# Algorithms for constrained optimization

- Dual Derivatives and Subgradients

For a given $\boldsymbol{\mu} \in \mathbb{R}^r$, suppose that $\mathbf{x}_\mu$ minimizes the Lagrangian $L(\mathbf{x}, \boldsymbol{\mu})$,

$$\mathbf{x}_\mu = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} L(\mathbf{x}, \boldsymbol{\mu}) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}) \right\}.$$

An important fact is that $\mathbf{g}(\mathbf{x}_\mu)$ is a supergradient of the dual function $q$ at $\boldsymbol{\mu}$:

obtained *essentially at no cost*

$$q(\bar{\boldsymbol{\mu}}) \leq q(\boldsymbol{\mu}) + (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \mathbf{g}(\mathbf{x}_\mu), \quad \forall \bar{\boldsymbol{\mu}} \in \mathbb{R}^r. \tag{10}$$

To see this, we use the definition of $q$ and $\mathbf{x}_\mu$ to write for all $\bar{\boldsymbol{\mu}} \in \mathbb{R}^r$,

$$\begin{aligned} q(\bar{\boldsymbol{\mu}}) = \inf_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) + \bar{\boldsymbol{\mu}}^T \mathbf{g}(\mathbf{x}) \right\} &\leq f(\mathbf{x}_\mu) + \bar{\boldsymbol{\mu}}^T \mathbf{g}(\mathbf{x}_\mu) \\ &= f(\mathbf{x}_\mu) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}_\mu) + (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \mathbf{g}(\mathbf{x}_\mu) \\ &= q(\boldsymbol{\mu}) + (\bar{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \mathbf{g}(\mathbf{x}_\mu). \end{aligned}$$

Note that this calculation is valid for all $\boldsymbol{\mu} \in \mathbb{R}^r$ for which there is a minimizing vector $\mathbf{x}_\mu$, regardless of whether $\boldsymbol{\mu} \geq \mathbf{0}$.

# Algorithms for constrained optimization

- Dual Derivatives and Subgradients

**Proposition 1.** *Let $\mathcal{X}$ be a compact set, and let $f$ and $\mathbf{g}$ be continuous over $\mathcal{X}$. Assume also that for every $\boldsymbol{\mu} \in \mathbb{R}^r$, $L(\mathbf{x}, \boldsymbol{\mu})$ is minimized over $\mathbf{x} \in \mathcal{X}$ at a unique point $\mathbf{x}_\mu$. Then $q$ is everywhere continuously differentiable and*

$$\nabla q(\boldsymbol{\mu}) = \mathbf{g}(\mathbf{x}_\mu), \quad \forall \boldsymbol{\mu} \in \mathbb{R}^r.$$

# Algorithms for constrained optimization

- Lagrangian Algorithms - Equality Constraints

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ s.t. \quad & \mathbf{h}(\mathbf{x}) = \mathbf{0}. \end{aligned}$$

The Lagrangian function is given by

$$l(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}). \tag{10}$$

The Lagrangian algorithm for this problem is given by

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \alpha_k (\nabla f(\mathbf{x}^{(k)}) + D\mathbf{h}(\mathbf{x}^{(k)})^T \boldsymbol{\lambda}^{(k)}), \\ \boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \beta_k \mathbf{h}(\mathbf{x}^{(k+1)}). \end{aligned} \tag{11}$$

The update equation for $\mathbf{x}^{(k)}$ is a gradient algorithm for minimizing the Lagrangian with respect to $\mathbf{x}$, and the update equation for $\boldsymbol{\lambda}^{(k)}$ is a gradient algorithm for maximizing the Lagrangian with respect to $\boldsymbol{\lambda}$.

# Algorithms for constrained optimization

- Lagrangian Algorithms - Equality Constraints

The following lemma establishes that if the algorithm converges, the limit must satisfy the Lagrange condition. More specifically, the lemma states that any fixed point of the algorithm must satisfy the Lagrange condition. A fixed point of an update algorithm is simply a point with the property that when updated using the algorithm, the resulting point is equal to the given point. For the case of the Lagrangian algorithm, which updates both $\mathbf{x}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$ vectors, a fixed point is a pair of vectors. If the Lagrangian algorithm converges, the limit must be a fixed point.

**Lemma 1.** *For the Lagrangian algorithm for updating* $\mathbf{x}^{(k)}$ *and* $\boldsymbol{\lambda}^{(k)}$*, the pair* $(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)})$ *is a fixed point iff it satisfies the Lagrange condition.*

# Algorithms for constrained optimization

- Lagrangian Algorithms - Equality Constraints

Below, we use $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ to denote a pair satisfying the Lagrange condition. Assume that $\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \succ \mathbf{0}$. Also assume that $\mathbf{x}^*$ is a regular point. With these assumptions, we are now ready to state and prove that the algorithm is locally convergent. For simplicity, we will take $\alpha_k$ and $\beta_k$ to be fixed constants (not depending on $k$), denoted $\alpha$ and $\beta$, respectively.

**Theorem 1.** *For the Lagrangian algorithm for updating* $\mathbf{x}^{(k)}$ *and* $\boldsymbol{\lambda}^{(k)}$*, provided that* $\alpha$ *and* $\beta$ *are sufficiently small, there is a neighborhood of* $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ *such that if the pair* $(\mathbf{x}^0, \boldsymbol{\lambda}^0)$ *is in this neighborhood, then the the algorithm converges to* $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ *with at least a linear order of convergence.*

# Algorithms for constrained optimization

- Lagrangian Algorithms - Inequality Constraints

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ s.t. \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0}. \end{aligned}$$

The Lagrangian function is given by

$$l(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}). \tag{10}$$

The Lagrangian algorithm for this problem is given by

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \alpha_k (\nabla f(\mathbf{x}^{(k)}) + Dg(\mathbf{x}^{(k)})^T \boldsymbol{\mu}^{(k)}), \\ \boldsymbol{\mu}^{(k+1)} &= [\boldsymbol{\mu}^{(k)} + \beta_k \mathbf{g}(\mathbf{x}^{(k+1)})]_+, \end{aligned}$$

where $[\cdot]+ = \max\{\cdot, 0\}$. The update equation for $\mathbf{x}^{(k)}$ is a gradient algorithm for minimizing the Lagrangian with respect to its $\mathbf{x}$ argument. The update equation for $\boldsymbol{\mu}^{(k)}$ is a projected gradient algorithm for maximizing the Lagrangian with respect to its $\boldsymbol{\mu}$ argument.

# Algorithms for constrained optimization

- Penalty Methods

$$\min \ f(\mathbf{x}) \tag{11}$$
$$s.t. \ \mathbf{x} \in \Omega.$$

$$\min \ f(\mathbf{x}) + \gamma P(\mathbf{x}), \tag{12}$$

penalty parameter

penalty function

**Definition 1.** *A function* $P : \mathbb{R}^n \to \mathbb{R}$ *is called a* penalty function *for the constrained optimization problem above if it satisfies the following three conditions:*

1. $P$ *is continuous.*
2. $P(\mathbf{x}) \geq 0$ *for all* $\mathbf{x} \in \mathbb{R}^n$.
3. $P(\mathbf{x}) = 0$ *if and only if* $\mathbf{x}$ *is feasible (i.e.,* $\mathbf{x} \in \Omega$*).*

# Algorithms for constrained optimization

- Penalty Methods – Choice of penalty function

Example.

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ s.t. \quad & g_i(\mathbf{x}) \leq 0, i = 1, \cdots, p. \end{aligned}$$

A possible choice for $P$ is

$$P(\mathbf{x}) = \sum_{i=1}^{p} g_i^+(\mathbf{x}),$$

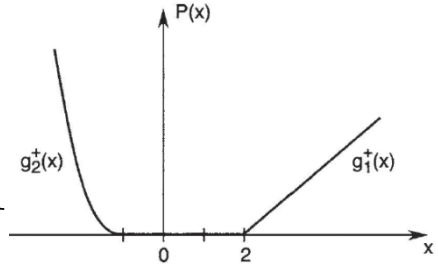
where

$$g_i^+(\mathbf{x}) = \max\{0, g_i(x)\} = \begin{cases} 0, & g_i(\mathbf{x}) \leq 0 \\ g_i(\mathbf{x}), & g_i(\mathbf{x}) > 0. \end{cases}$$

We refer to this penalty function as the absolute value penalty function, because it is equal to $\sum |g_i(\mathbf{x})|$, where the summation is taken over all constraints that are violated at $\mathbf{x}$.

# Algorithms for constrained optimization

- Penalty Methods – Choice of penalty function




Example: Let $g_1, g_2 : \mathbb{R} \to \mathbb{R}$ be defined by $g_1(x) = x - 2, g_2(x) = -(x+1)^3$. The feasible set defined by $\{x \in \mathbb{R} : g_1(x) \leq 0, g_2(x) \leq 0\}$ is simply the interval $[-1, 2]$. In this example, we have

$$g_1^+(x) = \max\{0, g_1(x)\} = \begin{cases} 0, & x \leq 2 \\ x - 2, & otherwise, \end{cases}$$

$$g_2^+(x) = \max\{0, g_2(x)\} = \begin{cases} 0, & x \geq -1 \\ -(x+1)^3, & otherwise, \end{cases}$$

and

$$P(x) = g_1^+(x) + g_2^+(x) = \begin{cases} x - 2, & x > 2 \\ 0, & -1 \leq x \leq 2 \\ -(x+1)^3, & x < -1. \end{cases}$$

# Algorithms for constrained optimization

- Penalty Methods – Choice of penalty function

The absolute value penalty function may not be differentiable at points $\mathbf{x}$ where $g_i(\mathbf{x}) = 0$. Therefore, in such cases we cannot use techniques for optimization that involve derivatives. A form of the penalty function that is guaranteed to be differentiable is the *Courant-Beltrami penalty function*, given by

$$P(\mathbf{x}) = \sum_{i=1}^{p} (g_i^+(\mathbf{x}))^2.$$

# Algorithms for constrained optimization

- Penalty Methods - Convergence

Denote by $\mathbf{x}^*$ a solution (global minimizer) to the problem. Let $P$ be a penalty function for the problem. For each $k = 1, 2, ...,$ let $\gamma_k \in \mathbb{R}$ be a given positive constant. Define an associated function $q(\gamma_k, \cdot) : \mathbb{R}^n \to \mathbb{R}$ by

$$q(\gamma_k, \mathbf{x}) = f(\mathbf{x}) + \gamma_k P(\mathbf{x}).$$

For each $k$, we can write the following associated unconstrained optimization problem:

$$\min q(\gamma_k, \mathbf{x}).$$

Denote by $\mathbf{x}^{(k)}$ a minimizer of $q(\gamma_k, \mathbf{x})$.

# Algorithms for constrained optimization

- Penalty Methods - Convergence

**Lemma 1.** *Suppose that $\{\gamma_k\}$ is a nondecreasing sequence; that is, for each $k$, we have $\gamma_k < \gamma_{k+1}$. Then, for each $k$ we have*

1. $q(\gamma_{k+1}, \mathbf{x}^{(k+1)}) \geq q(\gamma_k, \mathbf{x}^{(k)})$.
2. $P(\mathbf{x}^{(k+1)}) \leq P(\mathbf{x}^{(k)})$.
3. $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$.
4. $f(\mathbf{x}^*) \geq q(\gamma_k, \mathbf{x}^{(k)}) \geq f(\mathbf{x}^{(k)})$.

**Theorem 2.** *Suppose that the objective function $f$ is continuous and $\gamma_k \to \infty$ as $k \to \infty$. Then, the limit of any convergent subsequence of the sequence $\{\mathbf{x}^{(k)}\}$ is a solution to the constrained optimization problem.*

# Algorithms for constrained optimization

- Penalty Methods – Exact penalty

We desire an exact solution to the original constrained problem by solving the associated unconstrained problem $\min_{\mathbf{x}} \; f(\boldsymbol{x}) + \gamma P(\boldsymbol{x})$ with a finite $\gamma > 0$. It turns out that indeed this can be accomplished, in which case we say that the penalty function is *exact*. However, it is necessary that exact penalty functions be nondifferentiable.

Example:

$$\begin{aligned} &\min \; f(x) \\ &s.t. \; x \in [0, 1], \end{aligned}$$

where $f(x) = 5 - 3x$.

# Algorithms for constrained optimization

- Penalty Methods – Exact penalty

**Proposition 1.** *Consider the problem*

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &s.t.\ \mathbf{x} \in \Omega, \end{aligned}$$

*with $\Omega \subset \mathbb{R}^n$ convex. Suppose that the minimizer $\mathbf{x}^*$ lies on the boundary of $\Omega$ and there exists a feasible direction $\mathbf{d}$ at $\mathbf{x}^*$ such that $\mathbf{d}^T \nabla f(\mathbf{x}^*) > 0$. If $P$ is an exact penalty function, then $P$ is not differentiable at $\mathbf{x}^*$.*

*Proof.* We use contraposition. Suppose that $P$ is differentiable at $\mathbf{x}^*$. Then, $\mathbf{d}^T \nabla P(\mathbf{x}^*) = 0$, because $P(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega$. Hence, if we let $g = f + \gamma P$, then $\mathbf{d}^T \nabla g(\mathbf{x}^*) > 0$ for all finite $\gamma > 0$, which implies that $\nabla g(\mathbf{x}^*) \neq 0$. Hence, $\mathbf{x}^*$ is not a local minimizer of g, and thus $P$ is not an exact penalty function. $\square$

# Algorithms for constrained optimization

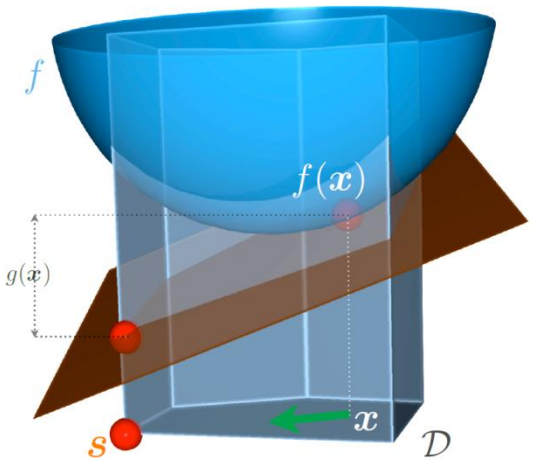
- Frank-Wolfe Algorithm

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{D}, \tag{1}$$

where $f$ is convex and continuously differentiable and $\mathcal{D}$ is a compact convex set.

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle$$

$$\begin{aligned} \mathbf{g}_k &= \operatorname*{argmin}_{\mathbf{g} \in \mathcal{D}} \langle \mathbf{g}, \nabla f(\mathbf{x}_k) \rangle, \\ \mathbf{x}_{k+1} &= (1 - \gamma_k)\mathbf{x}_k + \gamma_k \mathbf{g}_k, \quad \text{where } \gamma_k = \frac{2}{k+2}. \end{aligned} \tag{2}$$



# Algorithms for constrained optimization

- Frank-Wolfe Algorithm

The Frank-Wolfe algorithm is advantageous when

$$\mathbf{g}_k = \underset{\mathbf{g} \in \mathcal{D}}{\operatorname{argmin}} \langle \mathbf{g}, \nabla f(\mathbf{x}_k) \rangle$$

is easy to compute, e.g., when $\mathcal{D}$ is the unit ball of a norm, such as nuclear norm.

# Algorithms for constrained optimization

- Frank-Wolfe Algorithm - Convergence

**Lemma 1.** *For an iteration* $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma(\mathbf{g}_k - \mathbf{x}_k)$ *with an arbitrary stepsize* $\gamma \in [0, 1]$*, it holds that*

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \gamma g(\mathbf{x}_k) + \frac{\gamma^2}{2} C_f, \tag{3}$$

*if* $\mathbf{g}_k$ *is an approximate linear minimizer, i.e.,*

$$\langle \mathbf{g}_k, \nabla f(\mathbf{x}_k) \rangle = \min_{\hat{\mathbf{g}}_k \in \mathcal{D}} \langle \hat{\mathbf{g}}_k, \nabla f(\mathbf{x}_k) \rangle.$$

*Here* $g(\mathbf{x})$ *is the duality gap defined as*

$$g(\mathbf{x}) = \max_{\mathbf{g} \in \mathcal{D}} \langle \mathbf{x} - \mathbf{g}, \nabla f(\mathbf{x}) \rangle. \tag{4}$$

# Algorithms for constrained optimization

- Frank-Wolfe Algorithm - Convergence

**Theorem 1.** *For each $k \geq 1$, the iterate $\mathbf{x}_k$ in procedure (2) satisfies*

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{2C_f}{k+2}, \tag{5}$$

*where $\mathbf{x}^* \in \mathcal{D}$ is an optimal solution to problem (1) and $C_f$ is the curvature constant defined as*

$$C_f = \sup_{\mathbf{x},\mathbf{s}\in\mathcal{D},\gamma\in[0,1],\mathbf{y}=\mathbf{x}+\gamma(\mathbf{s}-\mathbf{x})} \frac{2}{\gamma^2}(f(\mathbf{y}) - f(\mathbf{x}) - \langle \mathbf{y}-\mathbf{x}, \nabla f(\mathbf{x})\rangle). \tag{6}$$

For $L$-smooth $f$, $C_f \leq L \max_{\mathbf{s}\in\mathcal{D}} \|\mathbf{s}-\mathbf{x}\|^2$.

# Algorithms for constrained optimization

- Frank-Wolfe Algorithm - Example

$$\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{y} - \sum_{i=1}^{N} x_i \mathbf{d}_i\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{7}$$

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2, \quad \Leftrightarrow \quad \min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{y}/s - \mathbf{D}\mathbf{x}\|_2^2, \\ &s.t. \quad \|\mathbf{x}\|_1 \leq s. \qquad\qquad\qquad s.t. \quad \|\mathbf{x}\|_1 \leq 1. \end{aligned} \tag{8}$$

$$\nabla f(\mathbf{x}) = \mathbf{D}^\top(\mathbf{D}\mathbf{x} - \mathbf{y}). \tag{9}$$

Now assume that $\mathbf{z}_t = \mathbf{D}\mathbf{x}_t - \mathbf{y} \in \mathbb{R}^n$ is already computed, then to compute $(\mathbf{u}_t \in \operatorname{argmin}_{\mathbf{u}\in\mathcal{X}} \nabla f(\mathbf{x}_t)^\top \mathbf{u})$ one needs to find the coordinate $i_t \in [N]$ that maximizes $|[\nabla f(\mathbf{x}_t)](i)|$ which can be done by maximizing $\mathbf{d}_i^\top \mathbf{z}_t$ and $-\mathbf{d}_i^\top \mathbf{z}_t$.

$$\begin{aligned} \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_\infty = \|\mathbf{D}^\top\mathbf{D}(\mathbf{x}-\mathbf{y})\|_\infty &= \max_{1\leq i\leq N} \left|\mathbf{d}_i^\top \mathbf{D}(\mathbf{x}-\mathbf{y})\right| \\ &\leq \max_{1\leq i\leq N} \|\mathbf{d}_i^\top \mathbf{D}\|_\infty \|\mathbf{x}-\mathbf{y}\|_1 \leq m^2\|\mathbf{x}-\mathbf{y}\|_1. \end{aligned}$$