Conjugate functions. Dual (sub)gradient method. Augmented Lagrangian method. ADMM.

Seminar

Optimization for ML. Faculty of Computer Science. HSE University



Definition



Recall that given $f: \mathbb{R}^n \rightarrow \mathbb{R},$ the function defined by

$$f^*(y) = \max_{x} \left[y^T x - f(x) \right]$$

is called its conjugate.

Conjugate function properties

Recall that given $f: \mathbb{R}^n \to \mathbb{R}$, the function defined by

$$f^*(y) = \max_{x} \left[y^T x - f(x) \right]$$

is called its conjugate.

• Conjugates appear frequently in dual programs, since

$$-f^*(y) = \min_{x} \left[f(x) - y^T x \right]$$

• If f is closed and convex, then $f^{**} = f$. Also,

$$x \in \partial f^*(y) \Leftrightarrow y \in \partial f(x) \Leftrightarrow x \in \arg\min_{z} \left[f(z) - y^T z \right]$$

• If f is strictly convex, then

$$\nabla f^*(y) = \arg\min_{z} \left[f(z) - y^T z \right]$$



Slopes of f and f^*

Assume that f is a closed and convex function. Then f is strongly convex with parameter $\mu \Leftrightarrow \nabla f^*$ is Lipschitz with parameter $1/\mu$.





i Question

$$f_1(x) = a^T x + b$$



i Question

$$f_1(x) = a^T x + b$$

$$f^*(s) = \sup_{x \in \mathbb{R}^n} \left(s^T x - a^T x - b \right) = \begin{cases} -b, & \text{if } s = a \\ \infty, & \text{else} \end{cases} = \delta\left([s = a] \right) - b$$
$$\operatorname{dom} f^*(s) = \{a\}$$



i Question

Find the conjugate function for

$$f_1(x) = a^T x + b$$

$$f^*(s) = \sup_{x \in \mathbb{R}^n} \left(s^T x - a^T x - b \right) = \begin{cases} -b, & \text{if } s = a \\ \infty, & \text{else} \end{cases} = \delta\left([s = a] \right) - b \\ \operatorname{dom} f^*(s) = \{a\} \end{cases}$$

i Question

$$f_2(s) = \delta\left([s=a]\right) - b$$



i Question

Find the conjugate function for

$$f_1(x) = a^T x + b$$

$$f^*(s) = \sup_{x \in \mathbb{R}^n} \left(s^T x - a^T x - b \right) = \begin{cases} -b, & \text{if } s = a \\ \infty, & \text{else} \end{cases} = \delta\left([s = a] \right) - b \\ \text{dom} f^*(s) = \{a\} \end{cases}$$

i Question

$$f_2(s) = \delta\left([s=a]\right) - b$$

$$(\delta([s = a]) - b)^* = \sup_{s \in \mathsf{dom}f_2(s)} (y^T s - \delta([s = a]) + b) = a^T y + b$$



i Question

Find the conjugate function for

 $f(x) = \log(1 + \exp(x))$



i Question

$$f(x) = \log(1 + \exp(x))$$

$$f^*(s) = \sup_{x \in \mathbb{R}^n} \left(sx - \log(1 + \exp(x)) \right)$$



i Question

Find the conjugate function for

$$f(x) = \log(1 + \exp(x))$$

$$f^{*}(s) = \sup_{x \in \mathbb{R}^{n}} (sx - \log(1 + \exp(x)))$$
$$f^{*}(s) = \begin{cases} \infty, & \text{if } s < 0\\ 0, & \text{if } s = 0\\ 0, & \text{if } s = 1\\ \infty, & \text{if } s > 1\\ ?, & \text{if } 0 < s < 1 \end{cases}$$

 $f \rightarrow \min_{x,y,z}$ Reminder: conjugate functions

i Question

Find the conjugate function for

$$f(x) = \log(1 + \exp(x))$$

 $s \in (0, 1)$:

$$s - \frac{\exp(x)}{1 + \exp(x)} = 0 \Leftrightarrow x_{opt} = \log \frac{s}{1 - s}$$

i Question

Find the conjugate function for

$$f(x) = \log(1 + \exp(x))$$

 $s \in (0, 1)$:

$$s - \frac{\exp(x)}{1 + \exp(x)} = 0 \Leftrightarrow x_{opt} = \log \frac{s}{1 - s}$$

Thus,

$$f^*(s) = \begin{cases} 0, & \text{if } s \in \{0, 1\}\\ s \log s + (1 - s) \log(1 - s), & \text{if } 0 < s < 1\\ \infty, & \text{else} \end{cases}$$

 $\mathsf{dom} f^*(s) = [0,1]$



Even if we can't derive dual (conjugate) in closed form, we can still use dual-based gradient or subgradient methods. Consider the problem:

 $\min_{x} \quad f(x) \quad \text{subject to} \quad Ax = b$



Even if we can't derive dual (conjugate) in closed form, we can still use dual-based gradient or subgradient methods. Consider the problem:

$$\min_{x} \quad f(x) \quad \text{subject to} \quad Ax = b$$

Its dual problem is:

$$\max_{u} \quad -f^*(-A^T u) - b^T u$$

where f^* is the conjugate of f. Defining $g(u) = -f^*(-A^T u) - b^T u$, note that:

$$\partial g(u) = A \partial f^*(-A^T u) - b$$



Even if we can't derive dual (conjugate) in closed form, we can still use dual-based gradient or subgradient methods. Consider the problem:

$$\min_{x} \quad f(x) \quad \text{subject to} \quad Ax = b$$

Its dual problem is:

$$\max_{u} \quad -f^*(-A^T u) - b^T u$$

where f^* is the conjugate of f. Defining $g(u) = -f^*(-A^T u) - b^T u$, note that: $\partial g(u) = A \partial f^*(-A^T u) - b$

Therefore, using what we know about conjugates

$$\partial g(u) = Ax - b$$
 where $x \in \arg\min_{z} \left[f(z) + u^T Az \right]$



Even if we can't derive dual (conjugate) in closed form, we can still use dual-based gradient or subgradient methods. Consider the problem:

$$\min_{x} \quad f(x) \quad \text{subject to} \quad Ax = b$$

Its dual problem is:

$$\max_{u} \quad -f^*(-A^T u) - b^T u$$

where f^* is the conjugate of f. Defining $g(u) = -f^*(-A^T u) - b^T u$, note that:

$$\partial g(u) = A \partial f^*(-A^T u) - b$$

Therefore, using what we know about conjugates

$$\partial g(u) = Ax - b$$
 where $x \in \arg\min_{z} \left[f(z) + u^T Az \right]$

Dual ascent method for maximizing dual objective:

i

$$x_k \in \arg\min_x \left[f(x) + (u_{k-1})^T Ax \right]$$

$$u_k = u_{k-1} + \alpha_k (Ax_k - b)$$

- Step sizes α_k , $k = 1, 2, 3, \ldots$, are chosen in standard ways.
- Proximal gradients and acceleration can be applied as they would usually.

 $f \to \min_{x,y,z}$



The following results hold from combining the last fact with what we already know about gradient descent:¹

- If f is strongly convex with parameter μ , then dual gradient ascent with constant step sizes $\alpha_k = \mu$ converges at sublinear rate $O(\frac{1}{\epsilon})$.
- If f is strongly convex with parameter μ and ∇f is Lipschitz with parameter L, then dual gradient ascent with step sizes $\alpha_k = \frac{2}{\frac{1}{\mu} + \frac{1}{L}}$ converges at linear rate $O(\log(\frac{1}{\epsilon}))$.

Note that this describes convergence in the dual. (Convergence in the primal requires more assumptions)

¹This is ignoring the role of *A*, and thus reflects the case when the singular values of *A* are all close to 1. To be more precise, the step sizes here should be: $\frac{\mu}{\sigma_{\max}(A)^2}$ (first case) and $\frac{2}{\frac{\sigma_{\max}(A)^2}{\mu} + \frac{\sigma_{\min}(A)^2}{L}}$ (second case).

Dual decomposition

Consider

$$\min_{x} \sum_{i=1}^{B} f_i(x_i) \quad \text{subject to} \quad Ax = b$$

Here $x = (x_1, \ldots, x_B) \in \mathbb{R}^n$ divides into B blocks of variables, with each $x_i \in \mathbb{R}^{n_i}$. We can also partition A accordingly:

$$A = [A_1 \dots A_B], \text{ where } A_i \in \mathbb{R}^{m \times n}$$

Simple but powerful observation, in calculation of subgradient, is that the minimization decomposes into B separate problems:

$$\begin{aligned} x^{\mathsf{new}} &\in \arg\min_{x} \left(\sum_{i=1}^{B} f_i(x_i) + u^T A x \right) \\ \Rightarrow x_i^{\mathsf{new}} &\in \arg\min_{x_i} \left(f_i(x_i) + u^T A_i x_i \right), \quad i = 1, \dots, B \end{aligned}$$



Dual decomposition

Consider

$$\min_{x} \sum_{i=1}^{B} f_i(x_i) \quad \text{subject to} \quad Ax = b$$

Here $x = (x_1, \ldots, x_B) \in \mathbb{R}^n$ divides into B blocks of variables, with each $x_i \in \mathbb{R}^{n_i}$. We can also partition A accordingly:

$$A = [A_1 \dots A_B], \text{ where } A_i \in \mathbb{R}^{m \times n}$$

Simple but powerful observation, in calculation of subgradient, is that the minimization decomposes into B separate problems:

$$x^{\mathsf{new}} \in \arg\min_{x} \left(\sum_{i=1}^{B} f_i(x_i) + u^T A x \right)$$

$$\Rightarrow x_i^{\mathsf{new}} \in \arg\min_{x_i} \left(f_i(x_i) + u^T A_i x_i \right), \quad i = 1, \dots, B$$

- **Broadcast:** Send *u* to each of the *B* processors, each optimizes in parallel to find *x_i*.
- **Gather:** Collect $A_i x_i$ from each processor, update the global dual variable u.

$$x_{i}^{k} \in \arg\min_{x_{i}} \left(f_{i}(x_{i}) + (u^{k-1})^{T} A_{i} x_{i} \right), \quad i = 1, \dots, B$$

$$u_{i}^{k} = u_{i}^{k-1} + \alpha_{k} \left(A_{i} x_{i}^{k} - b_{i} \right), \quad i = 1, \dots, B$$

Inequality constraints

Consider the optimization problem:

$$\min_{x} \sum_{i=1}^{B} f_i(x_i)$$
 subject to $\sum_{i=1}^{B} A_i x_i \leq b$



Inequality constraints

Consider the optimization problem:

$$\min_x \sum_{i=1}^B f_i(x_i) \quad \text{subject to} \quad \sum_{i=1}^B A_i x_i \leq b$$

Using dual decomposition, specifically the projected subgradient method, the iterative steps can be expressed as:

• The primal update step:

$$x_i^k \in \arg\min_{x_i} \left[f_i(x_i) + \left(u^{k-1} \right)^T A_i x_i \right], \quad i = 1, \dots, B$$

• The dual update step:

$$u^{k} = \left(u^{k-1} + \alpha_{k}\left(\sum_{i=1}^{B} A_{i}x_{i}^{k} - b\right)\right)_{+}$$

where $(u)_+$ denotes the positive part of u, i.e., $(u_+)_i = \max\{0, u_i\}$, for $i = 1, \ldots, m$.

Augmented Lagrangian method aka method of multipliers

Dual ascent disadvantage: convergence requires strong conditions. Augmented Lagrangian method transforms the primal problem:

$$\min_{x} f(x) + \frac{\rho}{2} \|Ax - b\|^{2}$$

s.t. $Ax = b$

Augmented Lagrangian method aka method of multipliers

Dual ascent disadvantage: convergence requires strong conditions. Augmented Lagrangian method transforms the primal problem:

$$\min_{x} f(x) + \frac{\rho}{2} \|Ax - b\|^{2}$$

s.t. $Ax = b$

where $\rho > 0$ is a parameter. This formulation is clearly equivalent to the original problem. The problem is strongly convex if matrix A has full column rank.

Augmented Lagrangian method aka method of multipliers

Dual ascent disadvantage: convergence requires strong conditions. Augmented Lagrangian method transforms the primal problem:

$$\min_{x} f(x) + \frac{\rho}{2} \|Ax - b\|^{2}$$

s.t. $Ax = b$

where $\rho > 0$ is a parameter. This formulation is clearly equivalent to the original problem. The problem is strongly convex if matrix A has full column rank.

Dual gradient ascent: The iterative updates are given by:

$$x_{k} = \arg\min_{x} \left[f(x) + (u_{k-1})^{T} A x + \frac{\rho}{2} \|Ax - b\|^{2} \right]$$
$$u_{k} = u_{k-1} + \rho(Ax_{k} - b)$$

- Advantage: The augmented Lagrangian gives better convergence.
- Disadvantage: We lose decomposability! (Separability is ruined)
- Notice step size choice $\alpha_k = \rho$ in dual algorithm.

Colab Example

• Dual subgradient and Augmented Lagrangian methods Comparison & Open in Colab.



Alternating direction method of multipliers or ADMM aims for the best of both worlds. Consider the following optimization problem:

Minimize the function:

$$\min_{x,z} f(x) + g(z)$$

s.t. $Ax + Bz = c$

Alternating direction method of multipliers or ADMM aims for the best of both worlds. Consider the following optimization problem:

Minimize the function:

$$\min_{x,z} f(x) + g(z)$$

s.t. $Ax + Bz = c$

We augment the objective to include a penalty term for constraint violation:

$$\min_{x,z} f(x) + g(z) + \frac{\rho}{2} \|Ax + Bz - c\|^2$$

s.t. $Ax + Bz = c$



Alternating direction method of multipliers or ADMM aims for the best of both worlds. Consider the following optimization problem:

Minimize the function:

$$\min_{x,z} f(x) + g(z)$$

s.t. $Ax + Bz = c$

We augment the objective to include a penalty term for constraint violation:

$$\min_{x,z} f(x) + g(z) + \frac{\rho}{2} \|Ax + Bz - c\|^2$$

s.t. $Ax + Bz = c$

where $\rho > 0$ is a parameter. The augmented Lagrangian for this problem is defined as:

$$L_{\rho}(x, z, u) = f(x) + g(z) + u^{T}(Ax + Bz - c) + \frac{\rho}{2} ||Ax + Bz - c||^{2}$$



ADMM repeats the following steps, for $k = 1, 2, 3, \ldots$:

1. Update x:

$$x_k = \arg\min_x L_\rho(x, z_{k-1}, u_{k-1})$$

2. Update z:

$$z_k = \arg\min_z L_\rho(x_k, z, u_{k-1})$$

3. Update *u*:

$$u_k = u_{k-1} + \rho(Ax_k + Bz_k - c)$$

ADMM repeats the following steps, for $k = 1, 2, 3, \ldots$:

1. Update x:

$$x_k = \arg\min_x L_\rho(x, z_{k-1}, u_{k-1})$$

2. Update z:

$$z_k = \arg\min_z L_\rho(x_k, z, u_{k-1})$$

3. Update *u*:

$$u_k = u_{k-1} + \rho(Ax_k + Bz_k - c)$$

Note: The usual method of multipliers would replace the first two steps by a joint minimization:

$$(x^{(k)}, z^{(k)}) = \arg\min_{x,z} L_{\rho}(x, z, u^{(k-1)})$$



ADMM Summary

- ADMM is one of the key and popular recent optimization methods.
- It is implemented in many solvers and is often used as a default method.
- The non-standard formulation of the problem itself, for which ADMM is invented, turns out to include many important special cases. "Unusual" variable y often plays the role of an auxiliary variable.
- Here the penalty is an additional modification to stabilize and accelerate convergence. It is not necessary to make ρ very large.