CS 224: Advanced Algorithms Spring 2017

Lecture 22 — April 7, 2017

Prof. Jelani Nelson *Scribe: Noah Golowich*

Today we will finish analysis of interior point methods using Newton's method, and start on "learning from experts."

1 Review of last time

Remember what we did on Tuesday: starting at some $x_0 \in \mathbb{R}^n$, by applying a number of updates like $x_{k+1} \leftarrow x_k - (\nabla^2 f(x_k))^{-1}\nabla f(x_k)$, we hoped to get $f(x_k) \to f(x^*)$ as long as f satisfies certain conditions (which in fact imply f is strongly convex). More precisely, we proved:

Theorem 1 (Newton's method)**.** *Given $\alpha \in [0,1]$, and some x_k, x_{k+1} in the update step above, write $x_\alpha = \alpha x_{k+1} + (1-\alpha)x_k$. If, for all α, and for all x_k, x_{k+1},*

$$(1-\epsilon)\nabla^2 f(x_k) \preceq \nabla^2 f(x_\alpha) \preceq (1+\epsilon)\nabla^2 f(x_k), \tag{1}$$

then $\delta(x_{k+1}) \le (\epsilon/(1-\epsilon))^k \delta(x_1)$, where $\delta(x) = \|\nabla f(x)\|_{(\nabla^2 f(x))^{-1}}$.

By strong convexity, we are at the minimum of f if and only if the gradient is 0, which is true iff its norm is 0. Thus this theorem generalizes the fact that $x_k \to x^*$, where x^* is a minimizer of f.

Now we define the idea of being "awesome" and "good" solutions that were discussed last class:

Definition 2. *We have* **fine centrality** *if $\delta_{\lambda_k}(x) \le 1/100$, and* **coarse centrality** *if $\delta_{\lambda_k}(x) \le 1/3$. And x is a* **perfectly central** *solution if $\delta_\lambda(x) = 0$.*

2 IPM Analysis

Recall that the overall algorithm is as follows:

1. Start with $\tilde{x}(\lambda_0)$.
2. $k \leftarrow 0$.
3. While λ_k isn't big enough, we let $\lambda_{k+1} \leftarrow (1+\alpha)\lambda_k$, do $O(1)$ Newton steps on $\tilde{x}(\lambda_k)$ to get $\tilde{x}(\lambda_{k+1})$, and then $k \leftarrow k+1$.

To analyze this algorithm, we must address several questions:

1. Remember that we start at $\lambda \approx 0$ and want to stop at large λ. How large?
2. Verify that (1) holds when we apply Newton's method.

3. We need to understand the rate at which we can increase the λs.

4. At the end of the day, we will end up with some finely central point for a large λ; when why does $\delta_\lambda(x) \leq 1/100$ for large λ imply that we're done? This is a problem set problem. In today's lecture, we show that if x is perfectly central, then it gives a solution to the LP; on the pset, we relax this to finely central solutions.

5. We need a finely central point $\tilde{x}(\lambda_0)$ to get started, for small λ_0.

Notation. $x(\lambda)$ is a minimizer for f_λ. $\tilde{x}(\lambda)$ is a finely central point for f_λ. For $n \in \mathbb{N}$, we let $J \in \mathbb{R}^n$ be the vector of all 1s, that is $J = [1, 1, \ldots, 1]^T$ (this was denoted using a slightly different symbol in lecture).

2.1 How large does λ need to be?

For $m, n \in \mathbb{N}, A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m, c \in \mathbb{R}^n$, remember the LP is $\min c^T x$, subject to $Ax \geq b$, and let an optimal point be x^*. Recall that $f_\lambda(x) = \lambda c^T x + p(s(x))$. Moreover, $p(s(x)) = \sum_{i=1}^m \ln(s(x_i))$, so that $\nabla f_\lambda(x) = \lambda c - A^T S_x^{-1} \cdot J$. Here J is the vector of 1s, and $S_x = diag(s_(x), \ldots, s_m(x))$, with $s(x) = Ax - b \geq 0 \in \mathbb{R}^m$. Then $\nabla^2 f_\lambda(x) = A^T S_x^{-2} A$.

We have that since f_λ is strictly convex, its gradient is zero at its unique minimizer (the "perfectly central" point at λ) $x(\lambda)$. Hence

$$0 = \langle 0, x(\lambda) - x^* \rangle = \langle \nabla f_\lambda(x(\lambda)), x(\lambda) - x^* \rangle,$$

so using the form of ∇f_λ above,

$$\lambda c^T (x(\lambda) - x^*) = \langle A^T S_{x(\lambda)}^{-1} J, x(\lambda) - x^* \rangle,$$

and the RHS of the above is $J^T S_{x(\lambda)}^{-1} A(x(\lambda) - x^*)$. But, $A(x(\lambda) - x^*) = s(x(\lambda)) - s(x^*)$, since $s(x) = Ax - b$, and the b's cancel out. Therefore, since S_x is a diagonal matrix,

$$\lambda c^T (x(\lambda) - x^*) = \sum_{i=1}^m \frac{s(x(\lambda)) - s(x^*)}{s(x(\lambda))} \leq m.$$

The last inequality follows since each term in the sum is at most 1, meaning that

$$c^T x(\lambda) \leq m/\lambda + OPT.$$

Therefore, if we set $\lambda > m/\epsilon$, meaning that $x(\lambda)$ gives cost at most $OPT + \epsilon$.

A few notes: If we want to solve the LP exactly, it is enough to take λ exponential in L (not shown here). Also, the number of steps to get accuracy ϵ is logarithmic in $1/\epsilon$.

2.2 Now we want to verify the Hessian Newton condition

To verify the Hessian condition necessary for convergence of Newton's method, we want to show that if we move from x to x' in a Newton iteration, then

$$(1 - \epsilon) A^T S_x^{-1} A \preceq A^T S_{x_\alpha}^{-2} A \preceq (1 + \epsilon) A^T S_x^{-1} A,$$

where we write $x_\alpha = x + \alpha(x' - x) = \alpha x' + (1-\alpha)x$. This Loewner ordering $A \preceq B$ means that for all $z \in \mathbb{R}^n$, $z^T A z \leq z^T B z$. We can drop all of the A's, by replacing z with Az. So, it suffices to show that

$$(1-\epsilon)S_x^{-2} \preceq S_{x_\alpha}^{-2} \preceq (1+\epsilon)S_x^{-2}.$$

It turns out to be true that $A \preceq B \Rightarrow A^{-1} \preceq B^{-1}$. Therefore, by this fact (we don't actually need it here, since S_x is diagonal), as well as taking the square root (which is certainly allowed since S_x is diagonal, we have that it suffices to show

$$\sqrt{1/(1+\epsilon)}S_x \preceq S_{x_\alpha} \preceq \sqrt{1/(1-\epsilon)}S_x.$$

By rescaling ϵ by at most a factor of 2, it suffices to show

$$(1-\epsilon)S_x \preceq S_{x_\alpha} \preceq (1+\epsilon)S_x,$$

or equivalently, by subtracting S_x an applying S_x^{-1},

$$-\epsilon I \preceq S_x^{-1}(S_{x_\alpha} - S_x) \preceq \epsilon I,$$

meaning all eigenvalues of the matrix in the middle are of magnitude at most ϵ; in other words, all of its diagonal entries have magnitude at most ϵ. Hence it suffices to show:

$$\|S_x^{-1}(s(x_\alpha) - s(x))\|_\infty \leq \epsilon.$$

Remember for $v \in \mathbb{R}^n$, $\|v\|_\infty$ and $\|v\|_p = (\sum |a_i|^p)^{1/p}$. Minkowski's inequality shows that $\|v\|_p$ is indeed a norm. Also, $p \geq q$ implies that $\|a\|_p \leq \|a\|_q$. We will bound the infinity norm by the 2 norm, so it suffices to bound:

$$\|S_x^{-1}(s(x_\alpha) - s(x))\|_2.$$

Now, since the b's cancel, remember that $s(x_\alpha) - s(x) = Ax_\alpha - Ax$, so we want to bound

$$\|S_x^{-1}A(x_\alpha - x)\|_2 = \alpha \cdot \|S_x^{-1}A(x' - x)\|,$$

and using definition of x' in terms of x (just the Newton update step),

$$\alpha \cdot \|S_x^{-1}A(\nabla^2 f_\lambda(x))^{-1}\nabla f_\lambda(x)\|_2. \tag{2}$$

Now, $\|v\|_2 = \|v\|_I$, with $\|v\|_A = \sqrt{x^T A x}$. Now, using shorthand for gradient and Hessian, the above norm squared is:

$$\nabla^T(\nabla^2)^{-T}A^T S_x^{-1}S_x^{-1}A(\nabla^2)^{-1}\nabla.$$

But $A^T S_x^{-2} A = \nabla^2$, and there is a $\nabla^{-2T} = \nabla^{-2}$ right next to it, so we get:

$$\nabla^T(\nabla^2)^{-1}\nabla,$$

so (2) is:

$$\alpha \cdot \|\nabla f_\lambda(x)\|_{(\nabla^2 f_\lambda(x))^{-1}} = \alpha \cdot \delta_\lambda(x) \leq \delta_\lambda(x) \leq 1/3$$

if x is coarsely central for λ.

Therefore, as long as we start step $k+1$ with a solution x that is coarsely central for λ_{k+1}, we will have that the Newton Hessian condition is verified with $\epsilon = \delta_{\lambda_k}(x)$ at step k, so as long as we keep this below 1/3 (which we will), we will be good.

2.3 Rate at which we can increase λ?

We want to figure out the largest increase of $\lambda_k \to \lambda_{k+1}$ to ensure that if $\tilde{x}(\lambda_k)$ is finely central, how big can we make λ_{k+1} while keeping $\tilde{x}(\lambda_k)$ coarsely central for $f_{\lambda_{k+1}}$. In particular, given $\delta_{\lambda_k}(\tilde{x}(\lambda_k)) \leq 1/100$, we want $\delta_{\lambda_{k+1}}(\tilde{x}(\lambda_k)) \leq 1/3$.

We will have $\lambda_{k+1} = (1+t)\lambda_k$; how big can we make t?

Look at

$$\delta_{\lambda_{k+1}}(\tilde{x}(\lambda_k)) = \|\nabla f_{\lambda_{k+1}}(\tilde{x}(\lambda_k))\|_{(\nabla^2 f_{\lambda_{k+1}}(\tilde{x}(\lambda_k)))^{-1}}.$$

Fortunately, since we are in an LP setting, the λ_{k+1} norm is the same as the λ_k norm: just look at the Hessian: it doesn't depend on λ! So, this is

$$\|(1+t)\lambda_k c - A^T S_{\tilde{x}(\lambda_k)}^{-1} J\|_M,$$

with $M = (\nabla^2 f_{\lambda_{k+1}}(\tilde{x}(\lambda_k)))^{-1}$. We want to make the stuff inside the norm look like the λ_kth centrality, plus extra additional stuff, then apply triangle inequality; in particular, add and subtract $tA^T S_{\tilde{x}(\lambda_k)}^{-1} J$, get

$$\|(1+t)(\lambda_k c - A^T S_{\tilde{x}(\lambda_k)}^{-1} J) + tA^T S_{\tilde{x}(\lambda_k)}^{-1} J\|_M$$

which by triangle inequality (which follows since this "matrix norm" is just a normal l_2 norm with a different basis), is at most:

$$(1+t)\|\lambda_k c - A^T S_{\tilde{x}}^{-1} J\|_M + t\|A^T S_{\tilde{x}}^{-1} J\|_M \leq (1+t)\delta_{\lambda_k}(\tilde{x}(\lambda_k)) + t\|A^T S_{\tilde{x}}^{-1} J\|_M.$$

This is at most

$$\frac{1+t}{100} + t\|A^T S_{\tilde{x}}^{-1} J\|_{(A^T S_{\tilde{x}}^{-2} A)^{-1}}. \tag{3}$$

The squared norm in the equation above is somewhat nasty; it is:

$$J^T S_{\tilde{x}}^{-1} A (A^T S_{\tilde{x}}^{-2} A)^{-1} A^T S_{\tilde{x}}^{-1} J \tag{4}$$

Now, in general, for $X \in \mathbb{R}^{m \times n}, v \in \mathbb{R}^m$, $X(X^T X)^{-1} X^T v$ is the orthogonal projection of v on the column space of X, assuming that X has full column rank.[1] (This assumption implies that A has more constraints than variables.) Here, we have $X = S_{\tilde{x}(\lambda_k)}^{-1} A$.

This norm (4) is the inner product of the all-one's vector after projecting onto orthogonal subspace and the all-one's vector J itself. But, for a general vector $v \in \mathbb{R}^m, S \subseteq \mathbb{R}^m$ a subspace, if the orthogonal projection of v onto S is u, then we have $\langle v, u\rangle = \langle u, u + (v-u)\rangle = \langle u, u\rangle$, since $\langle u, v-u\rangle = 0$; but, projecting onto orthogonal subspace just decreases norms. So, the quantity (3) is at most

$$\frac{1+t}{100} + t\|J\|_2 = \frac{1+t}{100} + t\sqrt{m}.$$

So, if we increase λ_{k+1} by a $(1+t)$ factor compared to λ_k, then the centrality for λ_{k+1} is $(1+t)/100 + t\sqrt{m} \leq 1/3$ if $\boxed{t = 1/(4\sqrt{m})}$.

[1] Assuming X is square, expand out the SVD of X, get that this thing is UU^T, where $X = U\Sigma V^T$. Here U actually refers to the $m \times n$ ($n \leq m$) matrix with orthogonal columns that "matters" in the SVD. It is now clear that $UU^T v$ gives the orthogonal projection of v onto the column space of U (which is the same as that of X, by what SVD is).

2.4 Finely central point for λ_0?

To get $\tilde{x}(\lambda_0)$, we need to make sure it has positive volume, so that there exists an interior point. In particular, we modify the LP to be:

$$\min c^T x + Nz,$$

for $N = 10^L$, and z is a new variable, subject to

$$Ax + zJ \geq b, \quad 0 \leq z \leq 2^{L+1}, \quad \forall i, -2^L \leq x_i \leq 2^L.$$

Now, the point $x = 0, z = \|b\|_\infty$ is an interior point (technically one can show that this is in the interior of the above polytope, but all we need is that it is finely central for λ_0). We also claim that an optimal solution to this modified LP gives us an optimal solution to the original one: in particular, in an optimal solution to this modified LP, you never place any mass on z whatsoever: assuming there's a feasible solution to the original LP, the individual x_i must be at most $O(2^L)$, meaning the objective has value at most $O(2^L)$, so in order to put mass on z and beat this objective we must have $z = O(1/2^L)$, which in turn implies that the new LP has optimum at $z = 0$ (this is not a complete proof; see the references for details).

To construct a finely central point, we define $\tilde{x} = (0, \|b\|_\infty)$, where $z = \|b\|_\infty$. $\tilde{x}$ is not necessarily finely central. Now, note that $\tilde{x}$ is perfectly central for $\lambda = 1$ with cost function $c' = A^T S_{\tilde{x}}^{-1} J$. Instead of slowly increasing λ, we actually decrease λ by a factor of $1 - t$, and repeating all of the analysis above gives a finely central point for this cost function c', for a very tiny λ. Once λ is super-tiny, a finely central point for this cost function is a finely central point for any other λ. Then, we just change the cost function to c, and we will still be finely central. Then, we can run everything forwards.

More precisely:

- $\tilde{x}$ is perfectly central for $\lambda = 1, c' = A^T S_{\tilde{x}}^{-1} J$.
- Let $\tilde{x}(1) \leftarrow (0, \|b\|_\infty)$, $\lambda \leftarrow 1$.
- For $\lambda > \lambda_0 = 2^{-\Theta(L)}$, let $\lambda \leftarrow (1 - 1/\sqrt{m})\lambda$, and run $O(1)$ Newton steps on $\tilde{x}$ to get a new $\tilde{x}(\lambda)$. This will be finely central for the new λ, and also coarsely central for the next λ (namely, $(1 - t)\lambda$) by the exact same analysis as was used above.
- Then output $\tilde{x}$ as $\tilde{x}(\lambda_0)$, use the forward algorithm for IPM.

2.5 Overview

How many iterations do we need in the outer loop? We start at λ_0, and end at $\lambda_t = m/\epsilon = (1 + 1/\sqrt{m})^T \lambda_0$. So, we want

$$(1 + 1/\sqrt{m})^T \lambda_0 > m/\epsilon,$$

with $\lambda_0 = 2^{-\Theta(L)}$, so taking logs gives us that

$$T \ln(1 + 1/\sqrt{m}) > L + \ln(m/\epsilon),$$

so

$$T \geq \sqrt{m}(L + \ln(m/\epsilon)).$$

Also, a bit of history:

- The first IPM for LP was given in [Kar84]; it was an iterative approach with an outer loop and inner loop. It needed $T = \Omega(mL)$.
- More recently, [Ren88], showed how to get $T = \tilde{O}(\sqrt{m}L)$.
- State-of-the-art: [LS14], get $T = \tilde{O}(\sqrt{rank(A)}L)$.

Throught this analysis, we assumed that the Hessian is invertible, and to get this, we need $n \leq m$ (this is necessary, not sufficient). (In fact, these full-rank conditions don't matter too much, since we can use the Moore-Penrose pseudoinverse instead.) But, there is a simple trick to make sure that the Hessian is invertible (this is a pset problem).

The fastest methods for lots of common problems nowadays (e.g. instead of Edmonds-Karp, Ford-Fulkerson), go through LP and use interior point methods.

3 Preview: Learning from Experts

Suppose some event will happen today (either it will rain or not rain [2]), or some stock will go up or down).

Based on what experts say, you want to figure out what to do. Say there are T days; each day, it will rain or not; there are n experts. Over time, you start to learn who the right people are. On day 1, you have no clue who is right. So, what to do?

- For $t = 1, \ldots, T$:
- Predict the majority vote amongst the experts who are alive.
- At the end of the day, "kill" (not actually; just ignore) all experts who were wrong that day.

Lemma 3. *If there exists a perfect expert, then we make at most* $\lceil \log_2 n \rceil$ *mistakes.*

Proof. Each time we make a mistake, the number of alive experts gets cut by a factor of at least 2; we can cut people like this at most $\log_2 n$ times. □

What if the best expert makes at most E errors? We may kill him/her accidentally. So, if all experts are dead, we just revive them all.

Lemma 4. *We make at most* $(E + 1)\lceil \log_2 n \rceil$ *errors.*

Proof. Break up time into phases; each phase is one mistake of the best expert: inside of each phase, we have at most $\log_2 n$ mistakes, by the same analysis. This gives the bound immediately. □

[2] Today, it is *definitely* raining.

Next time: for all $\eta \in (0, 1)$, we can get at most $(2 + \eta)E + \frac{2 \log n}{\eta}$ mistakes. This gives us a 2-competitive ratio with an *additive* $\log n$. The idea is that when someone is wrong, decrease your "confidence" in them, and take a weighted majority vote based on confidences. This is related to regret minimization.

References

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, December 1984.

[LS14] Yin Tat Lee and Aaron Sinford. Path Finding Methods for Linear Programming: Solving Linear Programs in $\sqrt{rank}$ Iterations and Faster Algorithms for Maximum Flow, 2014.

[Ren88] James Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, January 1988.