The Alternating Direction Method of Multipliers

With Adaptive Step Size Selection

Peter Sutor, Jr.

Project Advisor: Professor Tom Goldstein

October 8, 2015

Presentation Outline

The General Problem

- Convex Optimization:
  - We wish to find an optimal $x^* \in X$ such that:

    $$f(x^*) = \min \{f(x) : x \in X\},$$

    where $X \subset \mathbb{R}^n$ is called the *feasible set* and $f(x) : \mathbb{R}^n \longmapsto \mathbb{R}$ is the *objective function*.
  - Objective function f is convex on $\mathbb{R}^n$.
  - Feasible set X is a closed convex set.
- Large scale optimization:
  - Huge data-sets.
  - Traditional techniques for minimization may be too slow.
- Decentralized optimization.

How can ADMM help?

- Need robust methods for:
  - Arbitrary scale optimization.
  - Decentralized optimization.
- The Alternating Direction Method of Multipliers (ADMM):
  - Solves convex optimization problems by splitting them into smaller, easier to handle pieces.
  - Can solve these pieces in parallel.
  - Is robust, and handles the forms of optimization we want.

The Dual Problem

- Consider the following problem (*primal problem*):
  $$\min_x(f(x)) \text{ subject to } Ax = b.$$
- Important components of this problem:
  1. The Lagrangian: $L(x, y) = f(x) + y^T(Ax - b)$
     - We refer to the original x variable as the *primal variable* and the y variable as the *dual variable*.
  2. Dual function: $g(y) = \inf_x(L(x, y))$
     - New function made purely out of the dual variable.
     - Gives a lower bound on the objective value.
  3. Dual problem: $\max_{y \geq 0}(g(y))$
     - The problem of finding the best lower bound.
- End goal: recover $x^* = \arg\min_x(L(x, y^*))$, where x^* and y^* are corresponding optimizers.

The Dual Ascent Method (DAM)

- DAM is a gradient-type method that solves our dual problem; characterized by the k-iteration:

$$y^{(k+1)} = y^{(k)} + \alpha^{(k)} \nabla g(y^{(k)}),$$

  where $\alpha^{(k)}$ is a step size for the iteration k.
- Note that $\nabla g(y^{(k)}) = Ax^* - b$ and $x^* = \arg\min_x(L(x, y^{(k)}))$.
- Repeat for $k = 0$ to a given n number of steps, or until convergence:
  1. $x^{(k+1)} := \arg\min_x(L(x, y^{(k)}))$
  2. $y^{(k+1)} := y^{(k)} + \alpha^{(k)}(Ax^{(k+1)} - b)$

Dual Decomposition (DD)

- Let's say that our objective is *separable*; then:

$$f(x) = f_1(x_1) + \cdots + f_m(x_m), x = (x_1, ..., x_m)$$

- The same goes for the Lagrangian:

$$L(x, y) = L_1(x_1, y) + \cdots + L_m(x_m, y) - y^T b,$$

where $L_i = f(x_i) + y^T A_i x_i$.

- Thus, our x-minimization step in the DAM is split into m separate minimizations that can be carried out in parallel:

$$x_i^{(k+1)} := \arg\min_{x_i}(L_i(x_i, y^{(k)}))$$

Dual Decomposition (continued)

- Idea: decompose $y^{(k)}$, update x_i in parallel then add up the $A_i x_i^{(k+1)}$ terms.
- DD as proposed by Everett, Dantzig, Wolfe, and Benders:

  Repeat for $k = 0$ to a given n steps, or until convergence:
  1. $x_i^{(k+1)} := \arg\min_{x_i}(L_i(x_i, y^{(k)}))$, for $i = 1, \ldots, m$
  2. $y^{(k+1)} := y^{(k)} + \alpha^{(k)} \left(\sum_{i=1}^{m} A_i x_i^{(k+1)} - b \right)$
- Solve large problem by solving parallel sub-problems, coordinating at the dual variable.
- Drawbacks:
  - Needs assumption that f is separable.
  - Can be slow at times.

Method of Multipliers (MM)

- Need a more robust DAM? Use the Method of Multipliers.
- Swap the Lagrangian for an Augmented Lagrangian:

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + (\rho/2)||Ax - b||_2^2, \quad \rho > 0$$

- Method of Multipliers as proposed by Hestenes and Powell:

  Repeat for $k = 0$ to a given n, or until convergence:

  1. $x^{(k+1)} := \arg\min_x(L_\rho(x, y^{(k)}))$
  2. $y^{(k+1)} := y^{(k)} + \rho(Ax^{(k+1)} - b)$

  The ρ here is the dual update step length.

MM: The Dual Update Step

- If f is differentiable, the optimality conditions are:

Primal Feasibility:	$Ax^* - b = 0$
Dual Feasibility:	$\nabla f(x^*) + A^T y^* = 0$

- At each iteration k, $x^{(k+1)}$ minimizes $L_\rho(x, y^{(k)})$, so:

$$\begin{aligned} \nabla_x L_\rho(x^{(k+1)}, y^{(k)}) &= \nabla_x(f(x^{(k+1)})) + A^T(y^{(k)} + \rho(Ax^{(k+1)} - b)) \\ &= \nabla_x(f(x^{(k+1)})) + A^T y^{(k+1)} = 0 \end{aligned}$$

- Thus, our dual update $y^{(k+1)}$ makes $(x^{(k+1)}, y^{(k+1)})$ dual feasible; primal feasibility is achieved as $(Ax^{(k+1)} - b) \to 0$ as $k \to \infty$.

MM: the good, the bad and the ugly

- **The Good:** Conditions for convergence are much more relaxed than DD; e.g., f doesn't have to be differentiable.
- **The Bad:** Quadratic penalty from using the Augmented Lagrangian prevents us from being able to separate the x-update like in DD; thus, we can't use DD with MM.
- **The Ugly:** We can't use DD and MM simultaneously and have the advantages of both methods, at least not with the set-up we have here.

What is ADMM?

- Finds a way to combine advantages of DD and MM.
  - Robustness of the Method of Multipliers.
  - Supports Dual Decomposition $\rightarrow$ parallel x-updates.
- Problem form:

$$\min\left(f(x)+g(z)\right) \text{ subject to } Ax+Bz=c,$$

  where f and g are both convex.
- Objective is separable into two sets of variables.
- ADMM defines a special Augmented Lagrangian to enable decomposition:

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

ADMM Algorithm

- Algorithm proposed by Gabay, Mercier, Glowinski, and Marrocco in:

  Repeat for $k = 0$ to specified n, or until convergence:

  1. $x^{(k+1)} := \arg\min_x (L_\rho(x, z^{(k)}, y^{(k)}))$
  2. $z^{(k+1)} := \arg\min_z (L_\rho(x^{(k+1)}, z, y^{(k)}))$
  3. $y^{(k+1)} := y^{(k)} + \rho(Ax^{(k+1)} + Bz^{(k+1)} - c)$

- Doesn't minimize x and z together, as in MM. It instead solves a linear system of equations for the z-minimization step.
- The Augmented Lagrangian uses the extra penalty term $\frac{\rho}{2}||Ax + Bz - c||_2^2$ to enable this separation.

Optimality Conditions for ADMM

- For the differentiable case, the optimality conditions are:

$$
\begin{array}{ll}
\text{Primal Feasibility:} & Ax + Bz - c = 0 \\
\text{Dual Feasibility:} & \nabla f(x) + A^T y = 0, \\
 & \nabla g(z) + B^T y = 0
\end{array}
$$

- As $z^{(k+1)}$ minimizes $L_\rho(x^{(k+1)}, z, y^{(k)})$, it follows that:

$$
\begin{aligned}
0 &= \nabla g(z^{(k+1)}) + B^T y^{(k)} + \rho B^T (Ax^{(k+1)} + Bz^{(k+1)} - c) \\
&= \nabla g(z^{(k+1)}) + B^T y^{(k+1)}
\end{aligned}
$$

- Dual update makes $(x^{(k+1)}, z^{(k+1)}, y^{(k+1)})$ satisfy the second dual feasible condition.
- Other conditions are achieved as $k \to \infty$.

Convergence of ADMM

- Assumptions required:
  1. Functions f and g are closed, convex and proper.
     - A function is *closed* if for any $\alpha \in \mathbb{R}$, $x \in dom(f) : f(x) \leq \alpha$ is a *closed set*.
     - A convex function is *proper* if $f(x) < \infty$ for some x and $f(x) > -\infty$ for every x.
  2. For $\rho = 0$, L_ρ has a saddle point.

- If assumptions are true, then ADMM converges:
  - Iterates approach feasibility.
  - Objective function approaches optimal value.

Scaling Dual Variables

- Let $r = Ax + Bz - c$, then:

$$\begin{aligned} L_\rho(x, z, y) &= f(x) + g(z) + y^T r + (\rho/2)||r||_2^2 \\ &= f(x) + g(z) + (\rho/2)||r + (1/\rho)y||_2^2 - (1/2\rho)||y||_2^2 \\ &= f(x) + g(z) + (\rho/2)||r + u||_2^2 - constant_y \\ &= L_\rho(x, z, u), \end{aligned}$$

  where $u = (1/\rho)y$.

- Now the algorithm is:
  1. $x^{(k+1)} := \arg\min_x(L_\rho(x, z^{(k)}, u^{(k)}))$
  2. $z^{(k+1)} := \arg\min_z(L_\rho(x^{(k+1)}, z, u^{(k)}))$
  3. $u^{(k+1)} := u^{(k)} + (Ax^{(k+1)} + Bz^{(k+1)} - c)$

Writing problems in ADMM form

- Generic Problem: $\min(f(x))$, subject to $x \in \mathbb{S}$
- ADMM Form: $\min(f(x) + g(z))$, subject to $x - z = 0$, where $g(z) = \mathbb{I}_{\mathbb{S}}(z)$, the indicator function that z is in $\mathbb{S}$.
- Notice that $B = -I$, so z-minimization boils down to:

$$\arg\min(g(z) + (\rho/2)|| - z - v||_2^2) = \mathbf{prox}_{g,\rho}(v),$$

with $v = x^{(k+1)} + u^{(k)}$ (*Proximal Function*).
- Since $g(z)$ is the indicator function, do this by projecting v onto $\mathbb{S}$. Use soft-thresholding:

$$z_i^{(k+1)} := (v_i - \lambda/\rho)_+ - (-v_i - \lambda/\rho)_+$$

Common problems solved by ADMM

- Basis Pursuit
- Sparse Inverse Covariance Selection
- Huber Fitting
- Intersection of Polyhedra
- Lasso Problem
- Least Absolute Deviations
- Linear Programming
- ℓ_1 Regularized Logistic Regression
- Regressor Selection (nonconvex)
- Quadratic Programming
- Support Vector Machines (SVMs)
- Total Variation Minimization (e.g., image denoising)

Total Variation Minimization (1-D Case)

- In essence, total variation is an infinitesimal version of absolute value; definable for x in one dimension as $V(x) = \sum_i |x_{i+1} - x_i|$.
- In one dimension, problem is of the following form:

$$E(x, b) + \lambda V(x) = \min_x \frac{1}{2}||x - b||_2^2 + \lambda \sum_i |x_{i+1} - x_i|$$

where $x, b \in \mathbb{R}^n$.

- Let's write the problem in ADMM form:

$$\min_x \frac{1}{2}||x - b||_2^2 + \lambda \sum_i |x_{i+1} - x_i| + g(z)$$

subject to $dx - z = 0$, where $x, b \in \mathbb{R}^n$ and $g(z)$ is the indicator function as mentioned before.

Total Variation Minimization (continued)

- What is the x-minimization step?
- Using Augmented Lagrangian $L_\rho(x, z, u)$, the problem is minimizing for x:

$$\frac{1}{2}||x-b||_2^2+\lambda\sum_i |x_{i+1} - x_i|+g(z)+\frac{\rho}{2}||dx-z+u||_2^2-constant_y$$

- So set gradient in x to zero to minimize:

$$\nabla_x L_\rho(x, z, u) = x - b + \rho d^T(dx - z + u) = 0$$

- Group x terms on one side:

$$(I + \rho d^T d)x = \rho d^T(z - u) + b$$

- So: $x = (I + \rho d^T d)^{-1}(\rho d^T(z - u) + b)$

Total Variation Minimization (continued)

ADMM Algorithm to Solve T.V. Problem

1. Form difference matrix D approximating d in dx using stencil $[1, -1]$ along diagonal; circular wrapping.
2. Solve for $x^{(k+1)}$ (this is the x-update) the system:

$$(I + \rho D^T D)x^{(k+1)} = b + \rho D^T (z^{(k)} - y^{(k)})$$

3. The "*shrinkage*" *of* $P = Dx^{(k+1)} + y^{(k)}$ *is the z-update:*

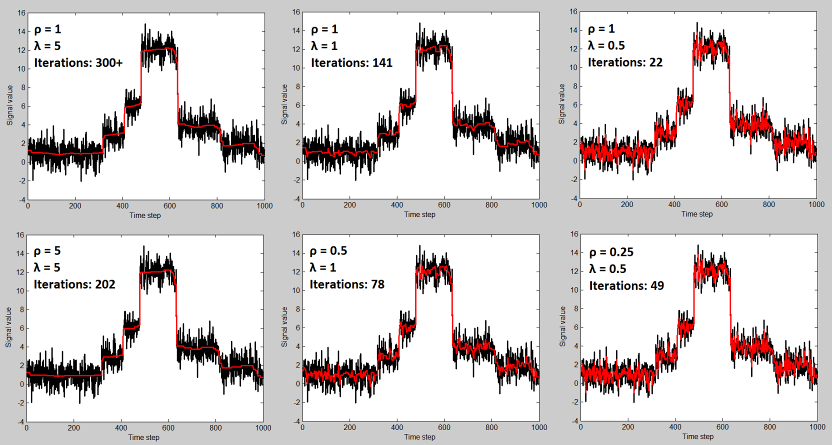
$$z^{(k+1)} := \max\left(0, P - \frac{\lambda}{\rho}\right) - \max\left(0, -P - \frac{\lambda}{\rho}\right)$$

4. *Finally, the dual update:*

$$y^{(k+1)} := y^{(k)} + Dx^{(k+1)} - z^{(k+1)}$$

5. *Repeat steps 2-4 until convergence or n iterations is reached.*

Examples of denoising in one dimension



In this project...

- Our goal is to make ADMM easier to use in practice: upload A, B, and c, then run appropriate function.
- Maximizing ADMM's potential means tweaking parameters such as step size ρ and more.
- Hope to create a comprehensive library for general ADMM use.
  - Generalized ADMM functionality.
  - Adaptive step-size selection.
  - Ready to go optimized functions for problems ADMM is most used for.
  - High performance computing capabilities (MPI).
  - Implementations in Python and Matlab.

Major Obstacles

1. There is no known step-size selection algorithm for ADMM:
   - In practice, choosing of ρ is typically done by fine-tuning and testing.
   - Optimal ρ changes with the problem, and perhaps even data.
   - It may be possible to dynamically choose optimal ρ at every iteration instead.
2. How to dynamically choose ρ?
   - Several possible strategies we will try.
   - Requires thorough testing to see which works and which works best.
3. When should the algorithm stop? May require multiple types of stopping conditions.
4. How to validate that given input will actually converge?

Stopping Conditions

- Primal (p) and Dual (d) residuals in ADMM at step $k+1$:
    - $p^{k+1} = Ax^{k+1} + Bz^{k+1} - c$
    - $d^{k+1} = \rho A^T B(z^{k+1} - z^k)$
- Reasonable stopping criteria: $||p^k||_2 \leq \epsilon^{pri}$ and $||d^k||_2 \leq \epsilon^{dual}$.
- Many ways to choose these tolerances.
- One common example, where $p \in \mathbb{R}^{n_1}$ and $d \in \mathbb{R}^{n_2}$:
    - $\epsilon^{pri} = \sqrt{n_1}\epsilon^{abs} + \epsilon^{rel} \max(||Ax^k||_2, ||Bz^k||_2, ||c||_2)$
    - $\epsilon^{dual} = \sqrt{n_2}\epsilon^{abs} + \epsilon^{rel}||A^T y^k||_2$

  where ϵ^{abs} and ϵ^{rel} are chosen constants referred to as *absolute* and *relative* tolerance.

Adaptive Step Size Selection Strategies

1. Perform a partial step with a pre-existing estimate for step-size. Perform linear interpolation of the residuals with unknown full step-size. Solve this to find what a better step size should have been. Use this as next step-size.
2. It is possible to optimize the dual problem's step size. Can keep the penalty term's ρ constant and manipulate the dual's step size only.
3. ADMM can be viewed as a type of Douglas-Rachford Method. Using results from Esser's paper on DRM, can solve for optimal step size via finding the gradient on the dual step in DRM.

Testing and Validation

- **Testing:**
  - Main functionality can be tested through randomized data: random A, B and c. This is standard for convex optimization.
  - For specific problems, e.g. SVM classifiers, can compare to existing solvers and datasets. For example: the MNIST handwritten data.
- **Validation:**
  - Can compare performances between adaptive step size selection strategies.
  - Can also compare these strategies to normal ADMM performance without adaptive step-size selection.

Project Timeline

- **Fall Semester Goals:**
  - **End of October:** Implement generic ADMM, solvers for the Lasso problem, TV Minimization, and SVMs.
  - **Early November:** Implement scripts for general testing, convergence checking, and stopping condition strategies.
  - **End of November:** Try out implementations of all three adaptive step-size selection strategies.
  - **Early December:** Finalize bells and whistles on ADMM options. Compile testing/validation data.
- **Spring Semester Goals:**
  - **End of February:** Implement the full library of standard problem solvers.
  - **End of March:** Finish implementing MPI in ADMM library.
  - **End of April:** Finishing porting code to Python version.
  - **Early May:** Compile new testing/validation data.

Deliverables

- **ADMM Library: Python and Matlab versions**
  - Contain general ADMM with adaptive step size routines and standard solvers for common problems ADMM solves.
  - Scripts for generating random test data and results.
  - Scripts for validating performance of adaptive ADMM to regular ADMM for each adaptive strategy.
- Report on observed testing/validation results and on findings with adaptive ADMM - may lead to a paper eventually.
- Datasets used for testing the standard solvers (or references to where to obtain them, if they are too big).

References

- S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers", *Foundations and Trends in Machine Learning*, vol. 3, no.1, pp. 1-122, 2010.
- Ernie Esser, *Applications of Lagrangian-Based Alternating Direction Methods and Connections to Split Bregman*, April 2009
- H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, no. 3, pp. 399-417, 1963.
- M. R. Hestenes, "Multiplier and gradient methods," *Journal of Optimization Theory and Applications*, vol. 4, pp. 302-320, 1969.
- J. Eckstein and M. Fukushima, "Some reformulations and applications of the alternating direction method of multipliers," *Large Scale Optimization: State of the Art*, pp. 119-138, 1993.