An Introduction to Alternating Direction Method of Multipliers (ADMM)

Yu Ding

The University of Texas MD Anderson Cancer Center

October 24th, 2024

Reference

Foundations and Trends® in
Machine Learning
Vol. 3, No. 1 (2010) 1–122
© 2011 S. Boyd, N. Parikh, E. Chu, B. Peleato
and J. Eckstein
DOI: 10.1561/2200000016

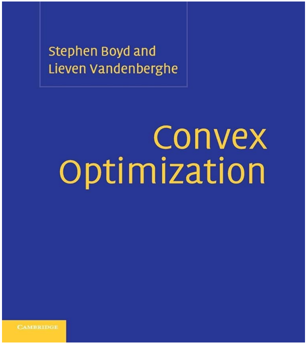
the essence of knowledge

Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers

Stephen Boyd[1], Neal Parikh[2], Eric Chu[3]
Borja Peleato[4] and Jonathan Eckstein[5]

[1] *Electrical Engineering Department, Stanford University, Stanford, CA 94305, USA, boyd@stanford.edu*
[2] *Computer Science Department, Stanford University, Stanford, CA 94305, USA, npparikh@cs.stanford.edu*
[3] *Electrical Engineering Department, Stanford University, Stanford, CA 94305, USA, echu508@stanford.edu*
[4] *Electrical Engineering Department, Stanford University, Stanford, CA 94305, USA, peleato@stanford.edu*
[5] *Management Science and Information Systems Department and RUTCOR, Rutgers University, Piscataway, NJ 08854, USA, jeckstei@rci.rutgers.edu*

[1]



[2]

Outline

Outline

A brief introduction to Lagrangian Duality

Primary Problem

$$
\begin{aligned}
&\min_{x} \ f(x) \\
&\text{s.t.} h_i(x) \leq 0, i \in \{1, \cdots, I\} \\
&\qquad \ell_j(x) = 0, j \in \{1, \cdots, J\}
\end{aligned}
$$

- The objective function $f(x)$ usually has poor properties (non-convex or no Lipschitz continuity).
- The feasible region is not always a convex set.

Lagrangian of the Primary Problem

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{I} \lambda_i h_i(x) + \sum_{j=1}^{J} \nu_j \ell_j(x),$$

where λ, ν are Lagrangian multipliers.

Dual function

$$g(\lambda, \nu) = \inf_{x \in \mathbf{dom} f} (f(x) + \sum_{i=1}^{I} \lambda_i h_i(x) + \sum_{j=1}^{J} \nu_j \ell_j(x)).$$

Note that, $g(\lambda, \nu)$ is concave w.r.t λ, ν, and the above infimum is taken over the domain of f.

Primary Problem

$$\min_{x} f(x)$$

$$\text{s.t.} h_i(x) \le 0, i \in \{1, \cdots, I\}$$

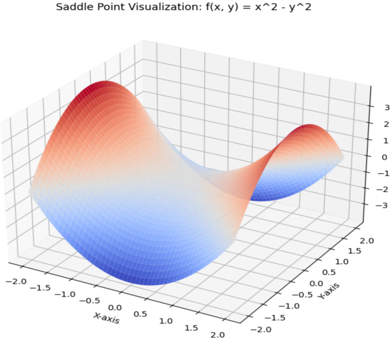
$$\ell_j(x) = 0, j \in \{1, \cdots, J\}$$

Dual Problem

$$\max_{\lambda,\nu} g(\lambda, \nu)$$

$$\text{s.t.} \lambda \ge 0.$$

The dual problem is much easier to solve, as it is a convex optimization with linear constraints.



Outline

Dual Asent

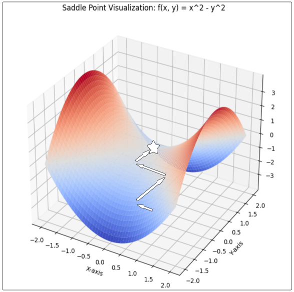
Consider an easier problem $\min_x f(x)$ s.t. $Ax = b$.
Its dual function is

$$g(y) = \inf_{x \in \textbf{dom} f} f(x) + y^T(Ax - b).$$

The gradient of $g(y)$ is $\frac{\partial g}{\partial y} = Ax^* - b$

The dual asent algorithm is

- Initialize dual guess $y^{(0)}$
- repeat for $k = 1, 2, 3, \cdots$
  - $x^{(k)} = \arg\min_{x \in \textbf{dom} f} f(x) + (y^{(k-1)})^T Ax$
  - $y^{(k)} = y^{(k-1)} + t_k(Ax^{(k)} - b)$



Augmented Lagrangian method
a.k.a method of multipliers

The disadvantage of dual ascent is that it requires strong conditions to ensure convergence, i.e., convexity and Lipschitz continuity, etc. (It can be considered similar to the gradient descent algorithm.)

We propose a new primal problem

$$
\begin{aligned}
&\min_x \; f(x) + \frac{\rho}{2}||Ax - b||_2^2 \\
&\text{s.t. } Ax = b
\end{aligned}
$$

The benefit that comes with the quadratic penalty term is that we can always adjust ρ such that the objective function is convex (under mild assumptions), as long as matrix A has full column rank.

Augmented Lagrangian method

$$\min_x \; f(x) + \frac{\rho}{2}||Ax - b||_2^2$$
$$\text{s.t. } Ax = b$$

- Initialize dual guess $y^{(0)}$
- repeat for $k = 1, 2, 3, \cdots$
  - $x^{(k)} = \arg\min_{x \in \mathbf{dom} f} f(x) + (y^{(k-1)})^T Ax + \frac{\rho}{2}||Ax - b||_2^2$
  - $y^{(k)} = y^{(k-1)} + t_k(Ax^{(k)} - b)$

Alternating Direction Method of Multipliers (ADMM)

Consider the problem

$$\min_{x,z} \; f(x) + g(z)$$
$$\text{s.t. } Ax + Bz = c$$

The augmented Lagrangian is

$$L_\rho(x, z, u) = f(x) + g(z) + u^T(Ax + Bz - c) + \frac{\rho}{2}||Ax + Bz - c||_2^2$$

The algorithm is

- repeat for $k = 1, 2, 3, \cdots$
  - $x^{(k)} = \arg\min_x L_\rho(x, z^{(k-1)}, u^{(k-1)})$
  - $z^{(k)} = \arg\min_z L_\rho(x^{(k)}, z, u^{(k-1)})$
  - $u^{(k)} = u^{(k-1)} + t_k(Ax^{(k)} + Bz^{(k)} - c)$

Outline

Divide and Conquer

It can split a large problem into a series of subproblems. Usually, each subproblem has a closed form solution. Consider problem

$$\min_x f(x) + g(Ax)$$

We can transform it into

$$\min_{x,z} f(x) + g(z), \text{ s.t. } Ax - z = 0.$$

$$L_\rho(x, z, u) = f(x) + g(z) + u^T(Ax + Bz - c) + \frac{\rho}{2}||Ax + Bz - c||_2^2$$

- repeat for $k = 1, 2, 3, \cdots$
  - $x^{(k)} = \arg\min_x g(z) + (u^{(k-1)})^T Ax + \frac{\rho}{2}||Ax + Bz^{(k-1)} - c||_2^2$
  - $z^{(k)} = \arg\min_z f(x) + (u^{(k-1)})^T Bz + \frac{\rho}{2}||Ax^{(k)} + Bz - c||_2^2$
  - $u^{(k)} = u^{(k-1)} + t_k(Ax^{(k)} + Bz^{(k)} - c)$

Distributed Optimization

Given $y \in \mathbb{R}^n$, $x \in \mathbb{R}^{n \times p}$, we have the group lasso problem as

$$\min_{\beta} \frac{1}{2}||y - X\beta||_2^2 + \lambda \sum_{g=1}^{G} c_g ||\beta_g||_2$$

Rewrite as

$$\min_{\alpha,\beta} \frac{1}{2}||y - X\beta||_2^2 + \lambda \sum_{g=1}^{G} c_g ||\alpha_g||_2, \text{ s.t. } \beta - \alpha = 0.$$

ADMM steps:

- repeat for $k = 1, 2, 3, \cdots$
  - $\beta^{(k)} = (X^T X + \rho I)^{-1}(X^T y + \rho(\alpha^{(k-1)} - \omega^{(k-1)}))$
  - for $g = 1, \cdots, G$ do in parallel
  - $\alpha_g^{(k)} = R_{c_g \lambda/\rho}(\beta^{(k)} + \omega_g^{(k-1)})$
  - $\omega^{(k)} = \omega^{(k-1)} + \beta^{(k)} - \alpha^{(k)}$

Outline

ADMM in CliPP

The problem, eq.(8), in CliPP is

$$\min_{\boldsymbol{\omega}} \; -\ell(\boldsymbol{\omega}) + \sum_{1\le i<j\le s} p_\lambda(|\omega_i - \omega_j|).$$

Define $\eta_{ij} = \omega_i - \omega_j$, we can rewrite it as

$$\min_{\boldsymbol{\omega},\boldsymbol{\eta}} \; -\ell(\boldsymbol{\omega}) + \sum_{1\le i<j\le s} p_\lambda(|\eta_{ij}|).$$

Then the augmented Lagrangian is

$$\begin{aligned} L(\boldsymbol{\omega},\boldsymbol{\eta},\boldsymbol{\tau},\lambda) = & -\ell(\boldsymbol{\omega}) + \sum_{1\le i<j\le s} p_\lambda(|\eta_{ij}|) + \frac{\alpha}{2}\sum_{1\le i<j\le s}(\eta_{ij} - \omega_i - \omega_j)^2 \\ & - \sum_{1\le i<j\le s} \tau_{ij}(\eta_{ij} - \omega_i - \omega_j). \end{aligned}$$

ADMM in CliPP

- repeat for $k = 1, 2, 3, \cdots$
  - eq.(S3) $\boldsymbol{\omega}^{(k)} = (\boldsymbol{B}^T\boldsymbol{B} + \alpha\boldsymbol{\Delta}^T\boldsymbol{\Delta})^{-1}[\alpha\boldsymbol{\Delta}^T(\boldsymbol{\eta}^{(k-1)} - \boldsymbol{\tau}^{(k-1)}) - \boldsymbol{B}^T\boldsymbol{A}]$
  - eq.(S4) $\eta_{ij}^{(k)} = \arg\min_{\eta_{ij}} \frac{\alpha}{2}(\delta_{ij} - \eta_{ij})^2 + p_\lambda(|\eta_{ij}|)$
  - eq.(S5) $\boldsymbol{\tau}^{(k)} = \boldsymbol{\tau}^{(k-1)} - \alpha(\boldsymbol{\Delta}\boldsymbol{\omega}^{(k)} - \boldsymbol{\eta}^{(k)})$

ADMM in CliPP

https://github.com/wwylab/CliPP/blob/master/src/kernel.cpp

- Update ω
- Update η
- Update τ

```cpp
linear = DELTA * (alpha * eta_old + tau_new) - B.cwiseProduct(A);

Minv = 1.0 / ((B.cwiseProduct(B)).array() + double(No_mutation) * alpha);
Minv_diag = Minv.asDiagonal();

trace_g = -alpha * Minv.sum();
if(isnan(trace_g)){
    std::cout << "Lambda: " << Lambda << "\titeration: " << k << "\tNan" << std::endl;
    return -1;
}


Minv_outer = Minv * (Minv.transpose());

inverted = Minv_diag.array() - 1.0 / (1.0 + trace_g) * (-alpha) * Minv_outer.array();
w_new = inverted * linear;
```

```cpp
eta_new(i, 0) = temp * tag1 + ST(temp, Lambda / alpha) * tag2 + ST(temp, gamma * Lambda / ((gamma - 1.0) * alpha)) / (1.0 - 1.0 / ((gamma - 1.0) * alpha)) * tag3 *tag4;

tau_new(i, 0) = tau_old(i, 0) - alpha * (w_new(ids(i, 0), 0) - w_new(ids(i, 1), 0) - eta_new(i, 0));
```

Thank you!

Questions?

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al.
Distributed optimization and statistical learning via the alternating direction method of multipliers.
Foundations and Trends® in Machine learning, 3(1):1–122, 2011.

Stephen Boyd and Lieven Vandenberghe.
Convex optimization.
Cambridge university press, 2004.