

Math 6880/7875: Advanced Optimization Alternating Methods, I

Akil Narayan¹

¹Department of Mathematics, and Scientific Computing and Imaging (SCI) Institute
University of Utah

March 1, 2022



Alternation

Alternating methods solve an optimization problem by cycling through certain optimization sub-problems.

One can alternate in terms of

- Objective components/sub-components
- Constraint sets
- Variable components
- Data (e.g., SGD)

Our tour will take us through:

- Coordinate descent
- Bregman methods
- Alternating direction method of multipliers
- Alternating projections
- Proximal methods
- Majorize-minimization/Minorize-maximization

Alternation

Alternating methods solve an optimization problem by cycling through certain optimization sub-problems.

One can alternate in terms of

- Objective components/sub-components
- Constraint sets
- Variable components
- Data (e.g., SGD)

Our tour will take us through:

- Coordinate descent
- Bregman methods
- Alternating direction method of multipliers
- Alternating projections
- Proximal methods
- Majorize-minimization/Minorize-maximization

A starting point: linear systems

We first consider solving a square, linear system:

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}$$

One historical approach to solving such a potentially large system without direct inversion is the Gauss-Seidel method.

First decompose A into its lower-triangular and *strictly* upper-triangular components:

$$A = L + U, \quad L = \begin{pmatrix} X & & & \\ X & X & & \\ \vdots & \vdots & \ddots & \\ X & X & \cdots & X \end{pmatrix}, \quad U = \begin{pmatrix} & X & \cdots & X \\ & & \ddots & \\ & & & \\ & & & X \end{pmatrix}$$

Gauss-Seidel

$$A = L + U$$

$$Ax = b \rightarrow Lx = b - Ux$$

The Gauss-Seidel method is an iterative approach. Let $x^{(k)}$ denote the k th iterate. Gauss-Seidel updates via,

$$x^{(k+1)} = L^{-1}(b - Ux^{(k)}).$$

The process is repeated, and convergence is understood. (E.g., if A is diagonally dominant.)

What is Gauss-Seidel doing? Note row j of this equation takes the form,

$$a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n = b_n.$$

The Gauss-Seidel version of this equation is,

$$a_{j1}x_1^{(k+1)} + a_{j2}x_2^{(k+1)} + \cdots + a_{jj}x_j^{(k+1)} + a_{j,j+1}x_{j+1}^{(k)} + \cdots + a_{jn}x_n^{(k)} = b_n.$$

Solving these for all j can be accomplished via forward substitution:

- Solve row 1 for x_1 (x_j , $j \geq 2$ from previous iteration)
- Solve row 2 for x_2 (x_1 from previous step, x_j $j \geq 3$ from previous iteration)
- Solve row 3 for x_3 (x_1, x_2 from previous step, x_j $j \geq 4$ from previous iteration)
- ...

I.e., this process cycles through equations, where each step is a *one-dimensional* problem by fixing all other variables.

Gauss-Seidel

$$A = L + U$$

The Gauss-Seidel method is an iterative approach. Let $x^{(k)}$ denote the k th iterate. Gauss-Seidel updates via,

$$x^{(k+1)} = L^{-1}(b - Ux^{(k)}).$$

The process is repeated, and convergence is understood. (E.g., if A is diagonally dominant.)

What is Gauss-Seidel doing? Note row j of this equation takes the form,

$$a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n = b_n.$$

The Gauss-Seidel version of this equation is,

$$a_{j1}x_1^{(k+1)} + a_{j2}x_2^{(k+1)} + \cdots + a_{jj}x_j^{(k+1)} + a_{j,j+1}x_j^{(k)} + \cdots + a_{jn}x_n^{(k)} = b_n.$$

Solving these for all j can be accomplished via forward substitution:

- Solve row 1 for x_1 (x_j , $j \geq 2$ from previous iteration)
- Solve row 2 for x_2 (x_1 from previous step, x_j $j \geq 3$ from previous iteration)
- Solve row 3 for x_3 (x_1, x_2 from previous step, x_j $j \geq 4$ from previous iteration)
- ...

I.e., this process cycles through equations, where each step is a *one-dimensional* problem by fixing all other variables.

Coordinate descent

For optimizing scalar functions, the same principle, cycling through individual variables for “lower-level” optimization problems, is the basic idea behind [coordinate descent](#).

$$\min_{x \in S} f(x), \quad f(x) = f((x_1, x_2, \dots, x_n)).$$

Coordinate proceeds by iteratively solving subproblems. Start with an initial iterate $x^{(0)}$, set $k = 0$.

$$x_1^{(k+1)} = \arg \min_{x_1} f(x_1, x_2^{(k)}, \dots, x_n^{(k)})$$

$$x_2^{(k+1)} = \arg \min_{x_2} f(x_1^{(k+1)}, x_2, x_3^{(k)}, \dots, x_n^{(k)})$$

\vdots

$$x_n^{(k+1)} = \arg \min_{x_n} f(x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{n-1}^{(k+1)}, x_n)$$

Then repeat, set $k \leftarrow k + 1$.

Coordinate descent visualized

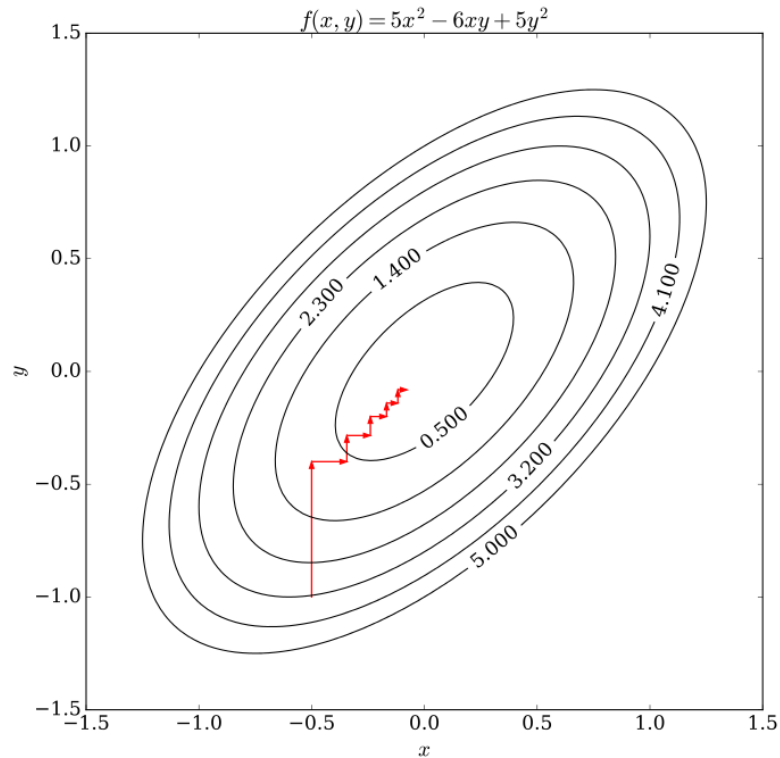


Image: Wikipedia, "Coordinate Descent" page.

Coordinate descent discussion

Each subproblem is generally much easier to solve than the full problem.

There are several variants of coordinate descent:

- We have described the deterministic cyclic approach
- Randomized descent (descent direction uniformly sampled at random each step)
- Block coordinate descent (subproblems over more than 1 component variable)
- Minimization is algorithmically not performed exactly at each step, but instead approximately solved, e.g., by taking a single gradient descent step.

Coordinate descent convergence

Assume randomized coordinate descent

Coordinate descent converges under some assumptions.

Theorem

Assume f is smooth and convex and that f_ is the value of a minimum. Then*

$$\mathbb{E}f(x^{(k)}) - f_* \lesssim \frac{1}{k}.$$

If f is strongly convex (with respect to the Euclidean norm), then

$$\mathbb{E}f(x^{(k)}) - f_* \lesssim r^k, \quad r \in (0, 1).$$

Bad examples of coordinate descent, I

Unfortunately, in general *some* smoothness is required.

Without smoothness, one can easily come up with bad examples.

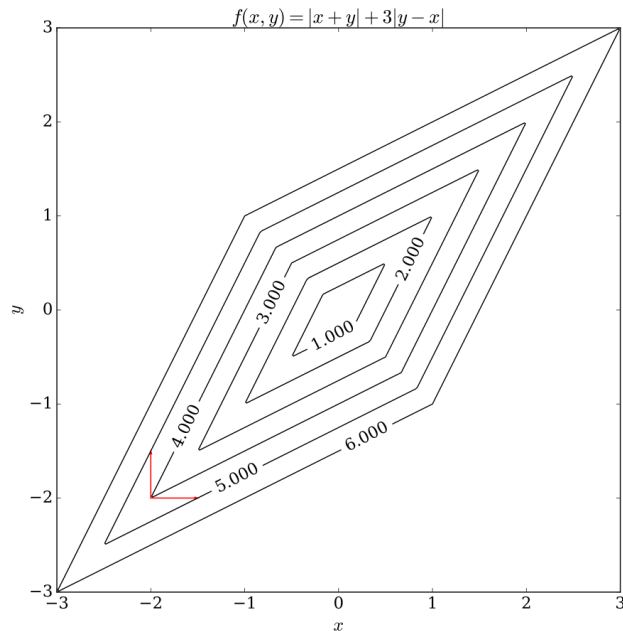


Image: Wikipedia, "Coordinate Descent" page.

Bad examples of coordinate descent, II

In particular: consider a deterministic cyclic coordinate descent.

- If f is smooth and convex and a full cycle through all n variables does not change the iterate, we are at a local minimum (because $\nabla f = 0$).
- If f is *not* smooth but still convex, and again a full cycle through all n variables does not change the iterate, there is no guarantee we are at a local minimum (coordinatewise minimization is not sufficient for stationarity).
- If f is convex and “part” of it is smooth, then coordinate-wise minimality implies global minimality.

Usage of coordinate descent

Coordinate descent is frequently used for optimizing non-smooth objectives of the form:

$$f(x) = g(x) + h(x),$$
$$h(x) = \sum_{i=1}^n h_i(x_i),$$

where g is smooth (and frequently convex) and h is convex but not smooth.

Such functions h are called [separable](#).

For example, the LASSO optimization in Lagrangian form,

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1$$

I.e., the ℓ_1 norm is separable. Also separable: box constraints via penalization.

Separable (and/or block separable) objectives are popular in sparse approximation, model/variable selection, kompressed sensing, and (group-)sparse regularization.

Usage of coordinate descent

Coordinate descent is frequently used for optimizing non-smooth objectives of the form:

$$f(x) = g(x) + h(x),$$

$$h(x) = \sum_{i=1}^n h_i(x_i),$$

where g is smooth (and frequently convex) and h is convex but not smooth.

Such functions h are called **separable**.

For example, the LASSO optimization in Lagrangian form,

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1$$

$\|x\|_1 = \sum_{j=1}^n |x_j|$

I.e., the ℓ_1 norm is separable. Also separable: box constraints via penalization.

Separable (and/or block separable) objectives are popular in sparse approximation, model/variable selection, compressed sensing, and (group-)sparse regularization.

Adaptive coordinate descent

There are many variants of coordinate descent. E.g., *adaptive* coordinate descent attempts to rotate variables via an empirical covariance matrix to decorrelate variables.

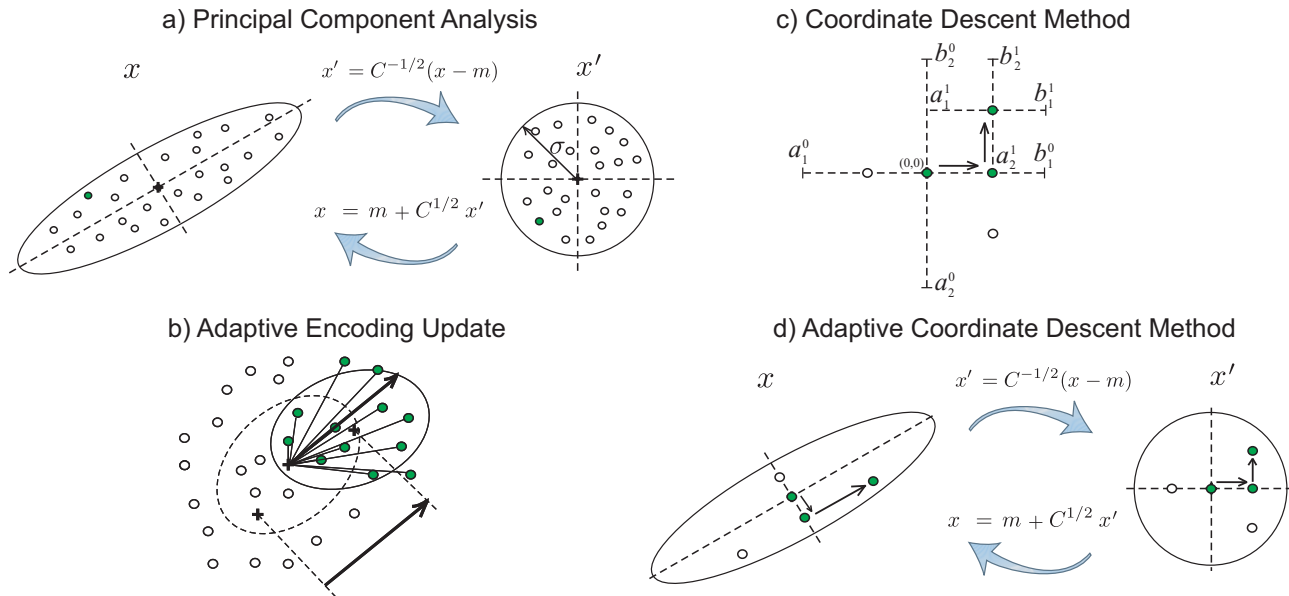


Image: Loshchilov et al, "Adaptive coordinate descent".

Bregman methods

Bregman methods excel at solving optimization problems of the form,

$$\min_{x \in \mathbb{R}^n} f(x) + g(x),$$

where $f(x)$ is a convex non-smooth “ ℓ^1 ”-type regularization term, and $g(x)$ is a convex (and typically smooth) data misfit term.

A common example is regularized basis pursuit problem:

$$\min_{x \in \mathbb{R}^n} \|x\|_1 + \frac{\lambda}{2} \|Ax - b\|_2^2,$$

or total variation regularization for reconstruction of an image u ,

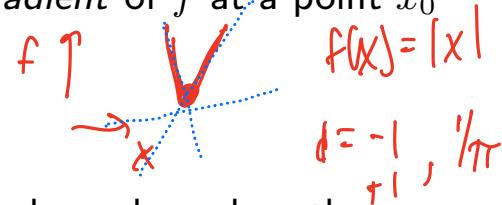
$$\min_u \|\nabla u\|_1 + \frac{\lambda}{2} \|Hu - f\|_2^2.$$

Subgradients

To introduce Bregman methods, we need the concept of **subdifferentials**, which quantify notions of descent for non-smooth convex functions.

Given a continuous, convex function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, a *subgradient* of f at a point x_0 is any $d \in \mathbb{R}^n$ such that for all x ,

$$f(x) - f(x_0) \geq d^T(x - x_0),$$



i.e., it is any direction for which a linear approximation is a lower bound on the function value.

The set $\partial f(x_0)$ of all subgradients of f at x_0 is the *subdifferential* of f at x_0 .

The subdifferential at x_0 is a closed, convex set.

The subdifferential $\partial f(x_0)$ is a singleton in \mathbb{R}^n if and only if f is differentiable.

With f provided, the *Bregman distance* between points $x, x_0 \in \mathbb{R}^n$ is

$$D_d(x, x_0) = f(x) - (f(x_0) + d^T(x - x_0)),$$

i.e., it is the “subgradient gap” in the direction d .

Subgradients

To introduce Bregman methods, we need the concept of [subdifferentials](#), which quantify notions of descent for non-smooth convex functions.

Given a continuous, convex function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, a *subgradient* of f at a point x_0 is any $d \in \mathbb{R}^n$ such that for all x ,

$$f(x) - f(x_0) \geq d^T(x - x_0),$$

i.e., it is any direction for which a linear approximation is a lower bound on the function value.

The set $\partial f(x_0)$ of all subgradients of f at x_0 is the *subdifferential* of f at x_0 .

The subdifferential at x_0 is a closed, convex set.

The subdifferential $\partial f(x_0)$ is a singleton in \mathbb{R}^n if and only if f is differentiable.

With f provided, the *Bregman distance* between points $x, x_0 \in \mathbb{R}^n$ is

$$D_d(x, x_0) = f(x) - (f(x_0) + d^T(x - x_0)),$$

i.e., it is the “subgradient gap” in the direction d .

The Bregman algorithm

$$\min_{x \in \mathbb{R}^n} f(x) + g(x),$$

The basic Bregman algorithm proceeds as follows: with x_0 an initial guess and $d_0 \in \partial f(x_0)$.

Set $k = 0$.

1. Solve

$$x_{k+1} = \arg \min_x D_{d_k}(x, x_k) + g(x).$$

This is frequently approximately solved, e.g., via descent methods.

2. Update the subgradient,

$$d_{k+1} = d_k - \nabla g(x_{k+1}) \in \partial f(x_{k+1}).$$

3. $k \leftarrow k + 1$ and iterate

One way to interpret the Bregman method: descend on g without going too far (in the f -Bregman sense) from the starting point.

Bregman method properties

The Bregman algorithm has some nice properties.

For example, if g is smooth and f satisfies certain weak continuity conditions, then

$$g(x_{k+1}) \leq g(x_k).$$

Recall that g is frequently a (smooth) data misfit term.

Hence Bregman methods ensure monotone behavior of data misfit.

There are various generalizations of Bregman methods.

E.g., *linearized* Bregman methods linearize g at each step so that subproblems frequently have explicit solutions.

One generalization that is relevant for us is the split Bregman method.

Bregman method properties

The Bregman algorithm has some nice properties.

For example, if g is smooth and f satisfies certain weak continuity conditions, then

$$g(x_{k+1}) \leq g(x_k).$$

Recall that g is frequently a (smooth) data misfit term.

Hence Bregman methods ensure monotone behavior of data misfit.

There are various generalizations of Bregman methods.

E.g., *linearized* Bregman methods linearize g at each step so that subproblems frequently have explicit solutions.

One generalization that is relevant for us is the split Bregman method.

Split Bregman, I

The split Bregman approach is an alternating algorithm.

We consider a special kind of objective,

$$\min_x \|\Phi(x)\|_1 + g(x),$$

e.g. $\Phi(x) = Ax$

where g is convex and smooth (typically g is a quadratic misfit), and Φ is convex (e.g., affine).

Split Bregman first rewrites the problem as a constrained one:

$$\min_{x: \Phi(x)=d} \|d\|_1 + g(x),$$

and subsequently relaxes the constraint,

$$\min_{x,d} \|d\|_1 + g(x) + \frac{\lambda}{2} \|d - \Phi(x)\|_2^2.$$

We can now define

$$f(x) = \|d\|_1 + g(x).$$

Split Bregman, II

$$\min_{x,d} f(x) + \frac{\lambda}{2} \|d - \Phi(x)\|_2^2, \quad f(x) = \|d\|_1 + g(x)$$

Split Bregman now essentially just applied “ordinary” Bregman to the above.

The algorithm is an alternating scheme by realizing that f is separable in (d, x) . In addition, one introduces an augmented Lagrange variable b for the data misfit term:

– Solve

$$\cancel{x_{k+1}} = \arg \min_x g(x) + \frac{\lambda}{2} \|d_k - \Phi(x) - b_k\|_2^2$$

– Solve

$$d_{k+1} = \arg \min_x \|d\|_1 + \frac{\lambda}{2} \|d - \Phi(x_{k+1}) - b_k\|_2^2$$

– Solve

$$b_{k+1} = b_k + \Phi(x_{k+1}) - d_{k+1}$$

Split Bregman methods frequently perform well for non-smooth ℓ_1 -type regularization objectives.

Alternating Direction Method of Multipliers (ADMM)

The ADMM algorithm combines advantages of **dual ascent**, and the method of multipliers.

We've seen dual ascent before: consider

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

By forming the Lagrangian,

$$L(x, \lambda) = f(x) + \lambda^T (Ax - b),$$

then the dual function is the minimum.

$$f_*(\lambda) = \min_x L(x, \lambda) \rightarrow \text{has an explicit form for "nice" problems.}$$

And when there is zero duality gap, we solve the dual problem and recover the primal solution,

$$\lambda_* = \max_{\lambda} f_*(\lambda),$$

$$x_* = \min_x L(x, \lambda_*).$$

Alternating Direction Method of Multipliers (ADMM)

The ADMM algorithm combines advantages of **dual ascent**, and the method of multipliers.

Dual ascent performs gradient ascent on the dual (maximization) problem.

In particular, the gradient of the dual problem is explicitly computable if the dual problem f_* is smooth.

In this case, the algorithm becomes,

- $x_{k+1} = \arg \min_x L(x, \lambda_k)$
- $\lambda_{k+1} = \lambda_k + \alpha_k (Ax_{k+1} - b),$

where $\alpha_k > 0$ is a stepsize.

Alternating Direction Method of Multipliers (ADMM)

The ADMM algorithm combines advantages of **dual ascent**, and the method of multipliers.

Dual ascent performs gradient ascent on the dual (maximization) problem.

In particular, the gradient of the dual problem is explicitly computable if the dual problem f_* is smooth.

In this case, the algorithm becomes,

- $x_{k+1} = \arg \min_x L(x, \lambda_k)$
- $\lambda_{k+1} = \lambda_k + \alpha_k (Ax_{k+1} - b)$,

where $\alpha_k > 0$ is a stepsize.

Dual ascent is computationally nice because it can directly exploit separability: If

$$f(x) = \sum_{j=1}^n f_j(x_j), \quad Ax = (a_1 \cdots a_n) x,$$

then one can form individual Lagrangians,

$$L_j(x_j, \lambda) = f_j(x_j) + \lambda^T a_j x_j - \frac{1}{n} \lambda^T b,$$

and update each x_j independently. Unfortunately, convergence of dual ascent requires unrealistic assumptions.

Alternating Direction Method of Multipliers (ADMM)

The ADMM algorithm combines advantages of dual ascent, and the [method of multipliers](#).

The method of multipliers employs augmented Lagrangians.

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

Augmented Lagrangians penalize the constraint above,

$$\min_x f(x) + \frac{\mu}{2} \|Ax - b\|_2^2 \quad \text{subject to } Ax = b,$$

and subsequently form the (“standard”) Lagrangian,

$$L_\mu(x, \lambda) = f(x) + \frac{\mu}{2} \|Ax - b\|_2^2 + \lambda^T (Ax - b).$$

Then again we form the dual function,

$$f_*(\lambda) = \min_x L_\mu(x, \lambda),$$

and perform ascent,

- $x_{k+1} = \arg \min_x L_\mu(x, \lambda_k)$
- $\lambda_{k+1} = \lambda_k + \mu(Ax_{k+1} - b),$

The method of multipliers converges under milder conditions than dual ascent ☺

But the augmented Lagrangian is not separable ☹

Alternating Direction Method of Multipliers (ADMM)

The **ADMM algorithm** combines advantages of dual ascent, and the method of multipliers.

First we assume f is block-separable and write the problem as,

$$\min_x f(x) + g(y), \quad \text{subject to } Ax + By = c.$$

Above we have (x, y) is the split version of the previous x .

ADMM proceeds by forming the augmented Lagrangian,

$$L_\mu(x, y, \lambda) = f(x) + g(y) + \frac{\mu}{2} \|Ax + By - c\|_2^2 + \lambda^T (Ax + By - c).$$

ADMM then constructs the alternating minimization scheme,

$$x_{k+1} = \arg \min_x L_\mu(x, y_k, \lambda_k)$$

$$y_{k+1} = \arg \min_y L_\mu(x_{k+1}, y, \lambda_k)$$

$$\lambda_{k+1} = \lambda_k + \mu(Ax_{k+1} + By_{k+1} - c)$$

Alternating Direction Method of Multipliers (ADMM)

The **ADMM algorithm** combines advantages of dual ascent, and the method of multipliers.

First we assume f is block-separable and write the problem as,

$$\min_x f(x) + g(y), \quad \text{subject to } Ax + By = c.$$

Above we have (x, y) is the split version of the previous x .

ADMM proceeds by forming the augmented Lagrangian,

$$L_\mu(x, y, \lambda) = f(x) + g(y) + \frac{\mu}{2} \|Ax + By - c\|_2^2 + \lambda^T (Ax + By - c).$$

ADMM then constructs the alternating minimization scheme,

$$x_{k+1} = \arg \min_x L_\mu(x, y_k, \lambda_k)$$

$$y_{k+1} = \arg \min_y L_\mu(x_{k+1}, y, \lambda_k)$$

$$\lambda_{k+1} = \lambda_k + \mu(Ax_{k+1} + By_{k+1} - c)$$

ADMM in practice


ADMM looks a lot like the method of multipliers: if we use block Gauss-Seidel on (x, y) for the method of multipliers formulation, this is ADMM.

The alternation between x and y keeps the procedure distributed if f and g are separable.


One also has nice convergence properties of ADMM: Under mild assumptions on f and g , then

- $\|Ax_k + By_k - c\|_2 \rightarrow 0$ as $k \uparrow \infty$.
- $f(x_k) + g(y_k) \rightarrow \min_{(x,y): Ax+By=c} f(x) + g(y)$ as $k \uparrow \infty$

Related papers I

-  Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, Foundations and Trends® in Machine Learning **3** (2011), no. 1, 1–122.
-  Tom Goldstein and Stanley Osher, *The Split Bregman Method for L1-Regularized Problems*, SIAM Journal on Imaging Sciences **2** (2009), no. 2, 323–343.
-  Gene H. Golub and Charles F. Van Loan, *Matrix Computations Johns Hopkins Studies in Mathematical Sciences*, 3rd ed., The Johns Hopkins University Press, 1996.
-  Ilya Loshchilov, Marc Schoenauer, and Michele Sebag, *Adaptive coordinate descent*, Proceedings of the 13th annual conference on Genetic and evolutionary computation, 2011, pp. 885–892.
-  Yu. Nesterov, *Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems*, SIAM Journal on Optimization **22** (2012), no. 2, 341–362.

Related papers II

-  Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin, *An Iterative Regularization Method for Total Variation-Based Image Restoration*, *Multiscale Modeling & Simulation* **4** (2005), no. 2, 460–489.
-  Ralph Tyrell Rockafellar, *Convex Analysis*; Princeton University Press, Princeton, NJ, 1996 (English).
-  P. Tseng, *Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization*, *Journal of Optimization Theory and Applications* **109** (2001), no. 3, 475–494.
-  Stephen J. Wright, *Coordinate descent algorithms*, *Mathematical Programming* **151** (2015), no. 1, 3–34.