Linear Programming in Low Dimensions

丛宇

UESTC

2024-09-24

1.1 What is LP?

Definition: A linear program (LP) is an optimization problem of the form:

$$
\begin{aligned}
& \max c^T x \\
s.t. \quad & Ax \geq b \\
& x \geq 0
\end{aligned}
$$

LPs are easy to solve

1.1 What is LP?

Definition: A linear program (LP) is an optimization problem of the form:

$$
\begin{aligned}
& \max c^T x \\
s.t. \quad & Ax \geq b \\
& x \geq 0
\end{aligned}
$$

LPs are easy to solve in low dimensions.

1.2 LP in Machine Learning

Machine learning is full of convex optimization problems. Why focus on the special case of LP?

- Solving LPs are fast (compared to convex optimization problems) e.g. linear programming SVM
- Sparse Linear Models
- Online learning via LPs
- …

2.1 LP in d Dimension

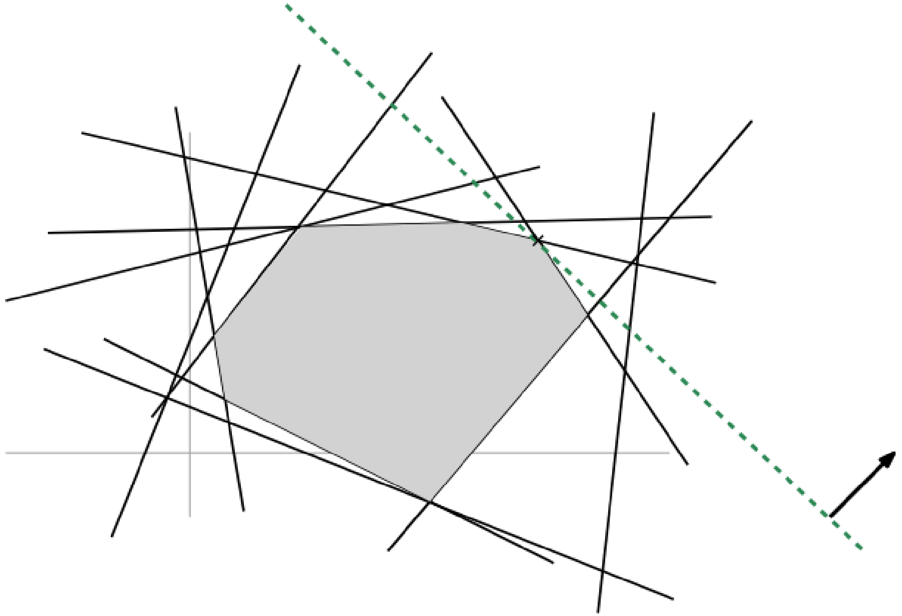
For fixed dimensions, LPs can be solved in linear time!

simplex method	det.	$O(n/d)^{d/2+O(1)}$
Megiddo [24]	det.	$2^{O(2^d)}n$
Clarkson [9]/Dyer [14]	det.	$3^{d^2}n$
Dyer and Frieze [15]	rand.	$O(d)^{3d}(\log d)^d n$
Clarkson [10]	rand.	$d^2 n + O(d)^{d/2+O(1)} \log n + d^4\sqrt{n}\log n$
Seidel [26]	rand.	$d!n$
Kalai [19]/Matoušek, Sharir, and Welzl [23]	rand.	$\min\{d^2 2^d n,\ e^{2\sqrt{d\ln(n/\sqrt{d})}+O(\sqrt{d}+\log n)}\}$
combination of [10] and [19, 23]	rand.	$d^2 n + 2^{O(\sqrt{d\log d})}$
Hansen and Zwick [18]	rand.	$2^{O(\sqrt{d\log((n-d)/d)})}n$
Agarwal, Sharir, and Toledo [4]	det.	$O(d)^{10d}(\log d)^{2d}n$
Chazelle and Matoušek [8]	det.	$O(d)^{7d}(\log d)^d n$
Brönnimann, Chazelle, and Matoušek [5]	det.	$O(d)^{5d}(\log d)^d n$
~~this paper~~ Chan	det.	$O(d)^{d/2}(\log d)^{3d}n$

table stolen from https://dl.acm.org/doi/10.1145/3155312

2.2 Seidel's Algorithm - $O(d!n)$

The well-loved randomized algorithm is extremely simple.



LPs are just finding an extreme point in some direction in a polytope.
https://www.cs.cmu.edu/~15451-f15/lectures/lect1021-lpII.pdf

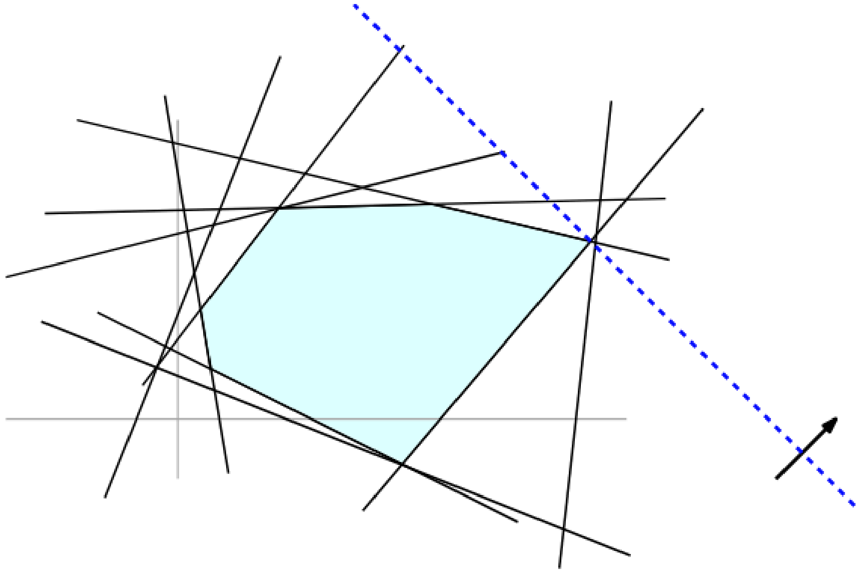
2.2 Seidel's Algorithm - $O(d!n)$

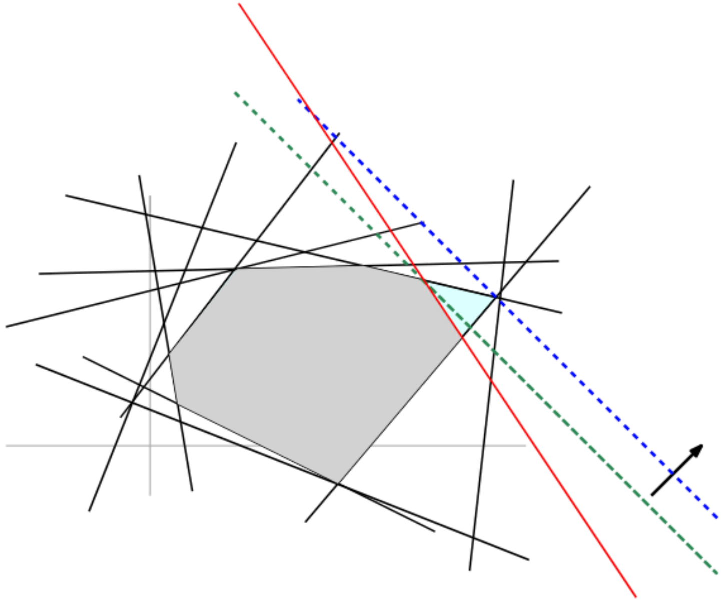
Let's add in the constraints one at a time and keep track of the current optimal solution x^*.

After adding a new constraint $a_m \cdot x \leq b_m$, there will be two cases,

1. x^* satisfies the new constraint,
2. x^* does not satisfy the new constraint.

2.2 Seidel's Algorithm - $O(d!n)$





The new optimal x^* should locate on a $d-1$ dimension hyperplane $a_m \cdot x = b_m$.

Assume that our d dimensional LP is feasible and the optimal x^* is qnique. Then x^* is defined by exactly d constraints(hyperplanes).

2.2 Seidel's Algorithm - $O(d!n)$

```
algorithm seidel(S, f, X) is
    R := empty set
    B := X
    for x in a random permutation of S:
        if f(B) ≠ f(B ∪ {x}):               // case 2
            B := seidel(R, f, X ∪ {x})
        R := R ∪ {x}
    return B
```

Case 2 is more expensives than 1. What is the chance of getting case 2 when inserting the m'th constraint?

If we are inserting constraints in a random order,

$$P(\text{case } 2) \leq \frac{d - |X|}{m} \leq \frac{d}{m}$$

2.2 Seidel's Algorithm - $O(d!n)$

```
algorithm seidel(S, f, X) is
    R := empty set
    B := X
    for x in a random permutation of S:
        if f(B) ≠ f(B ∪ {x}):                 // case 2
            B := seidel(R, f, X ∪ {x})
        R := R ∪ {x}
    return B
```

We need $O(d)$ time to do the violation test for case 2. Let the expected number of violation tests be $T(s, x)$,

$$T(s, x) \leq \sum_{i=d}^{s} \frac{d-x}{i}(1 + T(i, x+1))$$

After solving the recurence, we get $T(s, x) = O(d!s)$. Thus Seidel's alg has expected complexity $O(d!n)$ on any d-dimension LP with n constraints.