TCS Guide of Convex Optimization - Day 3 Homotopy Method

February 1, 2023

1 Why Homotopy

Good of cutting plane method:

- It works under oracle settings, it doesn't assume anything beyond the number of variables.

Bad of cutting plane method:

- It works under oracle settings, so the optimizer always only have partial information hence should never be "optimal".

Question: How can we design an optimization that always uses all the information about the problem?

Homotopy Method: To solve $\min_x f(x)$, we come up with a family of function $f_t(x)$ such that f_1 is easy to optimize while $f_0 = f$. It suffices to track the path $x_t \stackrel{\text{def}}{=} \min_x f_t(x)$. Often, x_t is differentiable and satisfies some differential equation. Hence, we can optimize the function by solving some ODE.

Lemma 1. *We have* $\frac{dx_t}{dt} = -(\nabla^2 f_t(x_t))^{-1} \nabla \frac{df_t}{dt}(x_t)$.

Proof. Note that

$$\nabla f_t(x_t) = 0.$$

Taking derivative on d/dt on both side, we have

$$\nabla \frac{df_t}{dt}(x_t) + \nabla^2 f_t(x_t) \frac{dx_t}{dt} = 0.$$

Solving gives the result. □

Joke example: Suppose we want to minimize $\min_x |x|$. We can define $f_t(x) = \sqrt{x^2 + t^2}$. $f_1(x)$ is easier to minimize because the function is more differentiable.

In the rest of the section, we will focus applying homotopy method for solving linear program. The most popular version is called interior point method (IPM).

2 Basic Property of Linear Programs

- Primal LP $\min_{Ax=b, x \geq 0} c^\top x$ and dual LP $\max_{A^\top y + s = c, s \geq 0} b^\top y$ where $x, s \in \mathbb{R}^n_{\geq 0}$.
- Under mild assumptions, OPT $= \min_{Ax=b, x \geq 0} c^\top x = \max_{A^\top y + s = c, s \geq 0} b^\top y$.

Lemma 2. *The duality gap* $c^\top x - b^\top y = x^\top s$. *In particular, for optimal* x, s, *we have* $x_i s_i = 0$ *for all* i.

Proof. Imagine a ball lies on $Ax = b$ with the force given by c. When the ball rest, it will get the force from the hyperplane $Ax = b$ and gets the force from the boundary $x \geq 0$. It is exactly given by $A^\top y + s = c$. If the ball is not touching $x \geq 0$, then no force and hence $s_i = 0$. Otherwise, $x_i = 0$. Therefore, $x_i s_i = 0$. □

The following lemma shows that we can represent all feasible vectors by just a positive vector μ. We can view it as an alternative representation.

Lemma 3. *For any positive vector* $\mu \in \mathbb{R}^n_{>0}$, *there is an unique feasible* x *and* s *such that*

$$x_i s_i = \mu_i.$$

Remark. For the rest of the talk, we write $T_x(\mu)$ be the unique x associated with μ. Similarly for $T_s(\mu)$.

Proof. Consider

$$x = \arg \min_{Ax=b, x \geq 0} c^\top x - \sum \mu_i \ln x_i.$$

Note that x is feasible in primal. Also, the optimal shows that

$$c - \frac{\mu}{x} = A^\top y$$

for some y. Let $s = \frac{\mu}{x}$. Then, we have $s \geq 0$

$$A^\top y + s = c.$$

So, s is feasible in dual. □

This representation is very interesting because $T_x(0), T_s(0)$ is exactly the optimal solution.

3 Interior Point Method and its Basic Properties

Remark: There are many versions of IPM.

3.1 One of the Framework

- **Idea:** Modify the linear program such that $T_x(1)$ and $T_s(1)$ is explicit given. Then follows the path from all 1 vector to all 0 vector.
- Note that the whole path are interior in the domain, hence the name interior point method.

Exercise 4. Show how to modify the linear program such that $T_x(1)$ and $T_s(1)$ are explicitly given in a way that the optimal solution doesn't change. **Hint:** First, solve an easier problem of modify the linear program to get an explicit interior point first. For example, you may consider getting new variable like changing $\{Ax = b, x \geq 0\}$ to $\{Ax + z^+ - z^- = b, x \geq 0, z^+ \geq 0, z^- \geq 0\}$.

IPM:

- Invariant: $\frac{1}{2} \cdot t \leq xs \leq \frac{3}{2} \cdot t$. Maintain it via the potential $\Phi(\frac{xs}{t} - 1) \stackrel{\text{def}}{=} \sum_i \phi(\frac{xs}{t} - 1) \leq C_\phi$.
- Parameter step size $h \approx \frac{1}{100}$.
- Initialize $x = T_x(1)$ and $s = T_s(1)$.
- While $t \geq \frac{\epsilon}{2n}$,
  - If $\Phi(\frac{xs}{t} - 1) \geq \frac{C_\phi}{2}$
    - Let $v = -ht \cdot \frac{\nabla \Phi(\frac{xs}{t} - 1)}{\|\nabla \Phi(\frac{xs}{t} - 1)\|_2}$.
    - Pick $\delta_x, \delta_s, \delta_y$ such that

$$\begin{aligned} S\delta_x + X\delta_s &= v, \\ A\delta_x &= 0, \\ A^\top \delta_y + \delta_s &= 0. \end{aligned}$$

    - Move $x \leftarrow x + \delta_x$, $s \leftarrow s + \delta_s$
  - $t \leftarrow (1 - \frac{h}{2\sqrt{n}})t$.

TODO:

1. Pick ϕ and C_ϕ
   (a) Assume $\Phi(\frac{xs}{t} - 1) \leq C_\phi$ implies $\frac{1}{2}t \leq xs \leq \frac{3}{2}t$.
   (b) Example: $\phi(u) = u^2$ and $C_\phi = \frac{1}{4}$.
2. Show that $\Phi \leq C_\phi$ all the time.
3. Bound the cost per iteration.

3.2 Basic Properties

- Total number of iteration is $\sqrt{n}\log\frac{n}{\epsilon}$.
- The duality gap at the end is at most ϵ.

Lemma 5. *(What is the step?)* $X^{-1}\delta_x = (I-P)\frac{v}{xs}$. $S^{-1}\delta_s = P\frac{v}{xs}$ *where*

$$P = S^{-1}A^\top(AS^{-1}XA^\top)^{-1}AX.$$

Proof. Direct calculations. □

Remark. You can think P is an almost orthogonal projection to the subspace $\mathrm{im}(S^{-1}A^\top)$ while $I-P$ is an almost orthogonal projection to the null space $\mathrm{null}(AX)$

Lemma 6. *(Is the step feasible?)* $\|\delta_x/x\|_2 \le 4h$, $\|\delta_s/s\|_2 \le 4h$. *(Alternatively,* $\|\ln x^{(\mathrm{new})} - \ln x\|_2 \le O(h)$ *and* $\|\ln s^{(\mathrm{new})} - \ln s\|_2 \le O(h)$*.) In particular,* x, s *are always feasible.*

Proof. Let $P_o = X^{1/2}S^{-1/2}A^\top(AS^{-1}XA^\top)^{-1}AX^{1/2}S^{-1/2}$. Note that P_o is orthogonal projection and hence

$$\begin{aligned} X^{-1}\delta_x &= (I-P)\frac{v}{xs} \\ &= (I - X^{-1/2}S^{-1/2}P_oX^{1/2}S^{1/2})\frac{v}{xs} \\ &= X^{-1/2}S^{-1/2}(I-P_o)X^{1/2}S^{1/2}\frac{v}{xs}. \end{aligned}$$

So, we have

$$\begin{aligned} \|X^{-1}\delta_x\|_2 &\le \sqrt{\frac{2}{t}}\|(I-P_o)X^{1/2}S^{1/2}\frac{v}{xs}\|_2 \\ &\le \sqrt{\frac{2}{t}}\|X^{1/2}S^{1/2}\frac{v}{xs}\|_2 \\ &\le \sqrt{\frac{2}{t}}\sqrt{2t}\|\frac{v}{xs}\|_2 \\ &\le 4h. \end{aligned}$$

For the log term, we note that

$$\ln x^{(\mathrm{new})} - \ln x \approx \frac{x^{(\mathrm{new})} - x}{x} = \frac{\delta_x}{x}.$$

The proof for s are similar. □

Lemma 7. *(Does the step decrease the potential?)Assume* $\phi''(u) \le O(|\phi'(u)|+1)$ *for all* u*, after the* x, s *update, we have* $\Phi^{(\mathrm{new})} - \Phi \le \langle \nabla\Phi, \frac{v}{t}\rangle + O(\|\nabla\Phi\|_2 + 1)h^2$*. In particular, ignoring the* h^2 *term,* Φ *is decreasing by* $-h\|\nabla\Phi\|_2$*.*

Proof. Let $u = \frac{xs}{t} - 1$. Note that

$$u^{(\mathrm{new})} - u = \frac{x\delta_s + s\delta_x + \delta_x\delta_s}{t} = \frac{v + \delta_x\delta_s}{t}.$$

By Taylor expansion, we have roughly

$$\Phi^{(\mathrm{new})} = \Phi + \left\langle \nabla\Phi, u^{(\mathrm{new})} - u \right\rangle + \sum \phi''(u_i)(u_i^{(\mathrm{new})} - u_i)^2.$$

Since $\delta_x\delta_s/t \approx \delta_x/x \cdot \delta_s/s$, we have $\|\delta_x\delta_s/t\|_2 \le 2\|\delta_x/x\|_2 \cdot \|\delta_s/s\|_2 = O(h^2)$. Hence, we have

$$\left\langle \nabla\Phi, u^{(\mathrm{new})} - u \right\rangle = \left\langle \nabla\Phi, \frac{v}{t} \right\rangle + O(h^2)\|\nabla\Phi\|_2.$$

For the last term, we note that $\|u^{(\mathrm{new})} - u\|_2 \le \|\frac{v}{t}\|_2 + \|\frac{\delta_x\delta_s}{t}\|_2 = O(h)$. Hence, we have

$$\begin{aligned} \sum \phi''(u_i)(u_i^{(\mathrm{new})} - u_i)^2 &\le O(1)\cdot\sum(|\phi'(u_i)|+1)(u_i^{(\mathrm{new})} - u_i)^2 \\ &= O(\|\nabla\Phi\|_2 + 1)h^2. \end{aligned}$$

□

Note that the above lemma basically showed Φ is bounded through the algorithm. When we change t multiplicative, the term $\Phi(\frac{xs}{t}-1)$ changes by roughly $\left\langle \nabla\Phi, \frac{h}{2\sqrt{n}} \right\rangle \leq \frac{h}{2}\|\nabla\Phi\|_2$. So, when our x, s step decrease by $-h\|\nabla\Phi\|_2$, it cancels out the effect of changing t.

3.3 ℓ_2 neighborhood version

In this case, we select $\phi(u) = u^2$ and $C_\phi = \frac{1}{4}$.

- Clearly, we have $\phi''(u) \leq O(|\phi'(u)| + 1)$.
- When $\Phi = \Omega(1)$, we have $\|\nabla\Phi\|_2 = \sqrt{\Phi} = \Omega(1)$. So,

$$\begin{aligned}\Phi^{(\mathrm{new})} - \Phi &\leq \left\langle \nabla\Phi, \frac{v}{t} \right\rangle + O(\|\nabla\Phi\|_2 + 1)h^2 \\ &\leq -h\|\nabla\Phi\|_2 + O(\|\nabla\Phi\|_2)h^2.\end{aligned}$$

  So, for small enough constant h, we showed the potential is decreasing.

- Naively, the cost per iteration is n^ω. So, total runtime is $n^{\omega+1/2}\log(1/\epsilon)$.

4 Robust Interior Point Method

The bottleneck of the algorithm is to solve the equation

$$\begin{pmatrix} S & X & 0 \\ A & 0 & 0 \\ 0 & I & A^\top \end{pmatrix} \begin{pmatrix} \delta_x \\ \delta_s \\ \delta_y \end{pmatrix} = \begin{pmatrix} v \\ 0 \\ 0 \end{pmatrix}.$$

Note that both S and X in the matrix changes slowly, but not exactly sparsely. If we can change a coordinate only when necessary, then we can update the matrix inverse instead of computing it again.

Here is a version of IPM that update the matrix only when necessary:

IPM method:

- Invariant: $\frac{1}{2}\cdot t \leq xs \leq \frac{3}{2}\cdot t$. Maintain it via the potential $\Phi(\frac{xs}{t}-1) \stackrel{\mathrm{def}}{=} \sum_i \phi(\frac{xs}{t}-1) \leq C_\phi$.
- Parameter step size $h \approx \frac{0.001}{\log n}$, approximation ratio to vectors $\delta \approx \frac{0.001}{\log n}$
- Initialize $x = T_x(1)$ and $s = T_s(1)$.
- While $t \geq \frac{\epsilon}{2n}$,
  - Pick $\overline{x} \in (1\pm\delta)x$, $\overline{s} \in (1\pm\delta)s$ and $\overline{t} \in (1\pm\delta)t$
  - If $\Phi(\frac{\overline{x}\overline{s}}{\overline{t}}-1) \geq \frac{C_\phi}{2}$
    - Let $v = -ht \cdot \frac{\nabla\Phi(\frac{\overline{x}\overline{s}}{\overline{t}}-1)}{\|\nabla\Phi(\frac{\overline{x}\overline{s}}{\overline{t}}-1)\|_2}$.
    - Pick $\delta_x, \delta_s, \delta_y$ such that

$$\begin{aligned}\overline{S}\delta_x + \overline{X}\delta_s &= v, \\ A\delta_x &= 0, \\ A^\top\delta_y + \delta_s &= 0.\end{aligned}$$

    - Move $x \leftarrow x + \delta_x$, $s \leftarrow s + \delta_s$
  - $t \leftarrow (1 - \frac{h}{2\sqrt{n}})t$.

Remark:

- We cannot use the ℓ_2 potential anymore (i.e. we cannot force xs is ℓ_2 close to t). If we only have ℓ_∞ approximation to x and s, we should only hope to force xs is ℓ_∞ close to t.

- Instead, we do
$$\phi(u) = \exp(\lambda u) + \exp(-\lambda u)$$
for some $\lambda = 10 \log n$ and $C_\phi = 100n$. Note that we still have $\Phi \leq C_\phi$ implies $xs \approx t$ (this is the reason for setting $\lambda = \log n$)
- We still have $\phi''(u) \leq O(|\phi'(u)| + 1)$. (up to log).
- When $100n \geq \Phi \geq 50n$, we have $\|\nabla\Phi\|_2 = \Theta(\sqrt{n})$ and hence
$$\begin{aligned}\Phi^{(\mathrm{new})} - \Phi &\leq \left\langle \nabla\Phi, \frac{v}{t} \right\rangle + \lambda^2 O(\|\nabla\Phi\|_2 + 1)h^2 \\ &\leq \left\langle \nabla\Phi, \frac{v}{t} \right\rangle + \lambda^2 O(\|\nabla\Phi\|_2)h^2.\end{aligned}$$
- The key difference is that v is computed using the approximate vectors. Note that
$$\frac{v}{t} \approx -\frac{h}{\sqrt{n}} \nabla\Phi(\frac{\overline{x}\overline{s}}{\overline{t}} - 1) \approx -h \cdot \frac{(1 \pm \lambda\delta)\nabla\Phi(\frac{xs}{t} - 1) \pm \lambda\delta}{\sqrt{n}}.$$
So, we have
$$\Phi^{(\mathrm{new})} - \Phi \leq (-h \pm \lambda\delta h \pm \lambda^2 h^2)\|\nabla\Phi\|_2$$
If $\delta = \frac{0.001}{\log n}$ and $h = \frac{1}{\log n}$, then we have Φ deceases by almost $h\|\nabla\Phi\|_2$. This finishes the proof for the potential decreases.

5 Discussion

What does this RIPM needs? It needs a data structure to maintain x and s as follows:

- Each step, x and s is moved by some linear algebra formula via $\overline{x}, \overline{s}$.
- We only need to know which coordinates of x, s moved a lot, so that we can update $\overline{x}, \overline{s}$.
- We also need to output x, s at the end of the algorithm.

6 Getting n^3 time without fast matrix

In some sense, RIPM reduces solving linear program to a streaming problem of "heavy hitter"-ish. Note that the linear system problem is changing on both the matrix and vector. We can further simplify it by noting:

$$\begin{pmatrix} M & v \\ 0 & -1 \end{pmatrix}^{-1} = \begin{pmatrix} M^{-1} & M^{-1}v \\ 0 & -1 \end{pmatrix}.$$

So, we can view the whole problem simply involves maintaining

$$M^{-1}e_j$$

for some sparsely updating M and a 1-sparse fixed vector e_j. Recall that our matrix M involves terms like

$$x, s, v$$

where v is just a coordinate-wise function of x, s.

Lemma 8. *(Naive maintenance) The cost of maintaining $M^{-1}e_j$ for a sequence of M is bounded by $O(n^2 T)$ where T is the total number of coordinate changes in M.*

Proof. Recall that
$$(M + e_i e_j^\top)^{-1} = M^{-1} + M^{-1}e_i(1 + e_j^\top M^{-1} e_i)^{-1} e_j^\top M^{-1}.$$
Given M^{-1} explicitly, we can compute RHS in n^2 time. Hence, each coordinate update to M takes n^2 time. □

Now, we bound the number of coordinate changes for the following greedy update algorithm:

- For each step, if $\overline{x}_i \notin (1 \pm \delta)x_i$, set $\overline{x}_i = x_i$. (Similar for other variables).

Lemma 9. *(Number of coordinate changes) The greedy update algorithm makes $O(n \log n \log(1/\epsilon))$ changes in total.*

Proof. Let T be the number of changes. Note that every time $\overline{x}$ moves, it moves by at least δ multiplicatively. Hence,

$$\sum_k \|\overline{x}_i^{(k+1)} - \overline{x}_i^{(k)}\|_1 = \Theta(T\delta).$$

Note that the total movement in ℓ_1 of $\overline{x}$ is always lower then $\overline{x}$. Hence, we have

$$\begin{aligned} T\delta &= O(\sum_k \|x_i^{(k+1)} - x_i^{(k)}\|_1) \\ &= O(\sqrt{n} \sum_k \|x_i^{(k+1)} - x_i^{(k)}\|_2) \\ &= n \log(1/\epsilon). \end{aligned}$$

Hence, we have the bound for T. □

Combining the previous two lemmas, we have a LP algorithm with runtime $\widetilde{O}(n^3 \log(1/\epsilon))$. Note that this does not use fast matrix multiplication. We will discuss next how to use fast matrix multiplication to improve the runtime.

7 Getting $n^{2.38}$ time by batch updates

- Naive approach: Greedy. Problem, all iteration can have $\sqrt{n}$ changes
- Batch update: most of the iteration has a small number of changes
- Explain how to do inverse maintence