PRINCETON UNIV. F'18 COS 521: ADVANCED ALGORITHM DESIGN

Lecture 17: Interior Point Methods

Lecturer: *Christopher Musco*

1 Convex Optimization

Recall that we are interested in solving problems of the form:

$$\min f(x) \text{ for } x \in \mathcal{K},$$

where f is a convex function and $\mathcal{K}$ is a convex set.

We have already analyzed gradient descent for this problem, which used a projection oracle for $\mathcal{K}$ and a gradient oracle for f to provide relatively low-accuracy solutions. The **Ellipsoid Method** used a separation oracle to provide higher accuracy solutions, but typically at a greater computational cost.

1.1 Linear programming

Today we are interested in the important special case of linear programming where:

$$f(x) = c^T x, \qquad\qquad \mathcal{K} = \{x \mid Ax \geq b\}$$

for $A \in \mathbb{R}^{n\times d}, c \in \mathbb{R}^d, b \in \mathbb{R}^n$.

The Ellipsoid Method was developed through the efforts of many mathematicians: Shor, Yudin, Nemirovskii, and others. However, if first gained a lot of attention when Katchiyan showed in 1979 [1] that it could be used to solve linear programs in polynomial time (with a dependency on L, the number of bits required to specify A, b, and c.)

As the first polynomial time solution for this ubiquitous problem, this discovery was so important that it was reported in the New York Times! However, the Ellipsoid Method did not immediately impact how linear programs were solved in practice. It is much slower than the Simplex method, which runs in exponential time in the worst case, but performs very well in practice[1].

Nevertheless, the Ellipsoid method was very influential theoretically – it helped launch a flurry of work on linear programming, which ultimately lead to a completely different polynomial time solution for the problem. In 1984, Karmarkar developed an **Interior Point Method** for solving linear programs [2], which was asymptotically faster than ellipsoid and, more importantly, performed much better in practice (it also made the New York Times). While it did not immediately unseat Simplex, after a few decades of improvement, the fastest linear programming libraries today are typically based on interior point methods.

[1]There has been a lot of interest in understanding why Simplex performs much better than its worst case complexity suggests. This question lead to influential work on "smoothed analysis" of algorithms [3].

2 High Level Strategy

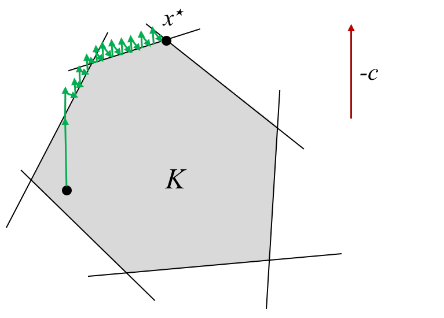
Interior point methods are based on a very different strategy than the Ellipsoid method. The idea is to start *inside* the convex set $\mathcal{K}$ – i.e. to start at an "interior point".

2.1 Aside: how to start inside the polytope

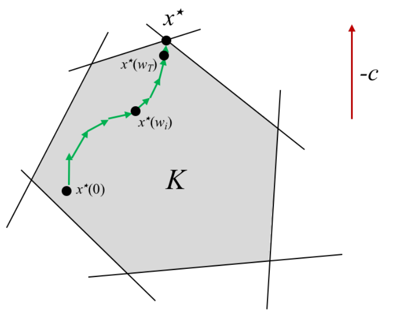
This might seem problematic – when analyzing the ellipsoid method, we reduced the entire problem to finding a point in $\mathcal{K}$. Shouldn't this be a hard problem? The trick is use another linear program, which has a trivial interior point, to find a starting point for our original linear program. To solve the original linear program, we will first solve this "Phase 1" linear program to find a starting point. In particular, consider:

$$\begin{aligned} &\min t \quad \text{s.t.} \\ &Ax \geq (1-t)b, \\ &t \geq 0. \end{aligned}$$

If the optimal solution $t^\star > 0$ then our original linear program is infeasible. If $t^\star = 0$ then $x^\star$ satisfies $Ax^\star \geq b$, so is a starting point inside our original polytope. At the same time, this new linear program has a trivial starting point: $(x, t) = (\vec{0}, 1)$, so it can be solved using an interior point method, or any other method that needs a starting point in $\mathcal{K}$.



(a) Gradient Descent optimization path.



(b) Ideal Interior Point optimization path.

Figure 1: The goal of interior point methods is to move towards an optimal solution while avoiding immediately approaching the boundary of $\mathcal{K}$.

Okay, so we have an interior point. To get some intuition, let's think about how gradient descent would begin optimizing from this point. Note that:

$$\nabla f(x) = c \text{ for all } x$$

As depicted in Figure 1, following the gradient would thus cause use to move straight in the diection of $-c$, until we hit a face of $\mathcal{K}$, at which point we would move outside of the polytope and need to project back. As shown, the boundary makes it very hard to use gradient information to make rapid progress. The Simplex method avoids this issue by jumping along edges of the polytope from one vertex to the other, but unfortunately, choosing which direction to not obvious and poor choices can lead to cycling, stalling, and ultimately an exponential number of steps.

Interior point methods, on the other hand, try to avoid reaching the boundary of $\mathcal{K}$ entirely (at least until they are nearly done). The idea is to still proceed in iterations, but if we examine the path of iterates, our approximate solutions move more directly towards the optimal, all while staying inside $\mathcal{K}$.

3 The Barrier Function

To do so, interior point methods solve a *sequence* of (slightly) changing optimization problems. These problems replace the hard constraint that $x \in \mathcal{K}$ with a "smoother" objective function term that increases as x gets closer and closer to the boundary of $\mathcal{K}$. In particular, we solve the intermediate problem:

Problem 1 (Smoothly constrained linear optimization)**.** *For scalar $w \geq 0$ and convex function B with $B(x) \to \infty$ as $x \to \partial\mathcal{K}$ (we use ∂K to denote the boundary of $\mathcal{K}$)*

$$\min F_w(x) = wc^T x + B(x).$$

A typical choice for B is the "log barrier" $B(x) = \sum_{i=1}^{n} -\log(a_i^T x - b_i)$.

Note that, since both $B(x)$ and $f(x)$ are convex, $F_w(x)$ is always convex for $w \geq 0$.

Let $x^\star(w)$ be the optimal solution for F_w. $x^\star(0)$ is called the "analytic center" of the polytope $\mathcal{K}$. It has nothing to do with c at all. As w approaches infinity, $x^\star(w)$ converges towards the optimal solution for the linear program.

The curve traced by $x^\star(w)$ for $w = 0 \to \infty$ is called the "central path". A key observation is that, for a small change Δ, $x^\star(w + \Delta)$ is close to $x^\star(w)$. We will take advantage of this property by solving F_w for slowly increasing values of w, starting with $w = 0$. For each value of w, F_w will itself be solved via an iterative method, but initialized with the optimal solution for our previous value of w, which ensures rapid convergence.

This yields the following outline for the interior point method that we will analyze:

Interior Point Method:

1. Start at (or really close to) $x^\star(0)$.
2. Choose a sequence of geometrically increasing values $w_1, \ldots, w_T$ with $w_{i+1} = (1+q)w_i$.
3. For $i = 1, \ldots, T$, use **Newton**'s method to find $x^\star(w_i) = \arg\min F(w_i)$, using $x^\star(w_{i-1})$ as a starting point.[2]
4. Return $x^\star(w_T)$.

We do not discuss the first initialization step in depth, although I will point to references with more details at the end of the lecture.

[2]In reality, we will only be able to find each $x^\star(w_i)$ approximately. Dealing with this detail complicates the analysis, without changing it substantial, but again more details can be found in [4].

Since our choices for w increase geometrically, the number of iterations T will ultimately depend on $\log(w_T/w_1)$. If we choose $w_T = n/\epsilon$, we get a solution within $c^T[x^\star(w_T)] \leq c^T x^\star + \epsilon$, where $x^\star$ is the true optimum for the linear program. See [4] or http://people.seas.harvard.edu/~cs224/fall14/lec/lec17.pdf for a short proof. Moreover, for linear programs with bounded integer inputs which take a total of L bits to represent, we can set $w_1 = 2^{-O(L)}$. Overall, this will lead to an iteration complexity of $O\left(\log_{1+q}(2^L n/\epsilon)\right)$. For small q, this approximately equals $O\left(\frac{1}{q} \cdot L\log(n/\epsilon)\right)$.

4 Newton's Method

The "inner loop" of our interior point method uses Newton's method to optimize the unconstrained convex function $F_w(x)$. This is good method to know about in general and it is used well beyond interior point methods. Its distinction from gradient descent is that Newton's method uses *second order* information about f. This allows it to converge much faster than gradient descent, even when the function we are optimizing is poorly behaved (e.g. $\|\nabla f(x)\|_2$ is huge). This will be the case for our objective + barrier function, which has gradient:

$$\nabla F_w(x) = \nabla\left[c^T x - \sum_{i=1}^n \log(a_i^T x - b_i)\right] = wc + \sum_{i=1}^n \frac{1}{(a_i^T x - b_i)} \cdot a_i.$$

This gradient blows up as x gets close the boundary of $\mathcal{K}$.

Newton's method is "immune" to this blow up, but with one caveat: it typically only converges is we start pretty close to the optimum of our function. This makes it well tailored to our interior point scheme, where we obtain a good starting point for optimizing F_{w_i} from our solution to $\min F_{w_{i-1}}$.[3]

4.1 The Hessian

Newton's method uses second order information in the form of the *Hessian* of the function it's trying to minimize. Recall that the gradient of a function f is:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}$$

[3]There are modified versions of Newton's method that are "globally convergent" for convex functions, meaning that they can be initialized with any value and still converge (just like gradient descent). However, there's a trade off – these versions require at least some bounds on how well behaving f's gradient is.

I.e. the gradient is a vector containing all partial first derivatives of f. The Hessian, $\nabla^2 f(x)$ is a $d \times d$ matrix containing all partial second derivatives of f:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_d \partial x_d} \end{bmatrix}$$

One way to think about the Hessian is that is measures how the gradient changes. For small z, $\nabla f(x+z) \approx \nabla f(x) + [\nabla^2 f(x)]z$.

For convex functions $\nabla f(x)$ is positive semidefinite for all x. I.e.

$$y^T \nabla f(x) y \geq 0 \qquad\qquad \forall y.$$

Exercise 1. *Prove this. And also prove that, if $\nabla f(x)$ is positive semidefinite for all x, then f is convex.*

Let's look at an example:

$$f(x) = \|Ax - b\|_2^2 = x^T A^T A x - 2b^T A x + b^T b$$

We have:

$$\nabla f(x) = 2A^T A x - 2b^T A$$

$$\nabla^2 f(x) = 2A^T A$$

$2A^T A$ is positive semidefinite because $2z^T A^T A z = 2\|Az\|_2^2 \geq 0$ for all z.

For the function we're looking at today we have:

$$\nabla F_w(x) = wc + \sum_{i=1}^{n} \frac{1}{(a_i^T x - b_i)} \cdot a_i$$

$$\nabla^2 F_w(x) = \sum_{i=1}^{n} \frac{1}{(a_i^T x - b_i)^2} \cdot a_i a_i^T.$$

To see that $\nabla^2 F_w(x)$ is PSD, note that it can be written as ADA^T for a positive diagonal matrix D.

4.2 Iteration

We are ready to define a standard version of Newton's method:

Newton's Method

- Choose initialization points x_0.

- For $i = 1, \dots T$, set $x_{i+1} = x_i - [\nabla^2 f(x_i)]^{-1} \nabla f(x_i)$

This method should look very similar to gradient descent. It proceeds by subtracting a term depending on $\nabla f(x_i)$ from x_i. However, this direction is "modified" by multiplying is by the matrix $[\nabla^2 f(x_i)]^{-1}$. Doing so ultimately provides a better search direction than gradient descent.

In the most extreme case, consider $f(x) = \|Ax - b\|_2^2$. Suppose we start at $x_0 = \vec{0}$. $\nabla f(\vec{0}) = -2b^T A$ and $[\nabla^2 f(x_i)]^{-1} \nabla f(x_i) = (2A^T A)^{-1} 2A^T b = (A^T A)^{-1} A^T b$, which is the optimal solution for ℓ_2 regression. So we solve the problem in one step using Newton's method! This does not happen for general functions, but there is typically a lot to gained by looking at the Hessian.

There is of course a cost: computing and inverting $\nabla^2 f(x_i)$ at each step of our iterative method can be much more expensive than e.g. a gradient step. For the problems considered here, $\nabla^2 f(x_i)$ takes $O(nd^2)$ time to construct and $O(d^3)$ time to invert. This cost needs to be offset by a fast convergence rate.

4.3 Convergence

There are many ways to measure convergence of an iterative method. When analyzing gradient descent we showed that for an approximate optimum $\tilde{x}$:

$$|f(\tilde{x}) - f(x^\star)| \leq \epsilon.$$

We could also have asked that

$$\|\tilde{x} - x^\star\| \leq \epsilon,$$

or even that

$$\|\nabla f(\tilde{x})\| \leq \epsilon,$$

since $\|\nabla f(x^\star)\| = 0$ for any optimum $x^\star$. While these metrics may not be directly comparable, they are all valid ways of understanding how close $\tilde{x}$ is to optimal.

One common way of analyzing Newton's method considers convergence in a matrix known as the "Newton's decrement":

Definition 1 (Newton Decrement). *The Newton decrement of a function f at point x, $\lambda_f(x)$, is defined:*

$$\lambda_f(x) = \sqrt{\nabla f(x)^T [\nabla^2 f(x)]^{-1} \nabla f(x)}$$

Since $\nabla f(x^\star) = 0$, we have that $\lambda_f(x^\star) = 0$ at any minimizer $x^\star$. We can also write $\lambda_f(x)^2 = \|[\nabla^2 f(x)]^{-1} \nabla f(x)\|_{\nabla^2 f(x)}^2$, so it's a measure of how small our gradient is, which intuitively is a good measure of how close we are to a minimizer.

To establish convergence in the Newton's decrement metric, we need to make an additional assumption on the function f we are optimizing, in addition to its convexity:

Theorem 2 (Convergence of Newton's method)**.** *If for all x, $\nabla^3 f(x)[h,h,h] \leq 2\left(h^T \nabla^2 f(x) h\right)^{3/2}$ (this property is called self-concordance) for all h, then:*

$$\lambda_f(x - [\nabla^2 f(x)]^{-1}\nabla f(x)) \leq \left(\frac{\lambda_f(x)}{1-\lambda_f(x)}\right)^2$$

Note that for our smoothly constrained linear programs, $\nabla^3 F_w$ and $\nabla^2 F_w$ only depend on the barrier function B, so our assumption applied to interior point methods will actually be that B is "self-concordant".

The self-concordance property bounds the size of the third derivative of a function, which relates to the smoothness of $\nabla^2 f(x)$. This makes sense – for second order information to be useful in quickly progressing towards an optimum, it should be relatively consistent as x changes. Above we consider the extreme case of $f(x) = \|Ax-b\|_2^2$, which could be solved in one step of Newton's method. In this case, $\nabla^3 f(x) = 0$, so the second order information is unchanging with x.

If self-concordance holds, Theorem 2 gives a very strong **local convergence** guarantee. If we start with $\lambda_f < 1$ (e.g. $\lambda_f = 1/4$) we're basically squaring our error every time. This leads to very fast convergence. In $O(\log\log(1/\epsilon))$ steps we go from $O(1)$ error to ϵ error. This is called quadratic convergence.

5 Interior Point Updates

To analyze our interior point method, we should like to apply Theorem 2. Doing so requires arguing that $x^\star_{w_i}$ gives a good initialization point for minimzing $F_{w_{i+1}}$. In particular, since we update w by multiplying it by $1+q$ for some $q \geq 0$, we want to find the largest q such that:

$$\lambda_{F_{(1+q)w}}(x^\star_w) \leq \frac{1}{4}.$$

The particular choice of $1/4$ here is arbitrary – for Theorem 2 to ensure rapid convergence, we just need that $\lambda_{F_{(1+q)w}}(x^\star_w) \leq 1/2 - c$ for some constant c. First we note that

$$\nabla F_{qw}(x^\star_w) = \nabla F_w(x^\star_w) + ((1+q)w - w)c = 0 + qw \cdot c.$$

Additionally,

$$\nabla^2 F_{qw}(x^\star_w) = \nabla^2 B(x^\star_w)$$

Finally, we have: $0 = \nabla F_w(x^\star_w) = wc + \nabla B(x^\star_w)$, so it must be that;

$$c = -\nabla B(x^\star_w)/w$$

So for $z = x^\star_w$, we get

$$\begin{aligned}\lambda^2_{F_{qw}} &= \frac{(qw)^2 \nabla B(z)^T [\nabla^2 B(z)]^{-1} \nabla B(z)}{w^2} \\ &= q^2 \nabla B(z)^T [\nabla^2 B(z)]^{-1} \nabla B(z)\end{aligned}$$

So our goal becomes to lower bound $\nabla B(z)^T[\nabla^2 B(z)]^{-1}\nabla B(z) \leq \nu$. The smaller ν, the larger we can set q, and thus the more progress we make on each step. ν is called the "concordance parameter" of our barrier function, B. B is called a ν-self concordant barrier.

Showing that $\nabla B(z)^T[\nabla^2 B(z)]^{-1}\nabla B(z) \leq \nu$ is equivalent to showing that:

$$\nabla B(z)^T[\nabla^2 B(z)]^{-1}\nabla B(z) \leq \nu \cdot \nabla B(z)^T[\nabla B(z)\nabla B(z)^T]^{-1}\nabla B(z).$$

To do so, we will show how to bound:

$$x^T[\nabla^2 B(z)]^{-1}x \leq O(n) \cdot x[\nabla B(z)\nabla B(z)^T]^{-1}x \tag{1}$$

for all x, which means that $\nu \leq m$.

Before proving (1), let's consider what it would give. It would allow us to set $q = O(1/\sqrt{n})$. As discussed previously, this would lead to a total of $O(\sqrt{n}L\log(n/\epsilon))$ outer iterations for the interior point method.

Within each iteration, the dominant runtime cost is the minimization of F_w with Newton's method. We don't cover the details here, but it turns out that it is sufficient to solve this problem to a small positive constant, which takes just a constant number of steps. Each step of Newton's method requires building and inverting the Hessian matrix. Assuming for simplicity that $d = n$, this takes $O(n^3)$ time. This gives a total running time of roughly $O(n^{3.5}L)$. Compare this to ellipsoid, which took $O(n^2L)$ iterations total, where each iteration was cheaper at $O(n^2)$ time, so the total complexity is roughly $O(n^4L)$.

6 Orderings for PSD matrices

Before proving (1), we introduce some useful notation related to positive semidefinite (PSD) matrices. Recall that a symmetric matrix A is PSD if $x^TAx \geq 0$ for all x. The "Loewner ordering", which uses symbols $\succeq$ and $\preceq$ is defined as follows:

$$B \preceq A \qquad \text{if} \qquad A - B \text{ is positive semidefinite.}$$

The Loewner ordering is a partial ordering, meaning that, while it's not possible for both $B \preceq A$ and $B \succeq A$ to hold, it is possible that *neither* hold. Even so, the ordering has many useful properties. For example, it's possible to show that:

$$B \preceq A \qquad \iff \qquad B^{-1} \succeq A^{-1}.$$

Proving 1 is equivalent to proving that $[\nabla^2 B(z)]^{-1} \preceq O(n) \cdot [\nabla B(z)\nabla B(z)^T]^{-1}$. To do so, it suffices to show that:

$$O(n) \cdot \nabla^2 B(z)] \succeq \nabla B(z)\nabla B(z)^T.$$

To prove this bound, we have:

$$\begin{aligned}
x^T\nabla B(z)\nabla B(z)^Tx &\leq \sum_{i=1}^n\sum_{j=1}^n \left(\frac{1}{a_i^Tz - b_i}x^Ta_i\right)\left(\frac{1}{a_j^Tz - b_j}x^Ta_j\right) \\
&\leq n\sum_{i=1}^n \left(\frac{1}{a_i^Tz - b_i}x^Ta_i\right)^2 \\
&= n \cdot x^T\nabla^2 B(z)x.
\end{aligned}$$

The inequality follows from the bound $ab \leq \frac{a^2+b^2}{2}$ for all a, b.

7 Other considerations

We necessarily skipped over a lot of details in covering interior point methods. Again, please see [4] for more detailed coverage.

References

[1] Khachiyan, Leonid G. A polynomial algorithm in linear programming. Doklady Academii Nauk SSSR. Vol. 244. 1979.

[2] Karmarkar, Narendra. A new polynomial-time algorithm for linear programming. Proceedings of the sixteenth annual ACM symposium on Theory of computing. ACM, 1984.

[3] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. Journal of the ACM, 2004.

[4] Nisheeth K. Vishnoi. Algorithms for Convex Optimization. EPFL lecture notes, 2018. https://nisheethvishnoi.wordpress.com/convex-optimization/.