Optimization with verification oracles

Sergei Chubanov

April 24, 2018

Outline

- Oracle model of computation
- Binary case with arbitrary functions
- Separable convex optimization
- Linear programming over finite sets

Oracle model

- Augmentation oracle
- Verification oracle

General scheme

- Optimization problem:

$$\min\{f(x) : x \in S\},$$

  where $f \in \mathcal{C}$.

- Verification or augmentation oracle for $\mathcal{C}$.
- Find $h^t \in \mathcal{C}$ and $x^t \in S$ such that x^t is optimal for h^t such that

$$f = \lim_{t \longrightarrow \infty} h^t.$$

- Use the oracle to verify optimality.

Binary optimization

- $S \subseteq \{0,1\}^n$.
- $f \in \mathcal{C}$, $f + g \in \mathcal{C}$, $\forall$ linear functions g.
- $\min\{f(x) : x \in S\}$.
- Augmentation oracle[1].

[1] The linear case: Schulz, A.S., Weismantel R., and Ziegler G.M. 0/1-integer programming: Optimization and augmentation are equivalent. *Lecture Notes in Computer Science* **979** 473-483 (1995)

Binary optimization: Greedy algorithm

Assume the following optimality condition:

$$x^* \in \arg\min_S f \iff x^* \in \arg\min_{U(x^*)} f$$

Greedy algorithm:

- Find x^{k+1} in $\arg\min_{U(x^k)} f$.
- $k := k + 1$.
- Repeat until $x^k \in \arg\min_{U(x^k)} f$.

Theorem

The greedy algorithm runs in polynomial time if f is integer-valued and polynomially computable.

Proof. Follows from a scaling algorithm.

Binary optimization: Scaling algorithm $\Longrightarrow$ Greedy alg.

- Augmentation for ("pay to change a bit"-function)

  $g(x) = f(x) + \delta \cdot ((-\mathbf{1})^{x^k})^T (x - x^k)^T$ at x^k :

  $$g(x^{k+1}) < g(x^k).$$

  $$f(x^{k+1}) \leq g(x^{k+1}) - \delta < g(x^k) - \delta = f(x^k) - \delta.$$

- Let $m \geq \|x\|_1, \forall x \in S$. If augmentation is not possible, then x^k is $2m\delta$-approximate:

  $$f(x^k) = g(x^k) \leq g(x^*) \leq f(x^*) + \|x^* - x^k\|_1 \cdot \delta \leq OPT + 2m\delta,$$

  where x^* is optimal. Then, $\delta := \delta/2$.

- Repeat until $\delta < \varepsilon/(2m)$.

Theorem

An ε-approximate solution in oracle time $O\left(m \log \frac{m(f(x^0)-LB)}{\varepsilon}\right)$.

Separable convex optimization

$$\min\{f(x) : x \in S\},$$

$$f(x) = \sum_{j=1}^{n} f_j(x_j), \quad S = \{x : Ax = b, \mathbf{0} \leq x \leq u\}.$$

- f_j are convex.
- The input data:
  - f is given by an oracle or by an approximation oracle.
  - No other conditions.
  - In general, f_j are non-smooth.
- The goal: An ε-approximate solution, i.e, x with

$$f(x) \leq OPT + \varepsilon.$$

Piecewise linear approximations

- Approximate $f(x)$ by $g(x) = \sum_j g_j(x_j)$ where g_j are piecewise linear.
- The approximate problem:

$$\min\{g(x) : x \in S\}.$$

- The approximate problem is equivalent to an LP where [2]:

  The number of variables = the total number of lin. pieces.

[2] Dantzig, G. 1956. Recent Advances in Linear Programming. Management Science 2, 131-144.

Local piecewise linear approximations

Hochbaum and Shanthikumar[3]:

- The problem is reduced to a sequence of LPs with $8n^2\Delta$ variables, where Δ is the maximum absolute value of determinants of A.
- The number of LPs in the sequence is polynomially bounded.

[3] Hochbaum, D. S. and Shanthikumar J. G. 1990. Convex separable optimization is not much harder than linear optimization. *Journal of the Association for Computing Machinery* **37**, 843-862.

Special cases

- Tseng and Bertsekas [4]: Polynomial time for a generalized network flow problem with convex costs.
- Karzanov and McCormik [5]: Polynomial time when the coefficient matrix is totally unimodular.

[4] Tseng, P., and Bertsekas, D.P. 2000. An ε-relaxation method for separable convex cost generalized network flow problems. *Mathematical Programming* **88**, 85-104.

[5] Karzanov, A. and McCormick, Th. 1997. Polynomial methods for separable convex optimization in unimodualr linear spaces with applications. *SIAM Journal on Computing* **26**, 1245-1275

A scaling algorithm

- $\Gamma =$ the initial objective value $-OPT$.
- $K_{\max} =$ maximum slope.
- T is the running time of the LP algorithm used.
- P is the running time of the oracle for f.

The separable convex problem can be solved by the scaling algorithm in polynomial time[6]:

Theorem

Using any polynomial LP-algorithm, an ε-approximate solution in time

$$O\left(\left(n^3 + T + P \cdot n \cdot \left(\log \frac{nK_{\max}\|u\|_\infty}{\varepsilon}\right)^2\right) \cdot n \cdot \log \frac{n \max\{1, \Gamma\}}{\varepsilon}\right).$$

[6] S. Ch. 2016. A Polynomial-Time Descent Method for Separable Convex Optimization Problems with Linear Constraints. SIAM J. Optim., 26(1), 856-889.

Basic idea: Local approximation in a scaling framework

1. x^k is the current solution in S.
2. Find $g : x \to \sum_{j=1}^n g_j(x_j)$ where each g_j consists of two linear pieces:
   (i) $\max(f, g)$ is a suitable approximation of f :
   $$\max_S (\max(f, g) - f) \le n\delta, \quad g(x^k) = f(x^k).$$
   (ii) There is a neighborhood $B \subseteq \{x : \mathbf{0} \le x \le u\}$ of x^k such that
   $$g(x) \ge f(x) + \frac{\delta}{2}, \forall x \in \partial B.$$
3. Solve $g(x) < g(x^k), x \in S$, (formulated as LP with $2n$ variables):
   - Let x be a solution. An improvement of $f(x^k)$ by $\ge \delta$:
   $$x^{k+1} = [x^k, x] \cap \partial B.$$
   - If no solutions, then $x^k$is $n\delta$-approximate: divide δ by 2.

Summary of the algorithm

- The algorithm can use any LP solver:
    - The approximate piecewise linear problems are formulated as LPs with $2n$ variables.
    - In the case of network flows, this step reduces to finding a negative-cost cycle in the residual graph.
- Algorithm's complexity:
    - The running time is polynomial when the LP solver is polynomial.
    - The sizes of the numbers are polynomial.

Integer linear programming

- An integer linear problem:

$$\min\{c^T x : x \in S\}, S \subset \mathbb{Z}^n, |S| \leq \infty.$$

- A verification oracle: Given an objective function y and $x^0 \in S$, whether x^0 is optimal for y.
- The existing results for $0, 1$-problems do not apply: A reduction by means of binary encodings fails because of the oracle; even if $S \subset \{-1, 0, 1\}$.

Normal fan

- Normal cone:

$$\forall x \in S : C(x) = \{y \in \mathbb{R}^n : y^T x = \min_{x' \in S} y^T x'\}$$

  or

$$\forall x \in S : C(x) = \{y \in \mathbb{R}^n : (x - x')^T y \leq 0, \forall x' \in S\}.$$

- Properties:
  - x is a vertex of $CH(S) \Leftrightarrow \dim C(x) = n$.
  - Full-dim. cones $C(x_1)$ and $C(x_2)$ share a facet (are adjacent) $\Leftrightarrow$ x_1 and x_2 are adjacent in $CH(S)$.
  - The normal fan $\mathcal{F}$: The cell complex formed by the full-dim. normal cones.

Stage 1: General position

$$c = c^0 + (c^1 - c^0) + \dots (c^k - c^{k-1}),$$

where $c^k = c$.
So, at the **first stage** the algorithm finds segments $[z^{i-1}, z^i]$ such that the following conditions are satisfied:

(i) z^i belongs to the interior of a normal cone $C(w^i) \in \mathcal{F}$ such that at the same time $c^i \in C(w^i)$.

(ii) If $[z^{i-1}, z^i]$ intersects a facet Y of some normal cone in $\mathcal{F}$, then it is transverse to Y and the respective intersection point is contained in the relative interior of Y.

(iii) $z^i - z^{i-1} = c^i - c^{i-1}$.

Stage 2

- Find all normal cones intersected by the curve $\cup_i [z^{i-1}, z^i]$.

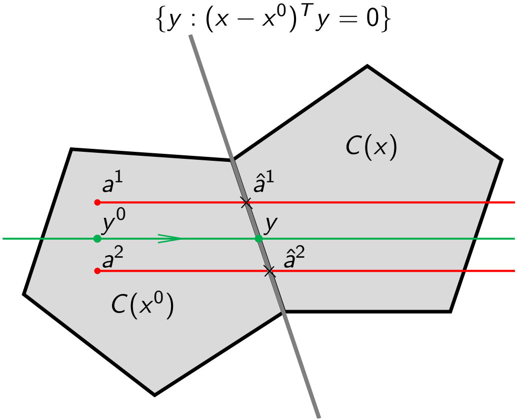

Figure: The green segment is a part of the curve.

Complexity

Theorem

- *The integer linear problem can be solved in oracle time which is polynomial in n and u_i.*
- *If $c^0 \in int(C(x^0))$ and x^0 is known, then the problem can be solved by visiting $\|u\|_1$ vertices of $CH(S)$.*
- *More generally, can be solved by visiting*

$$\sum_{i=1}^{k} |\{(c^i - c^{i-1})^T x : x \in S\}| - k$$

vertices of $CH(S)$, for any given $c^0, \ldots, c^k$ where $c^k = c$ in oracle time polynomial in n and

$$\|c^i - c^{i-1}\| / \gcd(c^i - c^{i-1}), \quad i = 1, \ldots, k.$$