \qquad a^T x \geq b.$$

This hyperplane is called a *cutting-plane*, or *cut*, since it 'cuts' or eliminates the halfspace $\{z \mid a^T z > b\}$ from our search; no such point could be in the target set X. This is illustrated in figure 1. We call the oracle that generates a cutting-plane at x (or the message that $x \in X$) a *cutting-plane oracle*. We can assume $\|a_k\|_2 = 1$, since dividing a and b by $\|a\|_2$ defines the same cutting-plane.

When the cutting-plane $a^T z = b$ contains the query point x, we refer to it as a *neutral cut* or *neutral cutting-plane*. When $a^T x > b$, which means that x lies in the interior of the halfspace that is being cut from consideration, the cutting-plane is called a *deep cut*. Figure (2) illustrates a neutral and deep cut. Intuition suggests that a deep cut is better, *i.e.*, more informative, than a neutral cut (with the same normal vector a), since it excludes a larger set of points from consideration.

1.2 Finding cutting-planes

We will have much to say (in §4 and §5) about how to generate cutting-planes for a variety of problems, but for now, we show how to find cutting-planes for a convex optimization problem in standard form, with differentiable objective and constraint functions. We take the target set X to be the optimal set for the problem, so the oracle must either declare the point x optimal, or produce a hyperplane that separates x from the optimal set. It is straightforward to include equality constraints, so we leave them out to simplify the exposition.

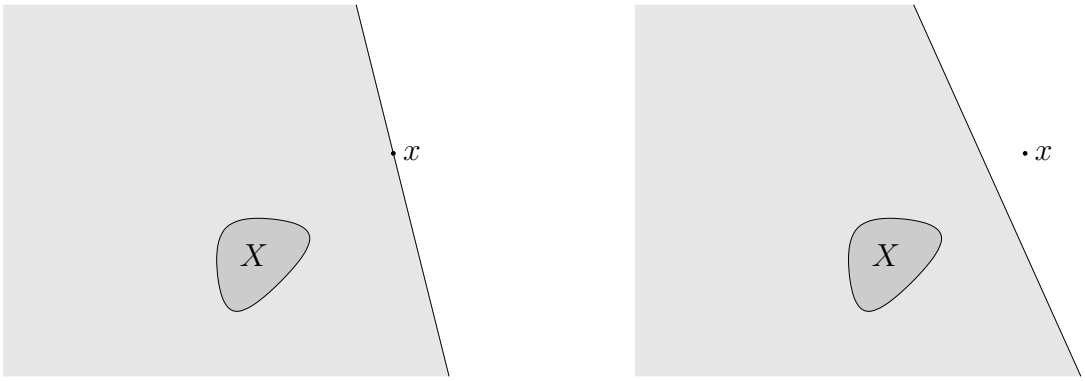

Figure 2: *Left:* a neutral cut for the point x and target set X. Here, the query point x is on the boundary of the excluded halfspace. *Right:* a deep cut for the point x and target set X.

1.2.1 Unconstrained convex problem

We first consider the unconstrained optimization problem

$$\text{minimize} \quad f_0(x), \tag{1}$$

where f_0 is convex and, for now, differentiable. To find a cutting-plane for this problem, at the point x, we proceed as follows. First, if $\nabla f_0(x) = 0$, then x is optimal, *i.e.*, in the target set X. So we assume that $\nabla f_0(x) \neq 0$. Recall that for all z we have

$$f_0(z) \geq f_0(x) + \nabla f_0(x)^T(z - x),$$

since f_0 is convex. Therefore if z satisfies $\nabla f_0(x)^T(z - x) > 0$, then it also satisfies $f_0(z) > f_0(x)$, and so cannot be optimal (*i.e.*, in X). In other words, we have

$$\nabla f_0(x)^T(z - x) \leq 0 \text{ for } z \in X,$$

and $\nabla f_0(x)^T(z - x) = 0$ for $z = x$. This shows that

$$\nabla f_0(x)^T(z - x) \leq 0$$

is a (neutral) cutting-plane for (1) at x.

This cutting-plane has a simple interpretation: in our search for an optimal point, we can remove the halfspace $\{z \mid \nabla f_0(x)^T(z - x) > 0\}$ from consideration because all points in it have an objective value larger than the point x, and therefore cannot be optimal. This is illustrated in figure 3.

The inequality $\nabla f_0(x)^T(z - x) \leq 0$ also serves as a cutting-plane when the objective function f_0 is quasiconvex, provided $\nabla f_0(x) \neq 0$, since

$$\nabla f_0(x)^T(z - x) \geq 0 \implies f(z) \geq f(x)$$

(see §**??**).

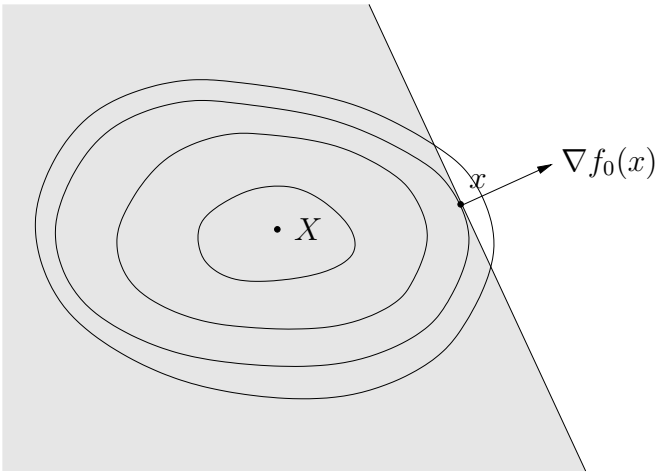

Figure 3: The curves show the level sets of a convex function f_0. In this example the optimal set X is a singleton, the minimizer of f_0. The hyperplane given by $\nabla f_0(x)^T(z-x) = 0$ separates the point x (which lies on the hyperplane) from the optimal set X, hence defines a (neutral) cutting-plane. All points in the unshaded halfspace can be 'cut' since in that halfspace we have $f_0(z) \geq f_0(x)$.

1.2.2 Convex feasibility problem

We consider the feasibility problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array}$$

where f_i are convex and, for now, differentiable. Here the target set X is the feasible set.

To find a cutting-plane for this problem at the point x we proceed as follows. If x is feasible, *i.e.*, satisfies $f_i(x) \leq 0$ for $i = 1, \ldots, m$, then $x \in X$. Now suppose x is not feasible. This means that there is at least one index j for which $f_j(x) > 0$, *i.e.*, x violates the jth constraint. From the inequality

$$f_j(z) \geq f_j(x) + \nabla f_j(x)^T(z-x),$$

we conclude that if

$$f_j(x) + \nabla f_j(x)^T(z-x) \geq 0,$$

then $f_j(z) > 0$, and so z also violates the jth constraint. It follows that any feasible z satisfies the inequality

$$f_j(x) + \nabla f_j(x)^T(z-x) \leq 0,$$

which gives us the required cutting-plane. Since $f_j(x) > 0$, this is a deep cutting-plane.

Here we remove from consideration the halfspace defined by $f_j(x) + \nabla f_j(x)^T(z-x) \geq 0$ because all points in it violate the jth inequality, as x does, hence are infeasible. This is illustrated in figure 4.

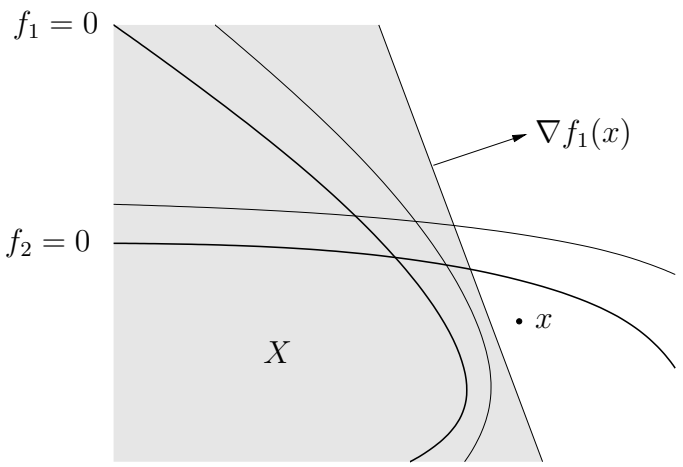

Figure 4: The curves show level sets of two convex functions f_1, f_2; the darker curves show the level sets $f_1 = 0$ and $f_2 = 0$. The feasible set X, defined by $f_1 \leq 0$, $f_2 \leq 0$, is at lower left. At the point x, the constraint $f_1(x) \leq 0$ is violated, and the hyperplane $f_1(x) + \nabla f_1(x)^T(z - x) = 0$ defines a deep cut.

1.2.3 Standard convex problem

By combining the methods described above, we can find a cutting-plane for the standard problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{2}$$

where $f_0, \ldots, f_m$ are convex and, for now, differentiable. As above, the target set X is the optimal set.

Given the query point x, we first check for feasibility. If x is not feasible, then we can construct a cut as

$$f_j(x) + \nabla f_j(x)^T(z - x) \leq 0, \tag{3}$$

where $f_j(x) > 0$ (*i.e.*, j is the index of any violated constraint). This defines a cutting-plane for the problem (2) since any optimal point must satisfy the jth inequality, and therefore the linear inequality (3). The cut (3) is called *feasibility cut* for the problem (2), since we are cutting away a halfplane of points known to be infeasible (since they violate the jth constraint).

Now suppose that the query point x is feasible. If $\nabla f_0(x) = 0$, then x is optimal and we are done. So we assume that $\nabla f_0(x) \neq 0$. In this case we can construct a cutting-plane as

$$\nabla f_0(x)^T(z - x) \leq 0,$$

which we refer to as an *objective cut* for the problem (2). Here, we are cutting out the halfspace $\{z \mid \nabla f_0(x)^T(z - x) > 0\}$ because we know that all such points have an objective value larger than x, hence cannot be optimal.

2 Localization algorithms

2.1 Basic cutting-plane and localization algorithm

We start with a set of initial linear inequalities

$$A_0 z \preceq b_0,$$

where $A_0 \in \mathbf{R}^{q\times n}$, that are known to be satisfied by any point in the target set X. One common choice for this initial set of inequalities is the ℓ_∞-norm ball of radius R, *i.e.*,

$$-R \leq z_i \leq R, \quad i = 1, \ldots, n,$$

where R is chosen large enough to contain X. At this point we know nothing more than

$$X \subseteq \mathcal{P}_0 = \{z \mid A_0 z \preceq b_0\}.$$

Now suppose we have queried the oracle at points $x^{(1)}, \ldots, x^{(k)}$, none of which were announced by the oracle to be in the target set X. Then we have k cutting-planes

$$a_i^T z \leq b_i, \quad i = 1, \ldots k,$$

that separate $x^{(k)}$ from X, respectively. Since every point in the target set must satisfy these inequalities, we know that

$$X \subseteq \mathcal{P}_k = \{z \mid A_0 z \preceq b_0,\ a_i^T z \leq b_i,\ i = 1, \ldots k\}.$$

In other words, we have *localized* X to within the polyhedron $\mathcal{P}_k$. In our search for a point in X, we need only consider points in the *localization polyhedron* $\mathcal{P}_k$. This is illustrated in figure 5.

If $\mathcal{P}_k$ is empty, then we have a proof that the target set X is empty. If it is not, we choose a new point $x^{(k+1)}$ at which to query the cutting-plane oracle. (There is no reason to choose $x^{(k+1)}$ outside $\mathcal{P}_k$, since we know that all target points lie in $\mathcal{P}_k$.) If the cutting-plane oracle announces that $x^{(k+1)} \in X$, we are done. If not, the cutting-plane oracle returns a new cutting-plane, and we can update the localization polyhedron by adding the new inequality. This iteration gives the basic cutting-plane or localization algorithm:

Basic conceptual cutting-plane/localization algorithm

given an initial polyhedron $\mathcal{P}_0$ known to contain X.
$k := 0$.
repeat
  Choose a point $x^{(k+1)}$ in $\mathcal{P}_k$.
  Query the cutting-plane oracle at $x^{(k+1)}$.
  If the oracle determines that $x^{(k+1)} \in X$, quit.
  Else, update $\mathcal{P}_k$ by adding the new cutting-plane: $\mathcal{P}_{k+1} := \mathcal{P}_k \cap \{z \mid a_{k+1}^T z \leq b_{k+1}\}$.
  If $\mathcal{P}_{k+1} = \emptyset$, quit.
  $k := k + 1$.

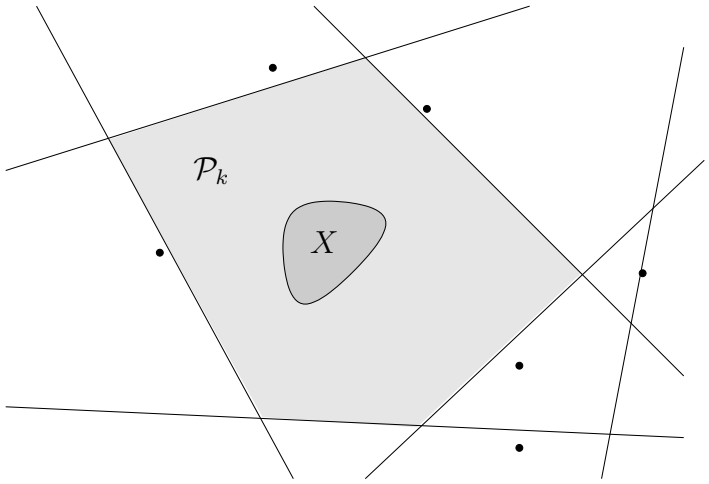

Figure 5: Points $x^{(1)}, \ldots, x^{(k)}$, shown as dots, and the associated cutting-planes, shown as lines. From these cutting-planes we conclude that the target set X (shown dark) lies inside the localization polyhedron $\mathcal{P}_k$, shown lightly shaded. We can limit our search for a point in X to $\mathcal{P}_k$.

Provided we choose $x^{(k+1)}$ in the interior of $\mathcal{P}_k$, this algorithm generates a strictly decreasing sequence of polyhedra, which contain X:

$$\mathcal{P}_0 \supseteq \cdots \supseteq P_k \supseteq X.$$

These inclusions are strict since each query point $x^{(j+1)}$ is in the interior of $\mathcal{P}_j$, but either outside, or on the boundary of, $\mathcal{P}_{(j+1)}$.

2.1.1 Measuring uncertainty and progress

The polyhedron $\mathcal{P}_k$ summarizes what we know, after k calls to the cutting-plane oracle, about the possible location of target points. The size of $\mathcal{P}_k$ gives a measure of our ignorance or uncertainty about target points: if $\mathcal{P}_k$ is small, we have localized the target points to within a small set; if $\mathcal{P}_k$ is large, we still have much uncertainty about where the target points might be.

There are several useful scalar measures of the size of the localization set $\mathcal{P}_k$. Perhaps the most obvious is its diameter, *i.e.*, the diameter d of the smallest ball that contains $\mathcal{P}_k$. If this ball is $\{\hat{x} + u \mid \|u\|_2 \leq d/2\}$, we can say that the target set has been localized to within a distance $d/2$ of the point $\hat{x}$. Using this measure, we can judge the progress in a given iteration by the reduction in the diameter of $\mathcal{P}_k$. (Since $\mathcal{P}_{k+1} \subseteq \mathcal{P}_k$, the diameter always decreases.)

Another useful scalar measure of the size of the localization set is its volume. Using this measure, we can judge the progress in iteration k by the fractional decrease in volume, *i.e.*,

$$\frac{\mathbf{vol}(\mathcal{P}_{k+1})}{\mathbf{vol}(\mathcal{P}_k)}.$$

This volume ratio is affine-invariant: if the problem (and choice of query point) is transformed by an affine change of coordinates, the volume ratio does not change, since if $T \in \mathbf{R}^{n\times n}$ is nonsingular,

$$\frac{\mathbf{vol}(T\mathcal{P}_{k+1})}{\mathbf{vol}(T\mathcal{P}_k)} = \frac{\mathbf{vol}(\mathcal{P}_{k+1})}{\mathbf{vol}(\mathcal{P}_k)}.$$

(The diameter ratio does not have this property.)

2.1.2 Convergence of cutting-plane methods

The convergence of a cutting-plane method can be judged in several ways. In one approach, we assume that the target set X has some minimum volume, or contains a ball of radius $r > 0$. We then show that the particular cutting-plane method produces a point in X within at most N iterations, where N depends on n, and possibly other problem parameters.

When a cutting-plane method is used to solve an optimization problem, we can judge convergence by the number of iterations required before we compute a point that is ϵ-suboptimal.

2.2 Choosing the query point

The cutting-plane algorithm described above is only conceptual, since the critical step, *i.e.*, how we choose the next query point $x^{(k+1)}$ inside the current localization polyhedron $\mathcal{P}_k$, is not fully specified. Roughly speaking, our goal is to choose query points that result in small localization polyhedra. We need to choose $x^{(k+1)}$ so that $\mathcal{P}_{k+1}$ is as small as possible, or equivalently, the new cut removes as much as possible from the current polyhedron $\mathcal{P}_k$. The reduction in size (say, volume) of $\mathcal{P}_{k+1}$ compared to $\mathcal{P}_k$ gives a measure of how informative the cutting-plane for $x^{(k+1)}$ is.

When we query the oracle at the point $x^{(k+1)}$, we do not know which cutting-plane will be returned; we only know that $x^{(k+1)}$ will be in the excluded halfspace. The informativeness of the cut, *i.e.*, how much smaller $\mathcal{P}_{k+1}$ is than $\mathcal{P}_k$, depends on the direction a_{k+1} of the cut, which we do not know before querying the oracle. This is illustrated in figure 6, which shows a localization polyhedron $\mathcal{P}_k$ and a query point $x^{(k+1)}$, and two cutting-planes that could be returned by the oracle. One of them gives a large reduction in the size of the localization polyhedron, but the other gives only a small reduction in size.

Since we want our algorithm to work well no matter which cutting-plane is returned by the oracle, we should choose $x^{(k+1)}$ so that, no matter which cutting-plane is returned by the oracle, we obtain a good reduction in the size of our localization polyhedron. This suggests that we should choose $x^{(k+1)}$ to be deep inside the polyhedron $\mathcal{P}_k$, *i.e.*, it should be some kind of center of $\mathcal{P}_k$. This is illustrated in figure 7, which shows the same localization polyhedron $\mathcal{P}_k$ as in figure 6 with a more central query point $x^{(k+1)}$. For this choice of query point, we cut away a good portion of $\mathcal{P}_k$ no matter which cutting-plane is returned by the oracle.

If we measure the informativeness of the kth cut using the volume reduction ratio $\mathbf{vol}(\mathcal{P}_{k+1})/\mathbf{vol}(\mathcal{P}_k)$, we seek a point $x^{(k+1)}$ such that, no matter what cutting-plane is returned by the oracle, we obtain a certain guaranteed volume reduction. For a cutting-plane with normal vector a, the least informative is the neutral one, since a deep cut with the

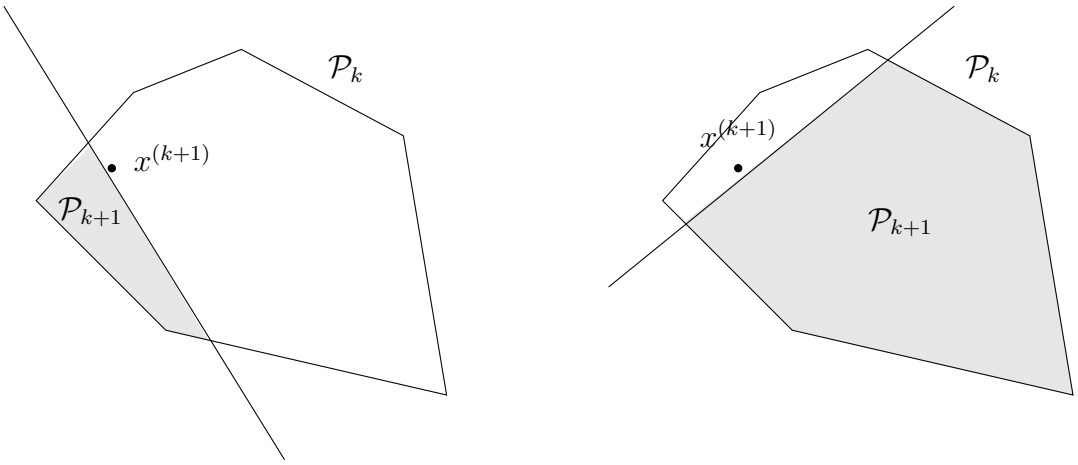

Figure 6: A localization polyhedron $\mathcal{P}_k$ and query point $x^{(k+1)}$, shown as a dot. Two possible scenarios are shown. *Left.* Here the cutting-plane returned by the oracle cuts a large amount from $\mathcal{P}_k$; the new polyhedron $\mathcal{P}_{k+1}$, shown shaded, is small. *Right.* Here the cutting-plane cuts only a very small part of $\mathcal{P}_k$; the new polyhedron $\mathcal{P}_{k+1}$, is not much smaller than $\mathcal{P}_k$.

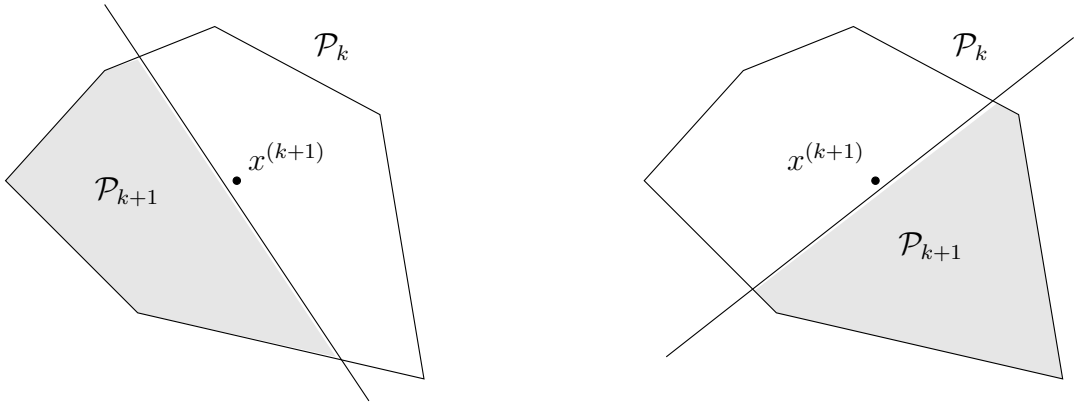

Figure 7: A localization polyhedron $\mathcal{P}_k$ and a more central query point x^{k+1} than in the example of figure 6. The same two scenarios, with different cutting-plane directions, are shown. In both cases we obtain a good reduction in the size of the localization polyhedron; even the worst possible cutting-plane at x would result in $\mathcal{P}_{k+1}$ substantially smaller than $\mathcal{P}_k$.

same normal vector leads to a smaller volume for $\mathcal{P}_{k+1}$. In a worst-case analysis, then, we can assume that the cuts are neutral, *i.e.*, have the form $a^T(z - x^{(k+1)}) \leq 0$. Let ρ denote the volume ratio, as a function of a,

$$\rho(a) = \frac{\mathbf{vol}(\mathcal{P}_k \cap \{z \mid a^T(z - x^{(k+1)}) \leq 0\})}{\mathbf{vol}(\mathcal{P}_k)}.$$

The function ρ is positively homogeneous, and satisfies

$$\rho(a) + \rho(-a) = 1.$$

To see this, note that a neutral cut with normal a divides $\mathcal{P}_k$ into two polyhedra,

$$\mathcal{P}_{k+1} = \mathcal{P}_k \cap \{z \mid a^T(z - x^{(k+1)}) \leq 0\}, \qquad \overline{\mathcal{P}}_{k+1} = \mathcal{P}_k \cap \{z \mid a^T(z - x^{(k+1)}) \geq 0\}.$$

The first is the new localization polyhedron, and the second is the polyhedron of points 'thrown out' by the cut. The sum of their volumes is the volume of $\mathcal{P}_k$. If the cut with normal $-a$ is considered, we get the same two polyhedra, with the new polyhedron and the polyhedron of 'thrown out' points switched.

From $\rho(a) + \rho(-a) = 1$, we see that the worst-case volume reduction satisfies

$$\sup_{a \neq 0} \rho(a) \geq 1/2.$$

This means that the worst-case volume reduction (over all possible cutting-planes that can be returned by the oracle) can never be better (smaller) than 1/2. The best possible (guaranteed) volume reduction we can have is 50% in each iteration.

2.3 Some specific cutting-plane methods

Several different choices for the query point have been proposed, which give different cutting-plane or localization algorithms. These include:

- The *center of gravity algorithm*, also called the *method of central sections*. The query point $x^{(k+1)}$ is chosen as the center of gravity of $\mathcal{P}_k$.
- *Maximum volume ellipsoid (MVE) cutting-plane method.* The query point $x^{(k+1)}$ is chosen as the center of the maximum volume ellipsoid contained in $\mathcal{P}_k$.
- *Analytic center cutting-plane method (ACCPM).* The query point $x^{(k+1)}$ is chosen as the analytic center of the inequalities defining $\mathcal{P}_k$.

Each of these methods has advantages and disadvantages, in terms of the computational effort required to determine the next query point, the theoretical complexity of the method, and the practical performance of the method. We will describe the center of gravity algorithm and the MVE method briefly, later in this section. The analytic center cutting-plane method, which combines good practical performance with reasonable simplicity, will be described in more detail in §3.

2.3.1 Bisection method on R

We first describe a very important cutting-plane method: the *bisection method*. We consider the special case $n = 1$, *i.e.*, a one-dimensional search problem. We will describe the traditional setting in which the target set X is the singleton $\{x^*\}$, and the cutting-plane oracle always returns a neutral cut. The cutting-plane oracle, when queried with $x \in \mathbf{R}$, tells us either that $x^* \leq x$ or that $x^* \geq x$. In other words, the oracle tells us whether the point x^* we seek is to the left or right of the current point x.

The localization polyhedron $\mathcal{P}_k$ is an interval, which we denote $[l_k, u_k]$. In this case, there is an obvious choice for the next query point: we take $x^{(k+1)} = (l_k + u_k)/2$, the midpoint of the interval. The bisection algorithm is:

Bisection algorithm for one-dimensional search.

given an initial interval $[l, u]$ known to contain x^*; a required tolerance $r > 0$
repeat
  $x := (l + u)/2$.
  Query the oracle at x.
  If the oracle determines that $x^* \leq x$, $u := x$.
  If the oracle determines that $x^* \geq x$, $l := x$.
until $u - l \leq 2r$

In each iteration the localization interval is replaced by either its left or right half, *i.e.*, it is *bisected*. The volume reduction factor is the best it can be: it is always exactly 1/2. Let $2R = u_0 - l_0$ be the length of the initial interval (*i.e.*, $2R$ gives it diameter). The length of the localization interval after k iterations is then $2^{-k}2R$, so the bisection algorithm terminates after exactly

$$k = \lceil \log_2(R/r) \rceil \tag{4}$$

iterations. Since x^* is contained in the final interval, we are guaranteed that its midpoint (which would be the next iterate) is no more than a distance r from x^*. We can interpret R/r as the ratio of the initial to final uncertainty. The equation (4) shows that the bisection method requires exactly one iteration per bit of reduction in uncertainty.

It is straightforward to modify the bisection algorithm to handle the possibility of deep cuts, and to check whether the updated interval is empty (which implies that $X = \emptyset$). In this case, the number $\lceil \log_2(R/r) \rceil$ is an upper bound on the number of iterations required.

The bisection method can be used as a simple method for minimizing a differentiable convex function on $\mathbf{R}$, *i.e.*, carrying out a line search. The cutting-plane oracle only needs to determine the sign of $f'(x)$, which determines whether the minimizing set is to the left (if $f'(x) \geq 0$) or right (if $f'(x) \leq 0$) of the point x.

2.3.2 Center of gravity method

The center of gravity method, or CG algorithm, was one of the first localization methods proposed. In this method we take the query point to be $x^{(k+1)} = \mathbf{cg}(\mathcal{P}_k)$, where the center

of gravity of a set $C \subseteq \mathbf{R}^n$ is defined as

$$\mathbf{cg}(C) = \frac{\int_C z \, dz}{\int_C \, dz},$$

assuming C is bounded and has nonempty interior.

The center of gravity turns out to be a very good point in terms of the worst-case volume reduction factor: we always have

$$\frac{\mathbf{vol}(\mathcal{P}_{k+1})}{\mathbf{vol}(\mathcal{P}_k)} \leq 1 - 1/e \approx 0.63.$$

In other words, the volume of the localization polyhedron is reduced by at least 37% at each step. Note that this guaranteed volume reduction is completely independent of all problem parameters, including the dimension n.

This guarantee comes from the following result: suppose $C \subseteq \mathbf{R}^n$ is convex, bounded, and has nonempty interior. Then for any nonzero $a \in \mathbf{R}^n$, we have

$$\mathbf{vol}\left(C \cap \{z \mid a^T(z - \mathbf{cg}(C)) \leq 0\}\right) \leq (1 - 1/e)\,\mathbf{vol}(C).$$

In other words, a plane passing though the center of gravity of a convex set divides its volume almost equally: the volume division inequity can be at most in the ratio $(1 - 1/e) : 1/e$, *i.e.*, about $1.72 : 1$. (See exercise **??**.)

In the CG algorithm we have

$$\mathbf{vol}(\mathcal{P}_k) \leq (1 - 1/e)^k \,\mathbf{vol}(\mathcal{P}_0) \approx 0.63^k \,\mathbf{vol}(\mathcal{P}_0).$$

Now suppose the initial polyhedron lies inside a Euclidean ball of radius R (*i.e.*, it has diameter $\leq 2R$), and the target set contains a Euclidean ball of radius r. Then we have

$$\mathbf{vol}(\mathcal{P}_0) \leq \alpha_n R^n,$$

where α_n is the volume of the unit Euclidean ball in $\mathbf{R}^n$. Since $X \subseteq \mathcal{P}_k$ for each k (assuming the algorithm has not yet terminated) we have

$$\alpha_n r^n \leq \mathbf{vol}(\mathcal{P}_k).$$

Putting these together we see that

$$\alpha_n r^n \leq (1 - 1/e)^k \alpha_n R^n,$$

so

$$k \leq \frac{n \log(R/r)}{-\log(1 - 1/e)} \approx 2.18 n \log(R/r).$$

We can express this using base-2 logarithms as

$$k \leq \frac{n (\log 2) \log_2(R/r)}{-\log(1 - 1/e)} \approx 1.51 n \log_2(R/r),$$

in order to compare this complexity estimate with the similar one for the bisection algorithm (4). We conclude that the CG algorithm requires at most $1.51n$ iterations per bit of uncertainty reduction. (Of course, the CG algorithm reduces to the bisection method when $n = 1$.)

Finally, we come to a very basic disadvantage of the CG algorithm: it is *extremely difficult* to compute the center of gravity of a polyhedron in $\mathbf{R}^n$, described by a set of linear inequalities. (It is possible to efficiently compute the center of gravity in very low dimensions, *e.g.*, $n = 2$ or $n = 3$, by triangulation.) This means that the CG algorithm, although interesting, is not a practical cutting-plane method. Variants of the CG algorithm have been developed, in which an approximate center of gravity, which can be efficiently computed, is used in place of the center of gravity, but they are quite complicated.

2.3.3 MVE cutting-plane method

In the maximum volume inscribed ellipsoid method, we take the next iterate to be the center of the maximum volume ellipsoid that lies in $\mathcal{P}_k$, which can be computed by solving a convex optimization problem. Since the maximum volume inscribed ellipsoid is affinely invariant, so is the resulting MVE cutting-plane method.

Recall that the maximum volume ellipsoid inscribed in $\mathcal{P}_k$, when scaled about its center $x^{(k+1)}$ by a factor n, is guaranteed to cover $\mathcal{P}_k$. This gives a simple ellipsoidal uncertainty bound.

2.4 Extensions

In this section we describe several extensions and variations on cutting-plane methods.

2.4.1 Multiple cuts

One simple extension is to allow the oracle to return a set of linear inequalities for each query, instead of just one. When queried at $x^{(k)}$, the oracle returns a set of linear inequalities which are satisfied by every $z \in X$, and which (together) separate $x^{(k)}$ and X. Thus, the oracle can return $A_k \in \mathbf{R}^{p_k \times n}$ and $b_k \in \mathbf{R}^{p_k}$, where $A_k z \preceq b_k$ holds for every $z \in X$, and $A_k x^{(k)} \not\preceq b_k$. This means that at least one of the p_k linear inequalities must be a valid cutting-plane by itself. The inequalities for which are not valid cuts by themselves are called *shallow cuts*.

It is straightforward to accommodate multiple cutting-planes in a cutting-plane method: at each iteration, we simply append the entire set of new inequalities returned by the oracle to our collection of valid linear inequalities for X. To give a simple example showing how multiple cuts can be obtained, consider the convex feasibility problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m. \end{array}$$

In §1.2 we showed how to construct a cutting-plane at x using any (one) violated constraint. We can obtain a set of multiple cuts at x by using information from *any* set inequalities, provided at least one is violated. From the basic inequality

$$f_j(z) \geq f_j(x) + \nabla f_j(x)^T (z - x),$$

we find that every $z \in X$ satisfies

$$f_j(x) + \nabla f_j(x)^T(z - x) \leq 0.$$

If x violates the jth inequality, this is a deep cut, since x does not satisfy it. If x satisfies the jth inequality, this is a shallow cut, and can be used in a group of multiple cuts, as long as one neutral or deep cut is present. Common choices for the set of inequalities to use to form cuts are

- the most violated inequality, *i.e.*, $\operatorname{argmax}_j f_j(x)$,
- any violated inequality (*e.g.*, the first constraint found to be violated),
- all violated inequalities,
- all inequalities.

2.4.2 Dropping or pruning constraints

The computation required to compute the new query point $x^{(k+1)}$ grows with the number of linear inequalities that describe $\mathcal{P}_k$. This number, in turn, increases by one at each iteration (for a single cut) or more (for multiple cuts). For this reason most practical cutting-plane implementations include a mechanism for dropping or pruning the set of linear inequalities as the algorithm progresses. In the conservative approach, constraints are dropped only when they are known to be redundant. In this case dropping constraints does not change $\mathcal{P}_k$, and the convergence analysis for the cutting-plane algorithm without pruning still holds. The progress, judged by volume reduction, is unchanged when we drop constraints that are redundant.

To check if a linear inequality $a_i^T z \leq b_i$ is redundant, *i.e.*, implied by the linear inequalities $a_j^T z \leq b_j$, $j = 1, \ldots, m$, we can solve the linear program

$$\begin{array}{ll} \text{maximize} & a_i^T z \\ \text{subject to} & a_j^T z \leq b_j, \quad j = 1, \ldots, m, \ j \neq i. \end{array}$$

The linear inequality is redundant if and only if the optimal value is larger than or equal to b_i. Solving a linear program to check redundancy of each inequality is usually too costly, and therefore not done.

In some cases there are other methods that can identify (some) redundant constraints, with far less computational effort. Suppose, for example, the ellipsoid

$$\mathcal{E} = \{Fu + g \mid \|u\|_2 \leq 1\} \tag{5}$$

is known to cover the current localization polyhedron $\mathcal{P}$. (In the MVE cutting-plane method, such as ellipsoid can be obtained by expanding the maximum volume ellipsoid inside $\mathcal{P}_k$ by a factor of n about its center.) If the maximum value of $a_i^T z$ over $\mathcal{E}$ is smaller than or equal to b_i, *i.e.*,

$$a_i^T g + \|F^T a_i\|_2 \leq b_i,$$

then the constraint $a_i^T z \leq b_i$ is redundant.

In other approaches, constraints are dropped even when they are not redundant, or at least not known to be redundant. In this case the pruning can actually increase the size of the localization polyhedron. Heuristics are used to rank the linear inequalities in relevance, and the least relevant ones are dropped first. One method for ranking relevance is based on a covering ellipsoid (5). For each linear inequality we form the fraction

$$\frac{a_i^T g - b_i}{\|F^T a_i\|_2},$$

and then sort the inequalities by these factors, with the lowest numbers corresponding to the most relevant, and the largest numbers corresponding to the least relevant. Note that any inequality for which the fraction exceeds one is in fact redundant.

One common strategy is to keep a fixed number N of linear inequalities, by dropping as many constraints as needed, in each iteration, to keep the total number fixed at N. The number of inequalities kept (*i.e.*, N) is typically between $3n$ and $5n$. Dropping non-redundant constraints complicates the convergence analysis of a cutting-plane algorithm, sometimes considerably, so we will not consider pruning in our analysis. In practice, pruning often improves the performance considerably.

2.4.3 Suboptimality bounds and stopping criteria for cutting-plane methods

A cutting-plane method can be used to solve a convex optimization problem with variable $x \in \mathbf{R}^n$, by taking X to be the set of optimal points, and using the methods described in §1.2 to generate cutting-planes. At each step we have a localization polyhedron $\mathcal{P}_k$ known to contain the optimal points, and from this we can find a bound on the suboptimality. Since f_0 is convex we have

$$f_0(z) \geq f_0(x^{(k+1)}) + \nabla f_0(x^{(k+1)})^T (z - x^{(k+1)})$$

for all z, and so

$$\begin{aligned} p^\star &= \inf\{f_0(z) \mid z \in C\} \\ &\geq \inf\{f_0(x^{(k+1)}) + \nabla f_0(x^{(k+1)})^T (z - x^{(k+1)}) \mid z \in \mathcal{P}_k\}, \end{aligned}$$

where $p^\star$ is the optimal value of the optimization problem, and C is its constraint set. Thus, we can obtain a lower bound on $p^\star$, at each step, by solving a linear program. This is usually not done, however, because of the computational cost. If a covering ellipsoid $\mathcal{E}_k \supseteq \mathcal{P}_k$ is available, we can cheaply compute a bound on $p^\star$:

$$p^\star \geq \inf\{f_0(x^{(k+1)}) + \nabla f_0(x^{(k+1)})^T (z - x^{(k+1)}) \mid z \in \mathcal{E}_k\}.$$

A simple stopping criterion based on these lower bounds is

$$x^{(k)} \text{ feasible and } f_0(x^{(k)}) - l_k \leq \epsilon,$$

where l_k is the lower bound computed at iteration k. This criterion guarantees that $x^{(k)}$ is no more than ϵ-suboptimal. A more sophisticated variation is to keep track of the best feasible point found so far (*i.e.*, the one with the smallest objective value) and also the best lower bound on $p^\star$ found so far. We then terminate when the difference between the objective value of the best feasible point, and the best lower bound, are within a tolerance ϵ.

2.5 Epigraph cutting-plane method

For a convex optimization problem, it is usually better to apply a cutting-plane method to the *epigraph form* of the problem, rather than directly to the problem.

We start with the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{6}$$

where $f_0, \ldots, f_m$ are convex and differentiable. In the approach outlined above, we take the variable to be x, and the target set X to be the set of optimal points. Cutting-planes are found using the methods described in §1.2.

Suppose instead we form the equivalent epigraph form problem

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & f_0(x) \leq t \\ & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{7}$$

with variables $x \in \mathbf{R}^n$ and $t \in \mathbf{R}$. We take the target set to be the set of optimal points for the epigraph problem (7), *i.e.*,

$$X = \{(x, f_0(x)) \mid x \text{ optimal for (7)}\}.$$

Let us show how to find a cutting-plane, in $\mathbf{R}^{n+1}$, for this version of the problem, at the query point (x, t). First suppose x is not feasible for the original problem, *e.g.*, the jth constraint is violated. Every feasible point satisfies

$$0 \geq f_j(z) \geq f_j(x) + \nabla f_j(x)^T (z - x),$$

so we can use the cut

$$f_j(x) + \nabla f_j(x)^T (z - x) \leq 0$$

(which doesn't involve the second variable). Now suppose that the query point x is feasible and $\nabla f_0(x) \neq 0$ (if $\nabla f_0(x) = 0$, then x is optimal). For any $(z, s) \in \mathbf{R}^{n+1}$ feasible for the problem (6), we have

$$s \geq f_0(z) \geq f_0(x) + \nabla f_0(x)^T (z - x)$$

Since x is feasible, $f_0(x) \geq p^\star$, which is the optimal value of the second variable. Thus, we can construct *two* cutting-planes in (z, s):

$$f_0(x) + \nabla f_0(x)^T (z - x) \leq s, \qquad s \leq f_0(x_0).$$

3 Analytic center cutting-plane method

In this section we describe in more detail the analytic center cutting-plane method (ACCPM).

Analytic center cutting-plane method (ACCPM)

given an initial polyhedron $\mathcal{P}_0 = \{z \mid A_0 z \preceq b_0\}$ known to contain X.
$k := 0$.
repeat
  Compute $x^{(k+1)}$, the analytic center of the inequalities $A_k x \preceq b_k$.
  Query the cutting-plane oracle at $x^{(k+1)}$.
  If the oracle determines that $x^{(k+1)} \in X$, quit.
  Else, update $\mathcal{P}_k$ by adding the new cutting-plane $a^T z \leq b$:
    $A_{k+1} = \left[\begin{array}{c} A_k \\ a^T \end{array}\right], \quad b_{k+1} = \left[\begin{array}{c} b_k \\ b \end{array}\right].$
  If $\mathcal{P}_{k+1} = \{z \mid A_{k+1} z \preceq b_{k+1}\} = \emptyset$, quit.
  $k := k + 1$.

3.1 Updating the analytic center

In ACCPM, each iteration requires computing the analytic center of a set of linear inequalities (and, possibly, determining whether the set of linear inequalities is feasible). In this section we describe several methods that can be used to do this.

To find the analytic center, we must solve the problem

$$\text{minimize} \quad -\sum_{i=1}^m \log(b_i - a_i^T x) \tag{8}$$

This is an unconstrained problem, but the domain of the objective function is the open polyhedron

$$\{x \mid a_i^T x < b_i,\ i = 1, \ldots, m\},$$

i.e., the interior of the polyhedron. We can use Newton's method to solve the problem (8) provided we can find a starting point $\hat{x}$ in the domain, *i.e.*, one that satisfies the strict inequalities $a_i^T \hat{x} < b_i$, $i = 1, \ldots, m$. Unfortunately, such a point is not obvious. The previous iterate x_{prev} in ACCPM is not a candidate, since by definition it has been cut from the previous polyhedron; it *must* violate at least one of the strict inequalities added in the previous iteration.

One simple approach is to use a phase I optimization method to find a point $\hat{x}$ that satisfies the strict inequalities (or determine that the inequalities are infeasible). We form the problem

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & a_i^T x - b_i \leq t, \quad i = 1, \ldots, m, \end{array}$$

with variables $x \in \mathbf{R}^n$ and $t \in \mathbf{R}$, and solve it using SUMT. For this problem we can use the previous iterate x_{prev} as the initial value for x, and choose any t that satisfies the inequalities, *e.g.*, $t = \max_i(a_i^T x_{\text{prev}} - b_i) + 1$. We can terminate when $t < 0$, which gives a strictly feasible starting point for (8), or when the dual objective is greater than or equal to zero. If the dual objective is greater than zero, we have a proof that the (nonstrict) inequalities are infeasible; if the dual variable is equal to zero, we have a proof that the strict inequalities are infeasible.

This simple approach has the advantage that it works for multiple cuts, and can be used to compute the analytic center of the initial polyhedron when it is more complex than a box (for which the analytic center is obvious).

In the next two sections we describe other methods that can be used to solve the problem (8), without carrying out the phase I process.

3.1.1 An initial point for neutral cut

When we add only one cut, and the cut is neutral, we can analytically find a point $\hat{x}$ that satisfies the strict inequalities, and therefore can be used as the starting point for the Newton process. Let $x_{\rm prev}$ denote the analytic center of the inequalities

$$a_i^T x \leq b_i, \quad i = 1, \ldots, m,$$

and suppose the new cut is

$$a_{m+1}^T (x - x_{\rm prev}) \leq 0.$$

We can represent the new cut in this form since it is neutral; the new cutting-plane passes through the point $x_{\rm prev}$. The problem is to find a point $\hat{x}$ that satisfies the $m+1$ strict inequalities

$$a_i^T x < b_i, \quad i = 1, \ldots, m, \qquad a_{m+1}^T x < b_{m+1} = a_{m+1}^T x_{\rm prev}. \tag{9}$$

The previous analytic center $x_{\rm prev}$ satisfies the first m strict inequalities, but does not satisfy the last one. In fact, for any $\delta \in \mathbf{R}^n$ that is small enough and satisfies $\delta^T a_{m+1} < 0$, the point $x + \delta$ satisfies the strict inequalities, and can be used as the starting point for the Newton process.

Using the inner ellipsoid determined by the Hessian of the barrier function at $x_{\rm prev}$, we can give an explicit step δ that will always work. Let H denote the Hessian of the barrier function at $x_{\rm prev}$,

$$H = \sum_{i=1}^m (b_i - a_i^T x_{\rm prev})^{-2} a_i a_i^T.$$

Recall from §**??** that the open ellipsoid

$$\mathcal{E}_{\rm in} = \{x_{\rm prev} + u \mid u^T H u < 1\}$$

is guaranteed to be inside the polyhedron, *i.e.*, satisfy the strict inequalities $a_i^T z < b_i$, $i = 1, \ldots, m$. A natural choice for a step is then

$$\delta = \mathrm{argmin}_{z \in \mathcal{E}_{\rangle\backslash}} a_{m+1}^T z$$

which is given by

$$\delta = \frac{-H^{-1} a_{m+1}}{\sqrt{a_{m+1}^T H^{-1} a_{m+1}}}.$$

In other words, we can simply take

$$\hat{x} = x_{\rm prev} - \frac{H^{-1} a_{m+1}}{\sqrt{a_{m+1}^T H^{-1} a_{m+1}}},$$

which is a point guaranteed to satisfy the strict inequalities (9), and start the Newton process for minimizing the new log barrier function from there.

3.2 Convergence proof

In this section we analyze the convergence of ACCPM. We take the initial polyhedron as the unit box, centered at the origin, with unit length sides, *i.e.*, the initial set of linear inequalities is

$$-(1/2)\mathbf{1} \preceq z \preceq (1/2)\mathbf{1},$$

so the first analytic center is $x^{(1)} = 0$. We assume the target set X contains a ball with radius $r < 1/2$, and show that the number of iterations is no more than a constant times n^2/r^2.

Assuming the algorithm has not terminated, the set of inequalities after k iterations is

$$-(1/2)\mathbf{1} \preceq z \preceq (1/2)\mathbf{1}, \qquad a_i^T z \leq b_i, \quad i = 1, \ldots, k. \tag{10}$$

We assume the cuts are neutral, so $b_i = a_i^T x^{(i)}$ for $i = 1, \ldots, k$. Without loss of generality we normalize the vectors a_i so that $\|a_i\|_2 = 1$. We will let $\phi_k : \mathbf{R}^n \to \mathbf{R}$ be the logarithmic barrier function associated with the inequalities (10),

$$\phi_k(z) = -\sum_{i=1}^{n} \log(1/2 + z_i) - \sum_{i=1}^{n} \log(1/2 - z_i) - \sum_{i=1}^{k} \log(b_i - a_i^T x).$$

The iterate $x^{(k+1)}$ is the minimizer of this logarithmic barrier function.

Since the algorithm has not terminated, the polyhedron $\mathcal{P}_k$ defined by (10) still contains the target set X, and hence also a ball with radius r and (unknown) center x_c. We have $(-1/2 + r\mathbf{1} \preceq x_c \preceq (1/2 + r)\mathbf{1}$, and the slacks of the inequalities $a_i^T z \leq b_i$ evaluated at x_c also exceed r:

$$b_i - \sup_{\|v\|_2 \leq 1} a_i^T (x_c + rv) = b_i - a_i^T x_c - r\|a_i\|_2 = b_i - a_i^T x_c - r \geq 0.$$

Therefore $\phi_k(x_c) \leq -(2n + k) \log r$ and, since $x^{(k)}$ is the minimizer of ϕ_k,

$$\phi_k(x^{(k)}) = \inf_z \phi_k(z) \leq \phi_k(x_c) \leq (2n + k) \log(1/r).$$

We can also derive a lower bound on $\phi_k(x^{(k)})$ by noting that the functions ϕ_j are self-concordant for $j = 1, \ldots, k$. Using the inequality (**??**), we have

$$\phi_j(x) \geq \phi_j(x^{(j)}) + \sqrt{(x - x^{(j)})^T H_j (x - x^{(j)})} - \log(1 + \sqrt{(x - x^{(j)})^T H_j (x - x^{(j)})})$$

for all $x \in \mathbf{dom}\, \phi_j$, where H_j is the Hessian of ϕ_j at $x^{(j)}$. If we apply this inequality to ϕ_{k-1} we obtain

$$\begin{aligned}
\phi_k(x^{(k)}) &= \inf_x \phi_k(x) \\
&= \inf_x \left(\phi_{k-1}(x) - \log(-a_k^T (x - x^{(k-1)})) \right) \\
&\geq \inf_v \left(\phi_{k-1}(x^{(k-1)}) + \sqrt{v^T H_{k-1} v} - \log(1 + \sqrt{v^T H_{k-1} v}) - \log(-a_k^T v) \right).
\end{aligned}$$

By setting the gradient of the righthand side equal to zero, we find that it is minimized at

$$\hat{v} = -\frac{1+\sqrt{5}}{2\sqrt{a_k^T H_{k-1}^{-1} a_k}} H_{k-1}^{-1} a_k,$$

which yields

$$\begin{aligned}
\phi_k(x^{(k)}) &\geq \phi_{k-1}(x^{(k-1)}) + \sqrt{\hat{v}^T H_{k-1} \hat{v}} - \log(1 + \sqrt{\hat{v}^T H_{k-1} \hat{v}}) - \log(-a_k^T \hat{v}) \\
&= \phi_{(k-1)}(x^{k-1}) + 0.1744 - \frac{1}{2}\log(a_k^T H_{k-1}^{-1} a_k) \\
&= 0.1744k - \frac{1}{2}\sum_{i=1}^{k} \log(a_i^T H_{i-1}^{-1} a_i) + 2n \log 2 \\
&\geq 0.1744k - \frac{k}{2}\log\left(\frac{1}{k}\sum_{i=1}^{k} a_i^T H_{i-1}^{-1} a_i\right) + 2n \log 2 \\
&\geq -\frac{k}{2}\log\left(\frac{1}{k}\sum_{i=1}^{k} a_i^T H_{i-1}^{-1} a_i\right) + 2n \log 2 \qquad (11)
\end{aligned}$$

because $\phi_0(x^{(0)}) = 2n \log 2$. We can further bound the second term on the righthand side by noting that

$$H_i = 4\,\mathbf{diag}(\mathbf{1} + 2x^{(i)})^{-2} + 4\,\mathbf{diag}(\mathbf{1} - 2x^{(i)})^{-2} + \sum_{j=1}^{i} \frac{1}{(b_j - a_j^T x^{(i)})^2} a_j a_j^T \succeq I + \frac{1}{n}\sum_{j=1}^{i} a_j a_j^T$$

because $-(1/2)\mathbf{1} \prec x^{(i)} \prec (1/2)\mathbf{1}$ and

$$b_i - a_i^T x^{(k)} = a_i^T (x^{(i-1)} - x^{(k)}) \leq \|a_i\|_2 \|x^{(i-1)} - x^{(k)}\|_2 \leq \sqrt{n}.$$

Define $B_0 = I$ and $B_i = I + (1/n)\sum_{j=1}^{i} a_j a_j^T$ for $i \geq 1$. Then

$$\begin{aligned}
n \log(1 + k/n^2) = n \log(\mathbf{Tr}\, B_k/n) &\geq \log\det B_k \\
&= \log\det B_{k-1} + \log(1 + \frac{1}{n} a_k^T B_{k-1}^{-1} a_k) \\
&\geq \log\det B_{k-1} + \frac{1}{2n} a_k^T B_{k-1}^{-1} a_k \\
&\geq \frac{1}{2n}\sum_{i=1}^{k} a_i^T B_{i-1}^{-1} a_i.
\end{aligned}$$

(The second inequality follows from the fact that $a_k^T B_{k-1}^{-1} a_k \leq 1$, and $\log(1+x) \geq (\log 2)x \geq x/2$ for $0 \leq x \leq 1$.) Therefore

$$\sum_{i=1}^{k} a_i^T H_{i-1}^{-1} a_i \leq \sum_{i=1}^{k} a_i^T B_{i-1}^{-1} a_i \leq 2n^2 \log(1 + \frac{k}{n^2}),$$

and we can simplify (11) as

$$\begin{aligned}\phi_k(x^{(k)}) &\geq -\frac{k}{2}\log\left(2\frac{\log(1+k/n^2)}{k/n^2}\right) - 2n\log 2 \\ &= -k\log\sqrt{2} + k\log\left(\frac{k/n^2}{\log(1+k/n^2)}\right) - 2n\log 2. \end{aligned} \tag{12}$$

Combining this lower bound with the upper bound (12) we find

$$-k\log\sqrt{2} + k\log\left(\frac{k/n^2}{\log(1+k/n^2)}\right) \leq k\log(1/r) + 2n\log(2/r). \tag{13}$$

From this it is clear that the algorithm terminates after a finite k: since the ratio $(k/n^2)/\log(1+k/n^2)$ goes to infinity, the left hand side grows faster than linearly as k increases.

We can derive an explicit bound on k as follows. Let $\alpha(r)$ be the solution of the nonlinear equation

$$\alpha/\log(1+\alpha) = 2\sqrt{2}/r^2.$$

Suppose $k > \max\{2n, n^2\alpha(r)\}$. Then we have a contradiction in (13):

$$k\log(2/r^2) \leq -k\log\sqrt{2} + k\log\left(\frac{k/n^2}{\log(1+k/n^2)}\right) \leq k\log(1/r) + 2n\log(2/r),$$

i.e., $k\log(2/r) \leq 2n\log(2/r)$. We conclude that

$$\max\{2n, n^2\alpha(r)\}$$

is an upper bound on the number of iterations. Note that it grows as n^2/r^2.

4 Subgradients

4.1 Motivation and definition

Localization and cutting-plane methods rely only on our ability to compute a cutting-plane for the problem at any point. In §1.2 we showed how to compute cutting-planes using the basic inequality

$$f(z) \geq f(x) + \nabla f(x)^T(z-x)$$

that holds for a convex differentiable function f. It turns out that an inequality like this holds for a convex function f even when it is not differentiable. This observation, and the associated calculus, will allow us to use cutting-plane methods to solve problems for which the objective and constraint functions are convex, but not differentiable.

We say a vector $g \in \mathbf{R}^n$ is a *subgradient* of $f : \mathbf{R}^n \to \mathbf{R}$ at $x \in \mathbf{dom}\, f$ if for all $z \in \mathbf{dom}\, f$,

$$f(z) \geq f(x) + g^T(z-x). \tag{14}$$

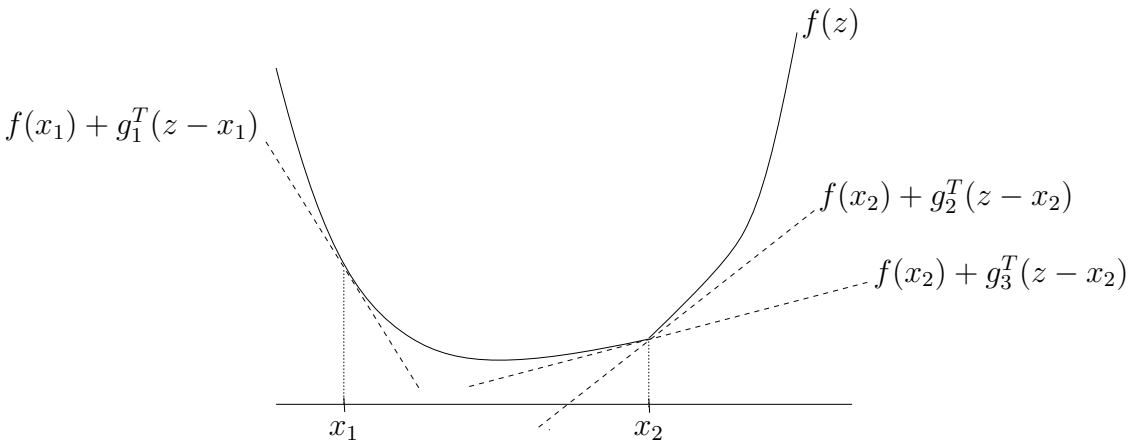

Figure 8: At x_1, the convex function f is differentiable, and g_1 (which is the derivative of f at x_1) is the unique subgradient at x_1. At the point x_2, f is not differentiable. At this point, f has many subgradients: two subgradients, g_2 and g_3, are shown.

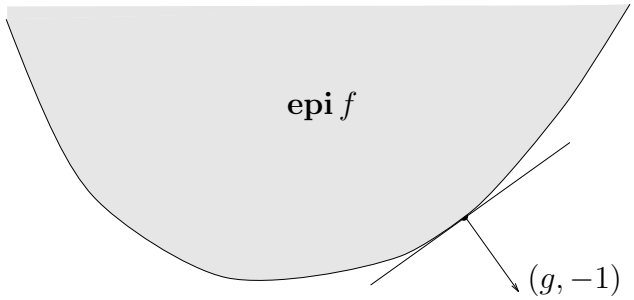

Figure 9: A vector $g \in \mathbf{R}^n$ is a subgradient of f at x if and only if $(g, -1)$ defines a supporting hyperplane to **epi** f at $(x, f(x))$.

If f is convex and differentiable, then its gradient at x is a subgradient. But a subgradient can exist even when f is not differentiable at x, as illustrated in figure 8. The same example shows that there can be more than one subgradient of a function f at a point x.

There are several ways to interpret a subgradient. A vector g is a subgradient of f at x if the affine function (of z) $f(x) + g^T(z - x)$ is a global underestimator of f. Geometrically, g is a subgradient of f at x if $(g, -1)$ supports **epi** f at $(x, f(x))$, as illustrated in figure 9.

A function f is called *subdifferentiable* at x if there exists at least one subgradient at x. The set of subgradients of f at the point x is called the *subdifferential* of f at x, and is denoted $\partial f(x)$. A function f is called subdifferentiable if it is subdifferentiable at all $x \in$ **dom** f.

Example. *Absolute value.* Consider $f(z) = |z|$. For $x < 0$ the subgradient is unique: $\partial f(x) = \{-1\}$. Similarly, for $x > 0$ we have $\partial f(x) = \{1\}$. At $x = 0$ the subdifferential is defined by the inequality $|z| \geq gz$ for all z, which is satisfied if and only if $g \in [-1, 1]$. Therefore we have $\partial f(0) = [-1, 1]$. This is illustrated in figure 10.

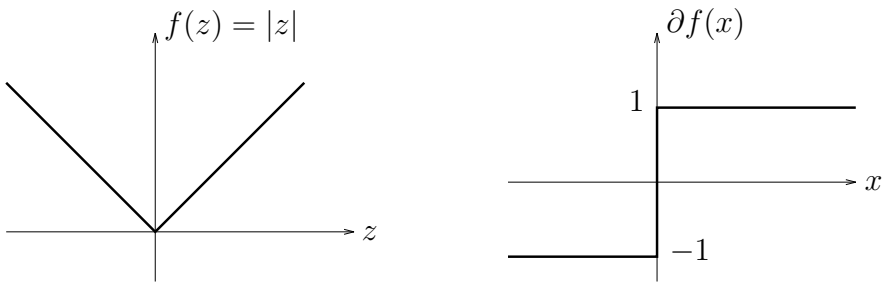

Figure 10: The absolute value function (left), and its subdifferential $\partial f(x)$ as a function of x (right).

4.2 Basic properties

The subdifferential $\partial f(x)$ is always a closed convex set, even if f is not convex. This follows from the fact that it is the intersection of an infinite set of halfspaces:

$$\partial f(x) = \bigcap_{z \in \mathbf{dom}\, f} \{g \mid f(z) \geq f(x) + g^T(z - x)\}.$$

4.2.1 Existence of subgradients

If f is convex and $x \in \mathbf{int}\,\mathbf{dom}\, f$, then $\partial f(x)$ is nonempty and bounded. To establish that $\partial f(x) \neq \emptyset$, we apply the supporting hyperplane theorem to the convex set $\mathbf{epi}\, f$ at the boundary point $(x, f(x))$, to conclude the existence of $a \in \mathbf{R}^n$ and $b \in \mathbf{R}$, not both zero, such that

$$\left[\begin{array}{c} a \\ b \end{array}\right]^T \left(\left[\begin{array}{c} z \\ t \end{array}\right] - \left[\begin{array}{c} x \\ f(x) \end{array}\right] \right) = a^T(z - x) + b(t - f(x)) \leq 0$$

for all $(z, t) \in \mathbf{epi}\, f$. This implies $b \leq 0$, and that

$$a^T(z - x) + b(f(z) - f(x)) \leq 0$$

for all z. If $b \neq 0$, we can divide by b to obtain

$$f(z) \geq f(x) - (a/b)^T(z - x),$$

which shows that $-a/b \in \partial f(x)$. Now we show that $b \neq 0$, *i.e.*, that the supporting hyperplane cannot be vertical. If $b = 0$ we conclude that $a^T(z - x) \leq 0$ for all $z \in \mathbf{dom}\, f$. This is impossible since $x \in \mathbf{int}\,\mathbf{dom}\, f$.

This discussion shows that a convex function has a subgradient at x if there is at least one nonvertical supporting hyperplane to $\mathbf{epi}\, f$ at $(x, f(x))$. This is the case, for example, if f is continuous. There are pathological convex functions which do not have subgradients at some points (see exercise 1), but we will assume in the sequel that all convex functions are subdifferentiable (at every point in $\mathbf{dom}\, f$).

4.2.2 Subgradients of differentiable functions

If f is convex and differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$, *i.e.*, its gradient is its only subgradient. Conversely, if f is convex and $\partial f(x) = \{g\}$, then f is differentiable at x and $g = \nabla f(x)$.

4.2.3 The minimum of a nondifferentiable function

A point $x^\star$ is a minimizer of a convex function f if and only if f is subdifferentiable at $x^\star$ and

$$0 \in \partial f(x^\star),$$

i.e., $g = 0$ is a subgradient of f at $x^\star$. This follows directly from the fact that $f(x) \geq f(x^\star)$ for all $x \in \mathbf{dom}\, f$.

This condition $0 \in \partial f(x^\star)$ reduces to $\nabla f(x^\star) = 0$ if f is differentiable at $x^\star$.

4.3 Calculus of subgradients

In this section we describe rules for constructing subgradients of convex functions. We will distinguish two levels of detail. In the 'weak' calculus of subgradients the goal is to produce one subgradient, even if more subgradients exist. This is sufficient in practice, since cutting-plane methods require only *a* subgradient at any point.

A second and much more difficult task is to describe the complete set of subgradients $\partial f(x)$ as a function of x. We will call this the 'strong' calculus of subgradients. It is useful in theoretical investigations, for example, when describing the precise optimality conditions.

Nonnegative scaling

For $\alpha \geq 0$, $\partial(\alpha f)(x) = \alpha \partial f(x)$.

Sum and integral

Suppose $f = f_1 + \cdots + f_m$, where $f_1, \ldots, f_m$ are convex functions. Then we have

$$\partial f(x) = \partial f_1(x) + \cdots + \partial f_m(x).$$

This property extends to infinite sums, integrals, and expectations (provided they exist).

Affine transformations of domain

Suppose f is convex, and let $h(x) = f(Ax + b)$. Then $\partial h(x) = A^T \partial f(Ax + b)$.

Pointwise maximum

Suppose f is the pointwise maximum of convex functions $f_1, \ldots, f_m$, *i.e.*,

$$f(x) = \max_{i=1,\ldots,m} f_i(x),$$

where the functions f_i are subdifferentiable. We first show how to construct a subgradient of f at x.

Let k be any index for which $f_k(x) = f(x)$, and let $g \in \partial f_k(x)$. Then $g \in \partial f(x)$. In other words, to find a subgradient of the maximum of functions, we can choose one of the functions that achieves the maximum at the point, and choose any subgradient of that function at the point. This follows from

$$f(z) \geq f_k(z) \geq f_k(x) + g^T(z - x) = f(x) + g^T(y - x).$$

More generally, we have

$$\partial f(x) = \mathbf{Co} \cup \{\partial f_i(x) \mid f_i(x) = f(x)\},$$

i.e., the subdifferential of the maximum of functions is the convex hull of union of subdifferentials of the 'active' functions at x.

Example. *Maximum of differentiable functions.* Suppose $f(x) = \max_{i=1,\ldots,m} f_i(x)$, where f_i are convex and differentiable. Then we have

$$\partial f(x) = \mathbf{Co}\{\nabla f_i(x) \mid f_i(x) = f(x)\}.$$

At a point x where only one of the functions, say f_k, is active, f is differentiable and has gradient $\nabla f_k(x)$. At a point x where several of the functions are active, $\partial f(x)$ is a polyhedron.

Example. ℓ_1*-norm.* The ℓ_1-norm

$$f(x) = \|x\|_1 = |x_1| + \cdots + |x_n|$$

is a nondifferentiable convex function of x. To find its subgradients, we note that f can expressed as the maximum of 2^n linear functions:

$$\|x\|_1 = \max\{s^T x \mid s_i \in \{-1, 1\}\},$$

so we can apply the rules for the subgradient of the maximum. The first step is to identify an active function $s^T x$, *i.e.*, find an $s \in \{-1, +1\}^n$ such that $s^T x = \|x\|_1$. We can choose $s_i = +1$ if $x_i > 0$, and $s_i = -1$ if $x_i < 0$. If $x_i = 0$, more than one function is active, and both $s_i = +1$, and $s_i = -1$ work. The function $s^T x$ is differentiable and has a unique subgradient s. We can therefore take

$$g_i = \begin{cases} +1 & x_i > 0 \\ -1 & x_i < 0 \\ -1 \text{ or } +1 & x_i = 0. \end{cases}$$

The subdifferential is the convex hull of all subgradients that can be generated this way:

$$\partial f(x) = \{g \mid \|g\|_\infty \leq 1, g^T x = \|x\|_1\}.$$

Supremum

Next we consider the extension to the supremum over an infinite number of functions, *i.e.*, we consider

$$f(x) = \sup_{\alpha \in \mathcal{A}} f_\alpha(x),$$

where the functions f_α are subdifferentiable. We only discuss the weak property.

Suppose the supremum in the definition of $f(x)$ is attained. Let $\beta \in \mathcal{A}$ be an index for which $f_\beta(x) = f(x)$, and let $g \in \partial f_\beta(x)$. Then $g \in \partial f(x)$. If the supremum in the definition is not attained, the function may or may not be subdifferentiable at x, depending on the index set $\mathcal{A}$.

Assume however that $\mathcal{A}$ is compact (in some metric), and that the function $\alpha \mapsto f_\alpha(x)$ is upper semi-continuous for each x. Then

$$\partial f(x) = \mathbf{Co} \cup \{\partial f_\alpha(x) \mid f_\alpha(x) = f(x)\}.$$

Example. *Maximum eigenvalue of a symmetric matrix.* Let $f(x) = \lambda_{\max}(A(x))$, where $A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n$, and $A_i \in \mathbf{S}^m$. We can express f as the pointwise supremum of convex functions,

$$f(x) = \lambda_{\max}(A(x)) = \sup_{\|y\|_2 = 1} y^T A(x) y.$$

Here the index set $\mathcal{A}$ is $\mathcal{A} = \{y \in \mathbf{R}^n \mid \|y_2\|_1 \leq 1\}$.

Each of the functions $f_y(x) = y^T A(x) y$ is affine in x for fixed y, as can be easily seen from

$$y^T A(x) y = y^T A_0 y + x_1 y^T A_1 y + \cdots + x_n y^T A_n y,$$

so it is differentiable with gradient $\nabla f_y(x) = (y^T A_1 y, \ldots, y^T A_n y)$.

The active functions $y^T A(x) y$ are those associated with the eigenvectors corresponding to the maximum eigenvalue. Hence to find a subgradient, we compute an eigenvector y with eigenvalue $\lambda_{\max}$, normalized to have unit norm, and take

$$g = (y^T A_1 y, y^T A_2 y, \ldots, y^T A_n y).$$

The 'index set' in this example is $\{y \mid \|y\| = 1\}$ is a compact set. Therefore

$$\partial f(x) = \mathbf{Co}\left\{\nabla g_y \mid A(x)y = \lambda_{\max}(A(x))y,\ \|y\| = 1\right\}.$$

Minimization over some variables

The next subgradient calculus rule concerns functions of the form

$$f(x) = \inf_y F(x, y)$$

where $F(x, y)$ is subdifferentiable and jointly convex in x and y. Again we only discuss the weak property.

Suppose the infimum over y in the definition of $f(\hat{x})$ is attained at $y = \hat{y}$, *i.e.*, $f(\hat{x}) = F(\hat{x}, \hat{y})$ and $F(x, \hat{y}) \geq F(\hat{x}, \hat{y})$ for all x. Then there exists a g such that $(g, 0) \in \partial F(\hat{x}, \hat{y})$, and any such g is a subgradient of f at $\hat{x}$.

Strong property. Let x_2 be such that $f(x_1) = F(x_1, x_2)$. Then $\partial f(x_1) = \{g_1 \mid (g_1, 0) \in \partial F(x_1, x_2)\}$, (and the resulting subdifferential is independent of the choice of x_2).

The optimal value function of a convex optimization problem

Suppose $f : \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}$ is defined as the optimal value of a convex optimization problem in standard form, with $z \in \mathbf{R}^n$ as optimization variable,

$$\begin{array}{ll} \text{minimize} & f_0(z) \\ \text{subject to} & f_i(z) \leq x_i, \;\; i = 1, \ldots, m \\ & Az = y. \end{array} \tag{15}$$

In other words, $f(x, y) = \inf_z F(x, y, z)$ where

$$F(x, y, z) = \begin{cases} f_0(z) & f_i(z) \leq x_i, \; i = 1, \ldots, m, \; Az = y \\ +\infty & \text{otherwise,} \end{cases}$$

which is jointly convex in x, y, z. Subgradients of f can be related to the dual problem of (15) as follows.

Suppose we are interested in subdifferentiating f at $(\hat{x}, \hat{y})$. We can express the dual problem of (15) as

$$\begin{array}{ll} \text{maximize} & g(\lambda) - x^T\lambda - y^T\nu \\ \text{subject to} & \lambda \succeq 0. \end{array} \tag{16}$$

where

$$g(\lambda) = \inf_z \left(f_0(z) + \sum_{i=1}^m \lambda_i f_i(z) + \nu^T Az \right).$$

Suppose strong duality holds for problems (15) and (16) at $x = \hat{x}$ and $y = \hat{y}$, and that the dual optimum is attained at $\lambda^\star$, $\nu^\star$ (for example,, because Slater's condition holds). From the inequality (**??**) we know that

$$f(x, y) \geq f(\hat{x}, \hat{y}) - \lambda^{\star T}(x - \hat{x}) - \nu^{\star T}(y - \hat{y})$$

In other words, the dual optimal solution provides a subgradient:

$$-(\lambda^\star, \nu^\star) \in \partial f(\hat{x}, \hat{y}).$$

4.4 Quasigradients

If $f(x)$ is quasiconvex, then g is a *quasigradient* at x_0 if

$$g^T(x - x_0) \geq 0 \Rightarrow f(x) \geq f(x_0),$$

Geometrically, g defines a supporting hyperplane to the sublevel set $\{x \mid f(x) \leq f(x_0)\}$. Note that the set of all quasigradients at x_0 form a cone.

> **Example.** *Linear fractional function.* $f(x) = \frac{a^T x + b}{c^T x + d}$. Let $c^T x_0 + d > 0$. Then $g = a - f(x_0)c$ is a quasigradient at x_0. If $c^T x + d > 0$, we have
>
> $$a^T(x - x_0) \geq f(x_0)c^T(x - x_0) \Longrightarrow f(x) \geq f(x_0).$$

Example. *Degree of a polynomial.* Define $f : \mathbf{R}^n \to \mathbf{R}$ by

$$f(a) = \min\{i \mid a_{i+2} = \cdots = a_n = 0\},$$

i.e., the degree of the polynomial $a_1 + a_2 t + \cdots + a_n t^{n-1}$. Let $a \neq 0$, and $k = f(a)$, then $g = \operatorname{sign}(a_{k+1}) e_{k+1}$ is a quasigradient at a

To see this, we note that

$$g^T(b - a) = \operatorname{sign}(a_{k+1}) b_{k+1} - |a_{k+1}| \geq 0$$

implies $b_{k+1} \neq 0$.

5 Examples and applications

We first give some examples of typical applications and interpretations of cutting-plane methods. We then describe a general class of problems where cutting-plane methods are useful: we show that cutting-plane methods can be used to solve large convex optimization problems by solving several smaller subproblems in sequence or in parallel.

5.1 Examples

Resource allocation

An organization (for example, a company) consists of N different units (for example, manufacturing plants) that operate with a large degree of autonomy, but share k resources (for example, budget, raw material, workforce). The amounts of each resource used by the different units is denoted by N vectors $y_i \in \mathbf{R}^k$, which must satisfy the constraint

$$\sum_{i=1}^{N} y_i \preceq Y,$$

where $Y \in \mathbf{R}^k$ gives the total available amount of each resource. The revenue R_i generated by unit i depends on some local decision variables $x_i \in \mathbf{R}^{n_i}$, that the unit can select independently of the other units, and on the amount y_i of the shared resources the unit uses. The total revenue of the organization is the sum of the revenues R_i of the individual units. The operation of the entire organization can be described by the variables $x_1, \ldots, x_N$, $y_1, \ldots, y_N$. The optimal values of these variables are the solution of the optimization problem

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{N} R_i(x_i, y_i) \\ \text{subject to} & \sum_{i=1}^{N} y_i \preceq Y. \end{array} \tag{17}$$

We will assume that the revenue functions R_i are concave and differentiable. Different algorithmic schemes are possible for the solution of (17), and can be interpreted as different management strategies.

A first possibility is to collect a complete description of the revenue functions R_i at some central location, solve the problem (17) by jointly optimizing over all the variables, and then

communicate the optimal values x_i, y_i to the units. In this scheme, all decisions are made centrally.

A second possibility is to use a decentralized decision process in two levels. A supervising unit is responsible for allocating resources y_i to the units, making sure that the limits Y are not exceeded. Each unit determines the value of its decision variables x_i locally, by maximizing its revenue for the value of y_i assigned to it, *i.e.*, by solving the problem

$$\text{maximize } R_i(x_i, y_i) \tag{18}$$

over the variable x_i, for *given* y_i. These local problems can be solved by any convex optimization algorithm. The question is: how should the supervising unit allocate the resources, and is it possible to make an optimal resource allocation without knowing the revenue functions R_i? We will see that cutting-plane methods allow us to implement this idea in a rigorous way.

We define N concave functions $r_i : \mathbf{R}^k \to \mathbf{R}$ by

$$r_i(y) = \sup_{x_i} R_i(x_i, y),$$

i.e., $r_i(y_i)$ is the optimal value of (18). Using these functions, we can express the global problem (17) as

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^N r_i(y_i) \\ \text{subject to} & \sum_{i=1}^N y_i \preceq Y. \end{array} \tag{19}$$

To solve this problem by a cutting-plane method, we need to be able to calculate the value and a subgradient of $-r_i$ at any given point $\bar{y}_i$. This can be achieved by numerically solving the unconstrained problem (18): If the optimal value of x_i is $\bar{x}_i$, then $r_i(\bar{y}_i) = -R_i(\bar{x}_i, \bar{y}_i)$, and a subgradient of $-r_i$ at $\bar{y}_i$ is given by $\nabla_{y_i} R_i(\bar{x}_i, \bar{y}_i)$. This last fact follows from our assumption that R_i is a concave and differentiable function: for all x_i, y_i:

$$\begin{array}{rcl} -R_i(x_i, y_i) & \geq & -R_i(\bar{x}_i, \bar{y}_i) - \nabla_{y_i} R_i(\bar{x}_i, \bar{y}_i)^T (y_i - \bar{y}_i) \\ & = & -r_i(\bar{y}_i) - \nabla_{y_i} R_i(\bar{x}_i, \bar{y}_i)^T (y_i - \bar{y}_i) \end{array}$$

(note that $\nabla_{x_i} R_i(\bar{x}_i, \bar{y}_i) = 0$), and therefore

$$-r_i(y_i) = \inf_{x_i} -R_i(x_i, y_i) \geq -r_i(\bar{y}_i) - \nabla_{y_i} R_i(\bar{x}_i, \bar{y}_i)^T (y_i - \bar{y}_i),$$

i.e., $-\nabla_{y_i} R_i(\bar{x}_i, \bar{y}_i)$ is a subgradient. In summary, we can solve problem (19) iteratively using a cutting-plane algorithm. At each iteration we find the value and a subgradient of $-r_i(y_i)$ by solving the local problems (18). In this scheme the problem data (*i.e.*, the functions R_i) can be distributed over the N units. The cutting-plane algorithm does not need a complete description of the functions R_i. Instead, it builds a piecewise-linear approximation, based on the function values and the subgradients.

Note that in this interpretation, we can interpret the subgradients as prices:

$$(\nabla_{y_i} R_i(\bar{x}_i, \bar{y}_i))_j$$

(which presumably would be nonnegative, *i.e.*, an increase in a resource level does not decreases the revenue), as the price that unit i is willing to pay per unit increase of resource j (or, equivalently, the price at which it is willing to sell the resource).

This leads to a third possible scheme, related to the dual of problem (19). The dual function of the corresponding minimization problem is

$$g(\lambda) = \sum_{i=1}^{N} \inf_{y_i}(-r_i(y_i) + \lambda^T y_i) - \lambda^T Y = -\sum_{i=1}^{N} r_i^*(\lambda) - \lambda^T Y$$

where r_i^* is defined as

$$r_i^*(\lambda) = \sup_{y_i} (r_i(y_i) - \lambda^T y_i) = \sup_{x_i, y_i} (R_i(x_i, y_i) - \lambda^T y_i).$$

(In other words, $r_i^*(-\lambda)$ is the conjugate of $-r_i$.) the dual of problem (19) is

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} r_i^*(\lambda) + \lambda^T Y \\ \text{subject to} & \lambda \succeq 0, \end{array} \tag{20}$$

which is a convex minimization problem with variable $\lambda \in \mathbf{R}^k$. Again, we can envision a decentralized solution method at two levels. At the global level, we solve (20) using a cutting-plane method. At each iteration we need to evaluate the value and a subgradient for the objective function at the current iterate $\bar{\lambda}$. To evaluate $r_i^*(\bar{\lambda})$ we solve the problem

$$\text{maximize } R_i(x_i, y_i) - \bar{\lambda}^T y_i$$

over x_i, y_i. The maximizers $\bar{x}_i$, $\bar{y}_i$ must satisfy

$$\nabla_{y_i} R_i(\bar{x}_i, \bar{y}_i) = \lambda, \quad \nabla_{x_i} R_i(\bar{x}_i, \bar{y}_i) = 0.$$

From $\bar{x}_i$, $\bar{y}_i$, we obtain the function value of r_i^* at $\bar{\lambda}$, *i.e.*, $r_i^*(\bar{\lambda}) = R_i(\bar{x}_i, \bar{y}_i) - \bar{\lambda}^T \bar{y}_i$. A subgradient is given by $-\bar{y}_i$, since by definition

$$r_i^*(\lambda) \geq R_i(x_i, y_i) - \lambda^T y_i$$

for all λ, x_i, y_i, and therefore,

$$r_i^*(\lambda) \geq R_i(\bar{x}_i, \bar{y}_i) - \lambda^T \bar{y}_i = r_i^*(\bar{\lambda}) - \bar{y}_i^T(\lambda - \bar{\lambda}).$$

The dual variables λ can be interpreted as resource prices. In this scheme, a central unit sets prices λ, determined iteratively by applying a cutting-plane method to (20). Based on the resource prices, the N units determine the amount y_i of the resources they purchase, and the optimal values of the local decision variables. They communicate back to the central unit the function values $r_i^*(\lambda) = R_i(\bar{x}_i, \bar{y}_i) - \bar{\lambda}^T \bar{y}_i$, *i.e.*, their revenue minus cost of resource purchase, and the amounts y_i of the resources they purchase. Based on this information, the resource prices are adjusted.

5.2 Decomposition

We now discuss general methods for decomposing nearly separable optimization problems into smaller problems that can be solved independently. We will consider a standard convex optimization problem of the form

$$\begin{array}{ll} \text{minimize} & f_0(x,y) + \tilde{f}_0(z,y) \\ \text{subject to} & f_i(x,y) \leq 0,\ i = 1,\ldots,m \\ & \tilde{f}_i(z,y) \leq 0,\ i = 1,\ldots,\tilde{m} \\ & A_1 x + A_2 y = b \\ & \tilde{A}_1 z + \tilde{A}_2 y = \tilde{b} \end{array} \tag{21}$$

with variables $x \in \mathbf{R}^n$, $z \in \mathbf{R}^{\tilde{n}}$, and $y \in \mathbf{R}^k$. It is clear that if y were fixed, the problem would decompose into two separate problems: a first problem

$$\begin{array}{ll} \text{minimize} & f_0(x,y) \\ \text{subject to} & f_i(x,y) \leq 0,\ i = 1,\ldots,m \\ & A_1 x + A_2 y = b, \end{array} \tag{22}$$

with x as variable, and a second problem

$$\begin{array}{ll} \text{minimize} & \tilde{f}_0(z,y) \\ \text{subject to} & \tilde{f}_i(z,y) \leq 0,\ i = 1,\ldots,\tilde{m} \\ & \tilde{A}_1 z + \tilde{A}_2 y = \tilde{b} \end{array} \tag{23}$$

with z as variable. For this reason, we call y the *complicating* or *coupling* variable. We are interested in formulating an algorithm that takes advantage of the structure in the problem, and solves problem (21) by solving a sequence of problems (25) and (23) for different values of y. We will see that the cutting-plane method offers a systematic way to implement this idea. We will discuss two formulations.

Differentiable objective and constraint functions

Problem (21) can be written as

$$\text{minimize } p^\star(y) + \tilde{p}^\star(y) \tag{24}$$

where $p^\star(y)$ and $\tilde{p}^\star(y)$ are defined as the optimal values of problems (25) and (23), respectively. From chapter **??**, we know that both functions are convex in y (but not necessarily differentiable). We can solve problem (24) by a cutting-plane method, provided we can efficiently evaluate the value and a subgradient of $p^\star$ and $\tilde{p}^\star$, at any given y. This can be achieved by solving the optimization problems that define $p^\star$ and $\tilde{p}^\star$, and the corresponding dual problems. We will give the details for $p^\star$. The dual function of (25) is

$$g(y,\lambda,\nu) = \inf_x L(x,y,\lambda,\nu)$$

where

$$L(x,y,\lambda,\nu) = f_0(x,y) + \sum_{i=1}^m \lambda_i f_i(x,y) + \nu^T(A_1 x + A_2 y - b).$$

To evaluate $p^\star(\bar{y})$, we solve the problem

$$\begin{array}{ll} \text{minimize} & f_0(x, \bar{y}) \\ \text{subject to} & f_i(x, \bar{y}) \leq 0,\ i = 1, \ldots, m \\ & A_1 x + A_2 \bar{y} = b \end{array} \tag{25}$$

with variable x. We first consider the case where the problem is feasible. We also assume that the primal and dual optima are attained and equal. Let $\bar{x}$ be primal optimal, and $\bar{\lambda}$, $\bar{\nu}$ be dual optimal. We will show that

$$\nabla_y L(\bar{x}, \bar{y}, \bar{\lambda}, \bar{\nu}) = \nabla_y f_0(\bar{x}, \bar{y}) + \sum_{i=1}^m \bar{\lambda}_i \nabla_y f_i(\bar{x}, \bar{y}) + A_2^T \bar{\nu}$$

is a subgradient for $p^\star$ at $\bar{y}$. To prove this, we first note that

$$\nabla_x L(\bar{x}, \bar{y}, \bar{\lambda}, \bar{\nu}) = 0$$

as a consequence of the KKT conditions for problem (25). By convexity of the Lagrangian L (jointly in x and y), we have, for all x, y, $\lambda \succeq 0$, and ν,

$$\begin{array}{rcl} L(x, y, \lambda, \nu) & \geq & L(\bar{x}, \bar{y}, \lambda, \nu) + \nabla_y L(\bar{x}, \bar{y}, \lambda, \nu)^T (y - \bar{y}) \\ \inf_x L(x, y, \lambda, \nu) & \geq & L(\bar{x}, \bar{y}, \lambda, \nu) + \nabla_y L(\bar{x}, \bar{y}, \lambda, \nu)^T (y - \bar{y}) \end{array}$$

and therefore, by weak duality,

$$p^\star(y) \geq L(\bar{x}, \bar{y}, \lambda, \nu) + \nabla_y L(\bar{x}, \bar{y}, \lambda, \nu)^T (y - \bar{y}).$$

Evaluating both sides of the inequality at $\bar{\lambda}$, $\bar{\nu}$ yields

$$p^\star(y) \geq p^\star(\bar{y}) + \nabla_y L(\bar{x}, \bar{y}, \bar{\lambda}, \bar{\nu})^T (y - \bar{y}),$$

which shows that $\nabla_y L(\bar{x}, \bar{y}, \bar{\lambda}, \bar{\nu})$ is a subgradient.

If the problem (25) is infeasible, *i.e.*, $p^\star(\bar{y}) = +\infty$, we can apply a theorem of alternatives and conclude that there exist $\bar{\lambda} \succeq 0$, $\bar{\nu}$, not both zero, such that

$$\inf_x G(x, \bar{y}, \bar{\lambda}, \bar{\nu}) > 0 \tag{26}$$

where

$$G(x, y, \lambda, \nu) = \sum_{i=1}^m \lambda_i f_i(x, y) + \nu^T (A_1 x + A_2 y - b).$$

Suppose the infimum in (26) is attained at $\bar{x}$, *i.e.*, $\nabla_x G(\bar{x}, \bar{y}\bar{\lambda}\bar{\nu}) = 0$. By convexity of G,

$$\begin{array}{rcl} G(x, y, \lambda, \nu) & \geq & G(\bar{x}, \bar{y}, \lambda, \nu) + \nabla_y G(\bar{x}, \bar{y}, \lambda, \nu)^T (y - \bar{y}) \\ \inf_x G(x, y, \lambda, \nu) & \geq & G(\bar{x}, \bar{y}, \lambda, \nu) + \nabla_y G(\bar{x}, \bar{y}, \lambda, \nu)^T (y - \bar{y}) \\ & > & \nabla_y G(\bar{x}, \bar{y}, \lambda, \nu)^T (y - \bar{y}). \end{array}$$

Therefore, if

$$\nabla_x G(x, \bar{y}, \bar{\lambda}, \bar{\nu})^T (y - \bar{y}) \geq 0, \tag{27}$$

then $\inf_x G(x, y, \bar{\lambda}, \bar{\nu}) > 0$ for all x, y, *i.e.*, $y \notin \mathbf{dom}\, p^\star$. In other words, (27) defines a neutral cut for the feasible set of (24).

Problems that are linear in the complicating variable

Here we consider the special case (21) that arises when the objective function is separable and the constraint functions are linear in the coupling variable:

$$\begin{array}{ll} \text{minimize} & f_0(x) + \widetilde{f}_0(z) + \widehat{f}_0(y) \\ \text{subject to} & f_i(x) + h_i^T y \leq 0,\ i = 1, \ldots, m \\ & \widetilde{f}_i(z) + \widetilde{h}_i^T y \leq 0,\ i = 1, \ldots, \widetilde{m} \\ & A_1 x + A_2 y = b \\ & \widetilde{A}_1 z + \widetilde{A}_2 y = \widetilde{b}. \end{array} \tag{28}$$

Remark. Note that this assumption can be made without loss of generality. We can rewrite (21) by introducing auxiliary variables w and $\tilde{w}$,

$$\begin{array}{ll} \text{minimize} & f_0(x, w) + \widetilde{f}_0(z, \tilde{w}) \\ \text{subject to} & f_i(x, w) \leq 0,\ i = 1, \ldots, m \\ & \widetilde{f}_i(z, \widetilde{w}) \leq 0,\ i = 1, \ldots, \widetilde{m} \\ & A_1 x + A_2 w = b \\ & \widetilde{A}_1 z + \widetilde{A}_2 \widetilde{w} = \widetilde{b} \\ & w = y,\ \ \widetilde{w} = y. \end{array}$$

This problem has two sets of variables (x, w) and $(z, \widetilde{w})$, coupled by coupling variable y. The constraints are linear in the coupling variables.

We can rewrite (28) as

$$\text{minimize } p^\star(Hy, A_2 y) + \widetilde{p}^\star(\widetilde{H} y, \widetilde{A}_2 y) + \widehat{f}_0(y) \tag{29}$$

where $p^\star(u, v)$ is the optimal value of

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) + u_i \leq 0,\ i = 1, \ldots, m \\ & A_1 x + v = b \end{array} \tag{30}$$

and $\widetilde{p}^\star(u, v)$ is the optimal value of

$$\begin{array}{ll} \text{minimize} & \widetilde{f}_0(z) \\ \text{subject to} & \widetilde{f}_i(z) + u_i \leq 0,\ i = 1, \ldots, m \\ & \widetilde{A}_1 z + v = \widetilde{b}. \end{array} \tag{31}$$

If we solve (29) using a cutting-plane method, we need at each iteration the function values and the subgradients of $p^\star$ and $\widetilde{p}^\star$. As described above, we can achieve this by solving (30) and (31) and the corresponding dual problems.

Exercises

1. *A convex function that is not subdifferentiable.* Verify that the following function $f : \mathbf{R} \to \mathbf{R}$ is not subdifferentiable at $x = 0$:

$$f(x) = \begin{cases} x & x > 0 \\ 1 & x = 0 \\ +\infty & x < 0. \end{cases}$$

2. Explain how you would calculate a subgradient for the following convex functions $f : \mathbf{R}^n \to \mathbf{R}$.

   (a) $f(x) = \sup_{-1 \le t \le 1}(x_1 + x_2 t + \cdots + x_n t^{n-1})$.

   (b) $f(x) = \|Ax + b\|_\infty$, where $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$.

   (c) $f(x) = \sup_{Ay \preceq b} y^T x$. (You can assume that the polyhedron defined by $Ay \preceq b$ is bounded.)

3. We consider the ℓ_1-norm minimization problem

$$\text{minimize} \quad \left\| \begin{bmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right\|_1 \tag{32}$$

   with variables $x_1 \in \mathbf{R}^{n_1}$, $x_2 \in \mathbf{R}^{n_2}$, and $y \in \mathbf{R}^p$. This problem is readily formulated and solved as an LP

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T w_1 + \mathbf{1}^T w_2 \\ \text{subject to} & -w_1 \preceq A_1 x_1 + B_1 y + b_1 \preceq w_1 \\ & -w_2 \preceq A_2 x_2 + B_2 y + b_2 \preceq w_2 \end{array}$$

   with variables w_1, w_2, x_1, x_2, and y.

   Here we will develop an alternative approach that is interesting when p is small and n_1 and n_2 are large. Intuition suggests that we should be able to take advantage of the fact that the problem is almost separable: if we knew the optimal y, the problem would decompose into two smaller problems,

$$\text{minimize} \quad \|A_1 x_1 + B_1 y + b_1\|_1 \tag{33}$$

   with variable x_1, and

$$\text{minimize} \quad \|A_2 x_2 + B_2 y + b_2\|_1 \tag{34}$$

   with variable x_2. We will apply convex duality and the ellipsoid method to make this idea precise.

(a) Use LP duality to show that the optimal value of

$$\text{minimize} \quad \|Ax + By + b\|_1 \tag{35}$$

with variable x, is equal to the optimal value of

$$\begin{array}{ll} \text{maximize} & -(By+b)^T z \\ \text{subject to} & A^T z = 0 \\ & \|z\|_\infty \leq 1 \end{array} \tag{36}$$

with variable z. In both problems we assume A, B, y, and b are given.

Let $p^\star(y)$ be the optimal value of (35) as a function of y. Is $p^\star(y)$ a convex function?

Suppose we choose a value $y = \hat{y}$ and solve problem (35) to evaluate $p^\star(\hat{y})$, using an algorithm that also returns the dual optimal solution $z^\star$ in (5). Show that $-B^T z^\star$ is a subgradient for $p^\star(y)$ at $\hat{y}$, *i.e.*,

$$p^\star(y) \geq p^\star(\hat{y}) - (B^T z^\star)^T (y - \hat{y})$$

for all y.

(b) We now return to (32). We can express the problem as

$$\text{minimize } p_1^\star(y) + p_2^\star(y), \tag{37}$$

where $p_1^\star(y)$ and $p_2^\star(y)$ are the optimal values of (33) and (34), respectively.

Develop a matlab code that solves (37) using ACCPM.