9. Dual decomposition

- dual methods
- dual decomposition
- network utility maximization
- network flow optimization

Dual methods

primal: minimize $f(x) + g(Ax)$

dual: maximize $-g^*(z) - f^*(-A^T z)$

reasons why dual problem may be easier to solve by first-order methods:

- dual problem is unconstrained or has simple constraints (for example, $z \succeq 0$)
- dual objective is differentiable or has a simple nondifferentiable term
- decomposition: exploit separable structure

(Sub-)gradients of conjugate function

assume $f : \mathbf{R}^n \to \mathbf{R}$ is closed and convex with conjugate

$$f^*(y) = \sup_x \left(y^T x - f(x)\right)$$

- f^* is subdifferentiable on (at least) $\mathbf{int}\,\mathbf{dom}\, f^*$ (page 2.4)
- maximizers in the definition of $f^*(y)$ are subgradients at y (page 5.15)

$$y \in \partial f(x) \quad \Longleftrightarrow \quad y^T x - f(x) = f^*(y) \quad \Longleftrightarrow \quad x \in \partial f^*(y)$$

- if f is strictly convex, maximizer is unique (hence, equal to $\nabla f^*(y)$) if it exists
- if f is strongly convex, then conjugate is defined for all y and differentiable with

$$\|\nabla f^*(y) - \nabla f^*(y')\| \leq \frac{1}{\mu}\|y - y'\|_* \quad \text{for all } y, y'$$

where μ is strong convexity constant of f with respect to $\|\cdot\|$; see page 5.19

Outline

- dual methods
- **dual decomposition**
- network utility maximization
- network flow optimization

Equality constraints

Primal and dual problems

$$\begin{array}{lll} \text{primal:} & \text{minimize} & f(x) \\ & \text{subject to} & Ax = b \\ \\ \text{dual:} & \text{maximize} & -b^T z - f^*(-A^T z) \end{array}$$

Dual gradient ascent algorithm (assuming $\operatorname{dom} f^* = \mathbf{R}^n$)

$$\begin{array}{rcl} \hat{x} & = & \underset{x}{\operatorname{argmin}} \, (f(x) + z^T A x) \\ z^+ & = & z - t(b - A\hat{x}) \end{array}$$

- step one computes a subgradient $\hat{x} \in \partial f^*(-A^T z)$
- in step two, $b - A\hat{x}$ is a subgradient of $b^T z + f^*(-A^T z)$ at z

of interest if calculation of $\hat{x}$ is inexpensive (for example, f is separable)

Dual decomposition

Convex problem with separable objective

$$\begin{array}{ll} \text{minimize} & f_1(x_1) + f_2(x_2) \\ \text{subject to} & A_1x_1 + A_2x_2 \preceq b \end{array}$$

constraint is *complicating* or *coupling* constraint

Dual problem

$$\begin{array}{ll} \text{maximize} & -f_1^*(-A_1^Tz) - f_2^*(-A_2^Tz) - b^Tz \\ \text{subject to} & z \succeq 0 \end{array}$$

can be solved by (sub-)gradient projection method if $z \succeq 0$ is the only constraint

Dual subgradient projection

Subproblem: to calculate $f_j^*(-A_j^T z)$ and a (sub-)gradient for it,

$$\text{minimize (over } x_j) \quad f_j(x_j) + z^T A_j x_j$$

- optimal value is $-f_j^*(-A_j^T z)$
- minimizer $\hat{x}_j$ is in $\partial f_j^*(-A_j^T z)$

Dual subgradient projection method

$$\begin{aligned} \hat{x}_j &= \underset{x_j}{\operatorname{argmin}} \left(f_j(x_j) + z^T A_j x_j \right) \quad \text{for } j = 1, 2 \\ z^+ &= (z - t(b - A_1\hat{x}_1 - A_2\hat{x}_2))_+ \end{aligned}$$

- minimization problems over x_1, x_2 are independent
- z-update is projected subgradient step ($u_+ = \max\{u, 0\}$ elementwise)

Interpretation as price coordination

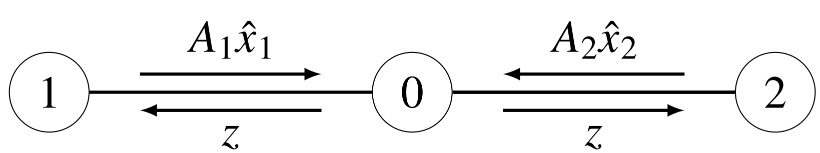
- $p = 2$ units in a system; unit j chooses decision variable x_j
- constraints are limits on shared resources; z_i is price of resource i

Dual update: depends on slacks $s = b - A_1x_1 - A_2x_2$

$$z^+ = (z - ts)_+$$

- increases price z_i if resource i is over-utilized ($s_i < 0$)
- decreases price z_i if resource i is under-utilized ($s_i > 0$)
- never lets prices get negative

Distributed architecture: central node sets prices z, peripheral node j sets x_j



Example

Quadratic optimization problem

$$
\begin{array}{ll}
\text{minimize} & \sum_{j=1}^{r} \left(\frac{1}{2}x_j^T P_j x_j + q_j^T x_j\right) \\
\text{subject to} & B_j x_j \preceq d_j, \quad j = 1, \ldots, r \\
& \sum_{j=1}^{r} A_j x_j \preceq b
\end{array}
$$

- without last inequality, problem would separate into r independent QPs
- we assume $P_j \succ 0$

Formulation for dual decomposition

$$
\begin{array}{ll}
\text{minimize} & \sum_{j=1}^{r} f_j(x_j) \\
\text{subject to} & \sum_{j=1}^{r} A_j x_j \preceq b
\end{array}
$$

where $f_j(x_j) = (1/2)x_j^T P_j x_j + q_j^T x_j$ with domain $\{x_j \mid B_j x_j \preceq d_j\}$

Dual problem

$$\begin{array}{ll} \text{maximize} & -b^T z - \sum_{j=1}^{r} f_j^*(-A_j^T z) \\ \text{subject to} & z \succeq 0 \end{array}$$

- gradient of $h(z) = \sum_j f_j^*(-A_j^T z)$ is Lipschitz continuous (since $P_j \succ 0$):

$$\|\nabla h(z) - \nabla h(z')\|_2 \leq \frac{\|A\|_2^2}{\min_j \lambda_{\min}(P_j)} \|z - z'\|_2$$

  where $A = [\ A_1 \quad \cdots \quad A_r\ ]$

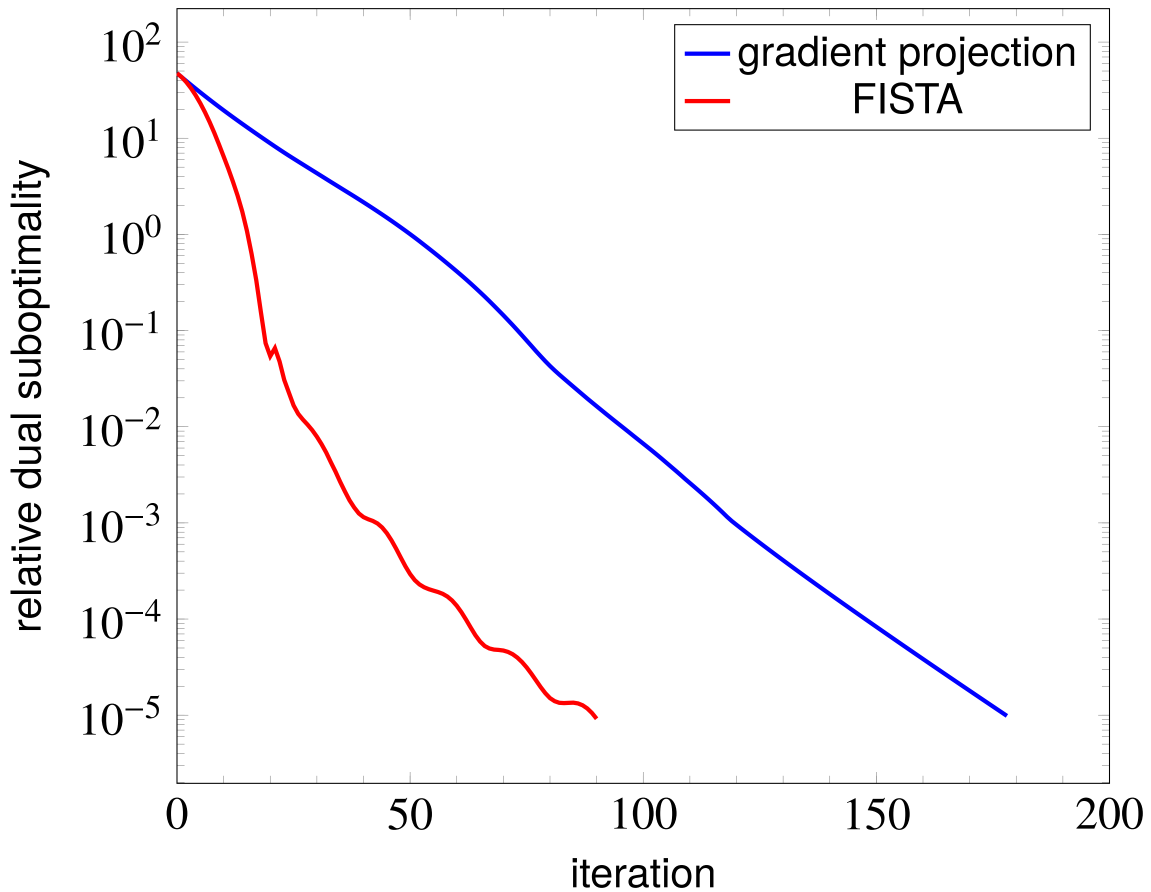
- function value of $-f_j^*(-A_j^T z)$ is the optimal value of the QP

$$\begin{array}{ll} \text{minimize (over } x_j) & (1/2)x_j^T P x_j + (q_j + A_j^T z)^T x_j \\ \text{subject to} & B_j x_j \preceq d_j \end{array}$$

- optimal solution $\hat{x}_j$ is gradient $\hat{x}_j = \nabla f_j^*(-A_j^T z)$

Numerical example

- 10 subproblems ($r = 10$), each with 100 variables and 100 constraints
- 10 coupling constraints
- projected gradient descent and FISTA, with the same fixed step size



Outline

- dual methods
- dual decomposition
- **network utility maximization**
- network flow optimization

Network utility maximization

Network flows

- n flows, with fixed routes, in a network with m links
- variable $x_j \geq 0$ denotes the rate of flow j
- flow utility is $U_j : \mathbf{R} \to \mathbf{R}$, concave, increasing

Capacity constraints

- traffic y_i on link i is sum of flows passing through it
- $y = Rx$, where R is the routing matrix

$$R_{ij} = \begin{cases} 1 & \text{flow } j \text{ passes over link } i \\ 0 & \text{otherwise} \end{cases}$$

- link capacity constraint: $y \preceq c$

Dual network utility maximization problem

primal: maximize $\sum_{j=1}^{n} U_j(x_j)$

subject to $Rx \preceq c$

dual: minimize $c^T z + \sum_{j=1}^{n} (-U_j)^*(-r_j^T z)$

subject to $z \succeq 0$

- r_j is column j of R
- dual variable z_i is price (per unit flow) for using link i
- $r_j^T z$ is the sum of prices along route j

(Sub-)gradients of dual function

Dual objective

$$
\begin{aligned}
f(z) &= c^T z + \sum_{j=1}^{n} (-U_j)^*(-r_j^T z) \\
&= c^T z + \sum_{j=1}^{n} \sup_{x_j} \left(U_j(x_j) - (r_j^T z) x_j \right)
\end{aligned}
$$

Subgradient

$$
c - R\hat{x} \in \partial f(z) \quad \text{where} \quad \hat{x}_j = \operatorname*{argmax}_{x_j} \left(U_j(x_j) - (r_j^T z) x_j \right)
$$

- $r_j^T z$ is the sum of link prices along route j
- $c - R\hat{x}$ is vector of link capacity margins for flow $\hat{x}$
- if U_j is strictly concave, this is a gradient

Dual decomposition algorithm

given initial link price vector z, repeat:

1. sum link prices along each route: calculate $\lambda_j = r_j^T z$ for $j = 1, \ldots, n$
2. optimize flows (separately) using flow prices

$$\hat{x}_j = \underset{x_j}{\operatorname{argmax}} \left(U_j(x_j) - \lambda_j x_j \right), \quad j = 1, \ldots, n$$

3. calculate link capacity margins $s = c - R\hat{x}$
4. update link prices using projected (sub-)gradient step with step t

$$z := (z - ts)_+$$

Decentralized:

- to find λ_j, $\hat{x}_j$ source j only needs to know the prices on its route
- to update s_i, z_i, link i only needs to know the flows that pass through it

Outline

- dual methods
- dual decomposition
- network utility maximization
- **network flow optimization**

Single commodity network flow

Network

- connected, directed graph with n links/arcs, m nodes
- node-arc incidence matrix $A \in \mathbf{R}^{m\times n}$ is

$$A_{ij} = \begin{cases} 1 & \text{arc } j \text{ enters node } i \\ -1 & \text{arc } j \text{ leaves node } i \\ 0 & \text{otherwise} \end{cases}$$

Flow vector and external sources

- variable x_j denotes flow (traffic) on arc j
- b_i is external demand (or supply) of flow at node i (satisfies $\mathbf{1}^T b = 0$)
- flow conservation: $Ax = b$

Network flow optimization problem

$$\begin{array}{ll} \text{minimize} & \phi(x) = \sum_{j=1}^{n} \phi_j(x_j) \\ \text{subject to} & Ax = b \end{array}$$

- ϕ is a separable sum of convex functions
- dual decomposition yields decentralized solution method

Dual problem (a_j is jth column of A)

$$\text{maximize} \quad -b^T z - \sum_{j=1}^{n} \phi_j^*(-a_j^T z)$$

- dual variable z_i can be interpreted as potential at node i
- $y_j = -a_j^T z$ is the potential difference across arc j
  (potential at start node minus potential at end node)

(Sub-)gradients of dual function

Negative dual objective

$$f(z) = b^T z + \sum_{j=1}^{n} \phi_j^*(-a_j^T z)$$

Subgradient

$$b - A\hat{x} \in \partial f(z) \quad \text{where} \quad \hat{x}_j = \operatorname{argmin} \left(\phi_j(x_j) + (a_j^T z) x_j \right)$$

- this is a gradient if the functions ϕ_j are strictly convex
- if ϕ_j is differentiable, $\phi_j'(\hat{x}_j) = -a_j^T z$

Dual decomposition network flow algorithm

given initial potential vector z, repeat

1. determine link flows from potential differences $y = -A^T z$

$$\hat{x}_j = \underset{x_j}{\operatorname{argmin}} \left(\phi_j(x_j) - y_j x_j\right), \quad j = 1, \ldots, n$$

2. compute flow residual at each node: $s := b - A\hat{x}$
3. update node potentials using (sub-)gradient step with step size t

$$z := z - ts$$

Decentralized:

- flow $\hat{x}_j$ is calculated from potential difference across arc j
- node potential z_i is updated from its own flow residual s_i

Electrical network interpretation

network flow optimality conditions (with differentiable ϕ_j)

$$Ax = b, \qquad y + A^T z = 0, \qquad y_j = \phi_j'(x_j), \quad j = 1, \ldots, n$$

network with node incidence matrix A, nonlinear resistors in branches

Kirchhoff current law (KCL): $Ax = b$

x_j is the current flow in branch j; b_i is external current extracted at node i

Kirchhoff voltage law (KVL): $y + A^T z = 0$

z_j is node potential; $y_j = -a_j^T z$ is jth branch voltage

Current–voltage characterics: $y_j = \phi_j'(x_j)$

for example, $\phi_j(x_j) = R_j x_j^2/2$ for linear resistor R_j

current and potentials in circuit are optimal flows and dual variables

Example: minimum queueing delay

Flow cost function and conjugate ($c_j > 0$ is link capacity):

$$\phi_j(x_j) = \frac{x_j}{c_j - x_j}, \qquad \phi_j^*(y_j) = \begin{cases} (\sqrt{c_j y_j} - 1)^2 & y_j > 1/c_j \\ 0 & y_j \leq 1/c_j \end{cases}$$

with $\operatorname{dom} \phi_j = [0, c_j)$

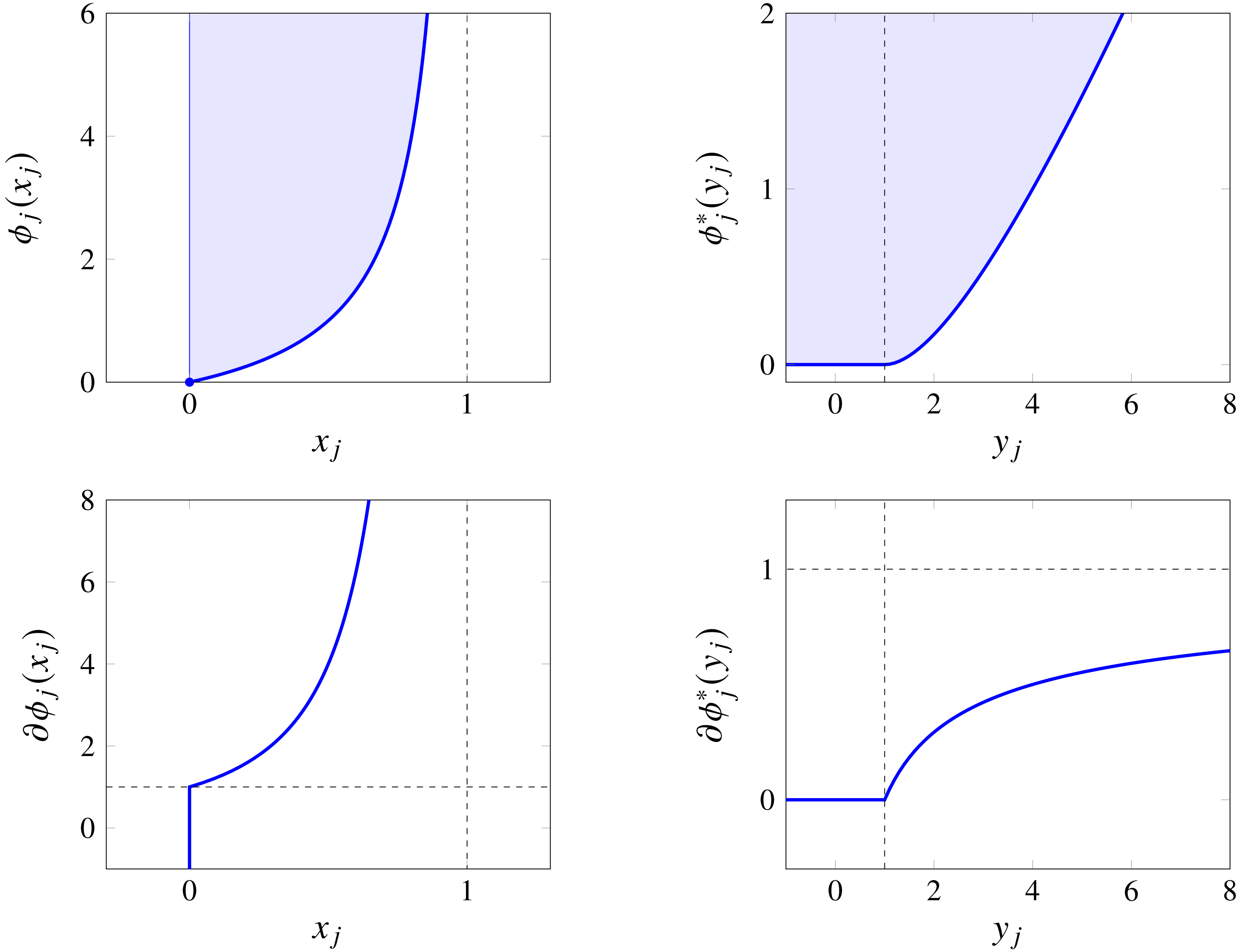
- ϕ_j is differentiable except at $x_j = 0$

$$\partial \phi_j(0) = (-\infty, 0], \qquad \phi_j'(x_j) = \frac{c_j}{(c_j - x_j)^2} \quad (0 < x_j < c_j)$$

- ϕ_j^* is differentiable

$$\phi_j^{*\prime}(y_j) = \begin{cases} 0 & y_j \leq 1/c_j \\ c_j - \sqrt{c_j / y_j} & y_j > 1/c_j \end{cases}$$

Flow cost function, conjugate, and their subdifferentials ($c_j = 1$)



References

- S. Boyd, *Lecture slides and notes for EE364b, Convex Optimization II*, lectures and notes on decomposition.
- M. Chiang, S.H. Low, A.R. Calderbank, J.C. Doyle, *Layering as optimization decomposition: A mathematical theory of network architectures*, Proceedings IEEE (2007).
- D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods* (1989).
- D.P. Bertsekas, *Network Optimization. Continuous and Discrete Models* (1998).
- L.S. Lasdon, *Optimization Theory for Large Systems* (1970).