Optimisation

Lecture 13 - Alternating Direction Method of Multipliers (ADMM)

Fall semester - 2024

Dr. Eng. Valentin Leplat

Innopolis University

November 25, 2024

Outline

1. Introduction
   - ADMM's main insights
   - Assumptions
2. Dual decomposition
3. Method of Multipliers
4. Alternating Direction Method of Multipliers (ADMM)
5. Commom patterns
6. Examples
7. Conclusions

Introduction

In a nutshell

- Main Purpose: The Alternating Direction Method of Multipliers (ADMM) is an algorithm that solves convex optimization problems.
- Main Principle: **ADMM** algorithm breaks problems into smaller pieces, each of which are then easier to handle.
- Motivation: found wide application in the last decade in areas such as large-scale problems arising in statistics and machine learning. *Reason*: **ADMM** is well suited to distributed convex optimization
- Origin: developed in the 1970s, with roots in the 1950s.
  Related to many other algorithms: dual decomposition, the method of multipliers, Douglas-Rachford splitting, proximal methods, and others.
- About the roadmap of this course: we first introduce some basic assumptions. Before introducing **ADMM** itself, two methods need to be presented first, namely **Dual decomposition** and **Method of Multipliers**. **ADMM** is born from the desire to get the best of those two methods.

Blanket assumptions

this lecture: underlying space is the Euclidean space $\mathbb{R}^n$, a particular case of Hilbert space $\mathcal{X}$ of finite dimension n, that is, a Banach space equipped with:

- an inner product $\langle ., . \rangle$, here we consider the dot product $\langle x, y \rangle = \sum_i^n x_i y_i$ for $x, y \in \mathbb{R}^n$,
- induced norm $\|.\| = \sqrt{\langle ., . \rangle}$

Set of assumptions on functions

Through this lecture:

- focus on the minimization of a function f over a convex set by basically tackling its dual problem or a modified version of it (”Augmented”).
- important to always keep in mind the set of assumptions made on the functions, in particular on f (the primal objective function).
- The most important assumptions will be highlighted in red when needed
- The derived results (mainly linked to the convergence results of the algorithms discussed in the present document) are only valid in the set of assumptions considered (= the paradigm).

Dual decomposition

Dual Problem

- Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $f : \mathbb{R}^n \to \mathbb{R}$ a convex function , we are interested in solving:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{1}$$

- Form the **Lagrangian** function: $L(x, y) = f(x) + \langle y, Ax - b \rangle$ where $Ax - b$ is the residual and $y \in \mathbb{R}^m$ are referred to as Lagrangian multipliers (= dual variables).

  What does it mean ?: we allow the constraints to be violated, but there is a price to pay which is given by the term $\langle y, Ax - b \rangle$.

Dual Problem

- Build the **dual** function: $q(y) := \min_x L(x, y)$
  Particular structure for $q(y)$: the dual function is always concave.
- Dual Problem: $\text{maximize}_y \quad q(y)$
- Let us assume that the maximization goes well.
  By denoting $y^\star := \underset{y \in \mathbb{R}^m}{\text{argmax}} \quad q(y)$, we may recover $x^\star := \underset{x \in \mathbb{R}^n}{\text{argmin}} \quad L(x, y^\star)$.
  Does this approach always work ? : NO. Actually this may fail even for linear problems, especially if we do not define properly the domain of $q(.)$!
  However, with enough and appropriate assumptions, this method works (ex: minimal curvature $\iff$ not too "flat").

Solve the Dual Problem: the Dual ascent scheme

- How to maximize the dual function ?
  → Use an **ascent** gradient method: $y^{k+1} := y^k + t^k \nabla q(y^k)$
  where t^k is the step size (positive scalar), k denotes the iteration counter, and $q(y)$ is assumed to be differentiable w.r.t. y [1].
- $\nabla q(y^k) = A\tilde{x} - b$, where $\tilde{x} := \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \quad L(x, y^k)$
- Dual ascent method is an iterative scheme such that at each iteration we have two steps:

1. $x^{k+1} := \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \quad L(x, y^k)$ %x-minimization
2. $y^{k+1} := y^k + t^k(Ax^{k+1} - b)$ %dual update

[1] it requires f to be closed and strictly convex, if not, a subgradient method should be used

Solve the Dual Problem: the Dual ascent scheme

Few remarks:

- Step 1: consider L, fix the dual vector and minimize over x.
- Step 2: calculate the residual
  1. If the residual is zero, optimality reached → stop
  2. Otherwise, update the dual variables by some positive number times the residual
- **Two significant differences with Lecture 12**:
  1. We do not build $q(.)$ explicitly (we only want to have access to its gradient)
  2. We do not care about $\text{dom}(q)$...

  because these two actions may simply be too complicated.
- This method works, but again with many strong assumptions.
- *Why do we solve the dual instead of the primal ?:* The reason has to do with the notion of dual decomposition...

Additional Assumption: Separability

- Assume that function f is separable:

$$f(x) = f_1(x_1) + ... + f_N(x_N), \quad x := (x_1, ..., x_N)^T$$

  where $1 \leqslant N \leqslant n$.
  It means that f can be expressed as a sum of functions of individual blocks.

- then L is separable in x: $L(x,y) = \sum_i^N L_i(x_i, y) - \langle y, b\rangle$
  with $L_i(x_i, y) = f_i(x_i) + \langle y, A_i x_i\rangle$ and $A_i \in \mathbb{R}^{m \times s(i)}$ where $s(i)$ denotes the size of block x_i.

- x-minimization step from dual ascent scheme splits into N independent problems:

$$x_i^{k+1} := \underset{x_i \in \mathbb{R}^{s(i)}}{\text{argmin}} \quad L_i(x_i, y^k)$$

  ....which can be solved in parallel !

Dual decomposition

Dual Decomposition algorithm (Everett, Dantzig, Wolfe, Benders, 1960-65) is:

1. $\forall i \in [1, N] : x_i^{k+1} := \underset{x_i \in \mathbb{R}^{s(i)}}{\operatorname{argmin}} \quad L_i(x_i, y^k)$ %x_i-minimization in parallel
2. $y^{k+1} := y^k + t^k(\sum_i^N A_i x_i^{k+1} - b)$ %dual update

- **Step 1**: minimization of each Lagrangian term L_i separately
- **Step 2**: gathering of the contributions to the equality constraints
  If the residual is zero → Stop
  Otherwise, update of the dual variables

Dual decomposition

- *What is required ?*:
  1. a scattering of the dual variables (y^k),
  2. update of x_i in parallel,
  3. a gathering $(\sum_i^N A_i x_i^{k+1})$. (provides coordination)
- → Distributed (convex) optimization !
  → Allow to solve large problems !
- works with a lot of assumptions; often slow.
  Reason: to have interesting properties on $q(y)$ for minimization schemes, it requires strong assumptions on f.

Method of Multipliers

Method of Multipliers (AKA. Augmented Lagrangian Methods)

- Goal: develop a method to make the Dual Ascent more robust (so that it would work for problems as linear problems, but not only !)
- Main idea: use an **augmented Lagrangian** (Hestenes, Powell, 1969):

$$L_\rho(x, y) = f(x) + \langle y, Ax - b\rangle + \frac{\rho}{2}\|Ax - b\|_2^2$$

where $\rho > 0$ is a constant (the penalty parameter for the constraint).

- **Method of Multipliers** (Hestenes, Powell; analysis in Bertsekas in 1982)

1. $x^{k+1} := \operatorname*{argmin}_{x \in \mathbb{R}^n} \quad L_\rho(x, y^k)$ %x-minimization
2. $y^{k+1} := y^k + \rho(Ax^{k+1} - b)$ %dual update with step length ρ

Method of Multipliers: Remarks

An interesting and alternative introduction of the concept: the method modifies the problem (1) as follows:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) + \frac{\rho}{2}\|Ax - b\|_2^2 \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{2}$$

Clearly extra term $\frac{\rho}{2}\|Ax - b\|_2^2$ does not change problem. On top of that:

Proposition

Let $h = f + g$ where f is convex function and g is a strongly convex function, then h is strongly convex function.

Hence, assuming, e.g., A has full rank, primal objective is strongly convex of parameter $\rho\sigma_{\min}^2(A)$ (f is then also strictly convex), the dual function gets good properties for the use of dual ascent method.

Method of Multipliers: Remarks

Differences with Dual Ascent method:

- Addition of a nonnegative term which is a quadratic penalty (always a cost) to the Lagrangian function.
- The gain for the dual update has a very specific step size = the penalty parameter ρ.

Why ?...

Method of Multipliers: dual update step

- Let us write the first-order optimality conditions for Problem (1) (for f differentiable):
  1. $Ax^\star - b = 0$ (Primal feasibility),
  2. $\nabla_x L(x^\star, y^\star) = \nabla_x f(x^\star) + A^T y^\star = 0$ (Dual feasibility).
- Step 1 from **Method of Multipliers**: x^{k+1} is the minimizer of $L_\rho(x, y^k)$, therefore it satisfies the following condition (unconstrained optimization problem):

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, y^k) \\ &= \nabla_x f(x^{k+1}) + A^T y^k + \rho(A^T A x^{k+1} - A^T b) \\ &= \nabla_x f(x^{k+1}) + A^T \left(y^k + \rho\left(Ax^{k+1} - b\right)\right) \\ &= \nabla_x f(x^{k+1}) + A^T y^{k+1} \end{aligned} \tag{3}$$

Method of Multipliers: dual update step

- With step size ρ, after each iteration of the Method of Multipliers, algorithm generates iterates (x^{k+1}, y^{k+1}) that are *dual feasible*.
- As the algorithm progresses, we hope that the residual $Ax^{k+1} - b \to 0$ to obtain the primal feasibility and hence reach the optimality.

Method of Multipliers: take-home messages

This method is similar to dual decomposition (particular case of dual ascent for which f is separable) but with two differences that induce:

- An advantage: Adding the quadratic term [2] allows the Method of Multipliers to converge under less strong assumptions (it basically works in any cases), for instance f can be non-differentiable, take on value $+\infty$ in the case f is an indicator function for a convex set $\mathcal{C} \subseteq \mathbb{R}^n$.
- A drawback: quadratic penalty term destroys splitting for the update of x (so, cannot do decomposition).

In summary: robust but non-separable.

[2] Recently, other types of augmented terms gained interests such as the Bregman divergences or entropy.

Alternating Direction Method of Multipliers (ADMM)

ADMM

- Goal: develop a method
  1. with the good robustness of method of multipliers,
  2. which can support decomposition for allowing distributed optimization
- proposed by Gabay, Mercier, Glowinski, Marrocco in 1976 (however, we have discovered in 80s that **ADMM** could be re-derived from materials from the 50s..., probably in Moscow :)).
  The main point is: this is not new.

ADMM - (Western) Problem form

- Given f and g convex functions, we are interested in solving:

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c, \end{aligned} \tag{4}$$

where:

1. x and z are two sets of variables (previous variable x splitted into two groups),
2. the objective is required to be separable across x and z,
3. the general equality constraint links x and z.

ADMM - Algorithm

- Build the **augmented Lagrangian**:

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

- **ADMM**: Iterative scheme, at each iteration → three steps:

1. $x^{k+1} := \underset{x}{\operatorname{argmin}} \quad L_\rho(x, z^k, y^k)$ %x-minimization
2. $z^{k+1} := \underset{z}{\operatorname{argmin}} \quad L_\rho(x^{k+1}, z, y^k)$ %z-minimization
3. $y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)$ %dual update with step length ρ

ADMM - Remarks

1. If we minimize over x and z jointly, **ADMM** boils down to the **Method of Multipliers**.
2. We get **Decomposition** since we minimize over x with z fixed, and vice versa.

By the way: we are free to minimize over x and z as many times as we want before updating the dual variables.

ADMM - Optimality conditions

- Let us write the first-order optimality conditions for Problem (4) (for f and g differentiable):
  1. $Ax^\star + Bz^\star - c = 0$ (Primal feasibility),
  2. $\nabla_x L(x^\star, z^\star, y^\star) = \nabla_x f(x^\star) + A^T y^\star = 0$ (x-Dual feasibility).
  3. $\nabla_z L(x^\star, z^\star, y^\star) = \nabla_z g(z^\star) + B^T y^\star = 0$ (z-Dual feasibility).
- Step 2 from **ADMM**: z^{k+1} is the minimizer of $L_\rho(x^{k+1}, z, y^k)$, therefore it satisfies the following condition (unconstrained optimization problem):

$$\begin{aligned} 0 &= \nabla_z L_\rho(x^{k+1}, z^{k+1}, y^k) \\ &= \nabla_z g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla_z g(z^{k+1}) + B^T \left(y^k + \rho \left(Ax^{k+1} + Bz^{k+1} - c \right) \right) \\ &= \nabla_x g(z^{k+1}) + B^T y^{k+1} \end{aligned} \tag{5}$$

ADMM - Optimality conditions

- With step size ρ, after each iteration of **ADMM**, algorithm generates iterates $(x^{k+1}, z^{k+1}, y^{k+1})$ that are z-*dual feasible*.
- As the algorithm progresses, we hope that:
  1. the residual $Ax^{k+1} + Bz^{k+1} - c \to 0$ to obtain the primal feasibility,
  2. $(x^{k+1}, z^{k+1}, y^{k+1})$ are x-*dual feasible*.

  and hence reach the optimality.

ADMM - The scaled dual form

It is often easier to express the **ADMM** algorithm in scaled form, where we replace the dual variable y by a scaled variable $u = \frac{y}{\rho}$

- Idea: Combine linear and quadratic terms in augmented Lagrangian:

$$\begin{aligned} L_\rho(x,z,y) &= f(x) + g(z) + \langle y, Ax + Bz - c\rangle + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ &= f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c + u\|_2^2 + \text{const} \end{aligned}$$

- In this form, the **ADMM** steps are:

1. $x^{k+1} := \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2\right)$ %x-minimization
2. $z^{k+1} := \underset{z}{\operatorname{argmin}} \left(g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|_2^2\right)$ %z-minimization
3. $u^{k+1} := u^k + (Ax^{k+1} + Bz^{k+1} - c)$ %dual update

ADMM - Convergence guarantees

- Under (very little !) assumptions :
  - on f and g:
    1. are convex,
    2. are closed: a function $l : \mathbb{R}^n \to \mathbb{R}$ is closed if its epigraph, denoted and defined by epi(f)=$\{(x,t)|f(x) \leqslant t$ for $x \in \mathbb{R}^n, t \in \mathbb{R}\}$, is closed.
    3. are proper: a function l is called proper if l does not take the value $-\infty$ and dom(l) is nonempty.
  - L_0 has a saddle point (roughly speaking; the primal problem needs to have a solution, otherwise there is no point)

    These assumptions do not require A and B to be full rank.
- the **ADMM** iterates satisfy, for any $\rho > 0$:
  1. *Residual* convergence: $r^k = Ax^k + Bz^k - c \to 0$ as $k \to \infty$, i.e., primal iterates approach feasibility.
  2. *Objective* convergence: $f(x^k) + g(z^k) \to f^\star + g^\star$, where $f^\star + g^\star$ is the optimal primal objective value.
  3. *Dual* convergence: $u^k \to u^\star$, where $u^\star$ is a dual solution.

ADMM - Convergence guarantees

- The convergence garantees are **very carefully** worded. Many things are not said because they are FALSE, for instance:

  1. x^k converges → FALSE
  2. z^k converges → FALSE
  3. (x^k, z^k) converge to the optimal set → FALSE
  4. (x^k, z^k) converge to $(x^\star, z^\star)$ which are the optimal primal solutions → FALSE

  In summary: we do not generically get primal convergence, but this is true under more assumptions (*Question: is it that relevant ? Because we are loosing the all point of this methods that work "no matter what"*).

- Convergence rate: roughly, **ADMM** behaves like first-order method
  Theory still being developed, see, e.g., in Hong and Luo (2012), Deng and Yin (2012), Iutzeler et al. (2014), Nishihara et al. (2015)

ADMM - Practicalities and tricks

- In practice, **ADMM** obtains a relatively accurate solution in a handful of iterations, but requires many, many iterations for a highly accurate solution. Hence it behaves more like a first-order method than a second-order method.
- Choice of ρ, can greatly influence practical convergence of **ADMM**:
  - ρ is too large → not enough emphasis on minimizing $f + g$
  - ρ is too small → not enough emphasis on feasibility

  (Boyd et al, 2010) give a strategy for varying ρ that is useful in practice (but without convergence guarantees).
- Like deriving duals, transforming a problem into one that **ADMM** can handle is sometimes a bit subtle, since different forms can lead to different algorithms.

Commom patterns

Common patterns

Let us consider the **ADMM** scaled form,

- x-update step from **ADMM** requires minimizing $f(x) + \frac{\rho}{2}\|Ax - v\|_2^2$ with $v = -Bz^k + c - u^k$ is a constant during x-update.[3]
- similar for z-update
- several special cases come up often (particular choice for g and/or for A, B and c)
- can simplify the update by **exploiting** the structure of these special cases

[3] Note that **ADMM** can be seen as a meta-algorithm, we are dealing with more abstract objects here compared to classical optimization algorithms where you end up working with primitives such as gradients or subgradients. Except for special cases, the operators are higher levels, they require to minimize a quadratic augmented function (can be really challenging !).

Decomposition

- Assume that function f is separable:

$$f(x) = f_1(x_1) + \ldots + f_N(x_N), \quad x := (x_1, \ldots, x_N)^T$$

  where $1 \leqslant N \leqslant n$.
- Assume that $A^T A$ is block diagonal w.r.t. blocks $x := (x_1, \ldots, x_N)^T$, (as a simple illustration consider the case $A = I$),
- then the $f(x) + \frac{\rho}{2}\|Ax - v\|_2^2$ splits into N components,
  $\rightarrow$ update of x_i in parallel, for $i = 1, \ldots, N$.

Quadratic objective

- $f(x) = 1/2\langle x, Px\rangle + \langle q, x\rangle + r$
  $\rightarrow$ x-update requires to minimize a quadratic without constraints, can be done in closed form:x^{k+1} is the solution of $\min_x \ l(x) := 1/2\langle x, Px\rangle + \langle q, x\rangle + r + \frac{\rho}{2}\|Ax - v\|_2^2$ where $v = -Bz^k + c - u^k$ is a constant. Hence we are looking for x such that:

  $$\begin{aligned} 0 &= \nabla l(x) \\ &= \left(P + \rho A^T A\right) x + \left(q - \rho A^T v\right) \end{aligned}$$

  Hence $x^{k+1} := \left(P + \rho A^T A\right)^{-1} \left(\rho A^T v - q\right)$
- The problem: the computation of the inverse !
  - Depending on sparsity patterns, you may use matrix inversion lemma:

    $$\left(P + \rho A^T A\right)^{-1} = P^{-1} - \rho P^{-1} A^T (I + \rho A P^{-1} A^T)^{-1} A P^{-1}$$

  - Dense case: (direct method) cache factorization of $\left(P + \rho A^T A\right)$ (LU or Cholesky if symmetric pd).

Connection to proximal operators

- Consider

  $$\min_{x} f(x) + g(x) \iff \min_{x,z} f(x) + g(z) \text{ subject to } x - z = 0$$

  a particular case of Problem (4) with $A = I_{n\times n}$, $B = -I_{n\times n}$ and $c = \bar{0}$

- Build the **augmented Lagrangian** (scaled form):

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

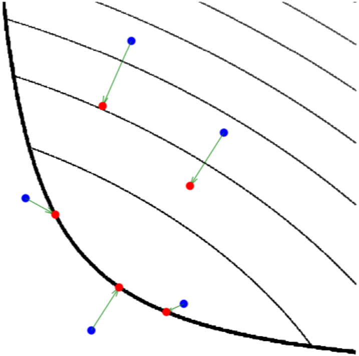
- Consider x-update with $v^k = z^k - u^k$:

  $$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x \left(f(x) + \frac{\rho}{2}\|x - v^k\|_2^2 \right) \\ &:= \mathbf{prox}_{1/\rho,f}(v^k) \end{aligned}$$

  Where $\mathbf{prox}_{1/\rho,f}(v^k)$ is referred to as the proximal operator $\mathbf{prox}_{1/\rho,f} : \mathbb{R}^n \to \mathbb{R}^n$ of f with parameter $1/\rho$.

Interpretation of proximal operators

What a prox does: $\mathbf{prox}_{\lambda,f}$(blue points) → red points



Two cases:

1. three points in the domain of the function stay in the domain and move towards the minimum of the function
2. the other two move to the boundary of the domain and towards the minimum of the function

Note that:

- thin black lines:= level curves of a convex function f
- thicker black line := the boundary of its domain
- λ := trade-off parameter between the two terms.
  What happens if λ is big/small ?

Special cases of proximal operators

When f is the indicator function:

$$I_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if} \quad x \in \mathcal{C}, \\ +\infty & \text{if} \quad x \notin \mathcal{C}, \end{cases}$$

where $\mathcal{C}$ is a closed nonempty convex set. The proximal operator becomes:

$$\begin{aligned} \mathbf{prox}_{\lambda,f}(v) &:= \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{1}{2\lambda}\|x-v\|_2^2 \right) \\ \iff &:= \underset{x\in\mathcal{C}}{\operatorname{argmin}} \left(\underbrace{f(x)}_{=0} + \frac{1}{2\lambda}\|x-v\|_2^2 \right) \\ \iff &= \underset{x\in\mathcal{C}}{\operatorname{argmin}} \left(\|x-v\|_2^2 \right) = \Pi_{\mathcal{C}}(v) \end{aligned}$$

Hence the proximal operator of f reduces to Euclidean projection onto $\mathcal{C}$.
...Proximal operators can thus be viewed as generalized projections !

Proximal operators associated to ℓ^1-norm

The proximal operator (see Appendix A for all the details):

$$\mathbf{prox}_{\lambda,f}(v) := \underset{x}{\operatorname{argmin}} \left(\|x\|_1 + \frac{1}{2\lambda}\|x - v\|_2^2 \right)$$

can be computed by the following update rule:

$$\mathbf{prox}_{\lambda,f}(v) = S_\lambda(v)$$

where S_λ is the soft thresholding operator applied on v componentwise:

$$\left[\mathbf{prox}_{\lambda,f}(v)\right]_i = \operatorname{sgn}(v_i)\left(|v_i| - \lambda\right)_+$$

Remark: the proximal operator has been computed in closed form !

Proximal operators and MatLab/Python Toolboxes

For further readings,

1. getting access to an open-source MatLab toolbox called Unlocbox dedicated to convex optimization (in particular proximal methods and ADMM), the reader is invited to consult:
   ▸ UNLOCBOX - documentation and ▸ UNLOCBOX - Proximal operators
2. Examples of Proximal operators that can be solved in closed form:
   ▸ Link
3. Python libraries for efficient computation of proximal operators
   ▸ THE PROXIMITY OPERATOR REPOSITORY

Examples

Constrained convex optimization

- Consider **ADMM** for a generic convex problem:

$$\begin{array}{ll} \min_{x} & f(x) \\ \text{subject to} & x \in \mathcal{C}, \end{array}$$

- **ADMM** form: take g as the indicator function of the set $\mathcal{C}$:

$$\begin{array}{ll} \min_{x,z} & f(x) + g(z) \\ \text{subject to} & x - z = 0, \end{array}$$

- Build the **augmented Lagrangian** (scaled form):

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

Constrained convex optimization

By combining the special cases of proximal operators presented previously, we get the **ADMM** steps (equivalent to Douglas-Rachford, here):

1. $x^{k+1} := \mathbf{prox}_{1/\rho,f}(z^k - u^k)$ %x-minimization
2. $z^{k+1} := \Pi_{\mathcal{C}}\left(x^{k+1} + u^k\right)$ %z-minimization
3. $u^{k+1} := u^k + (x^{k+1} - z^{k+1})$ %dual update

Alternating projections, revisited

Consider finding a point x in intersection of (simple) convex sets $\mathcal{C}$ and $\mathcal{D}$.

- Let us formulate this as an optimization problem:

$$\begin{array}{ll} \min\limits_{x} & 0 \\ \text{subject to} & x \in \mathcal{C}, x \in \mathcal{D} \end{array}$$

- **ADMM** form: take f, g as the indicators function of the set $\mathcal{C}$ and $\mathcal{D}$ resp.:

$$\begin{array}{ll} \min\limits_{x,z} & f(x) + g(z) \\ \text{subject to} & x - z = 0, \end{array}$$

Alternating projections, revisited

By combining the special cases of proximal operators presented previously, we get the **ADMM** steps:

1. $x^{k+1} := \Pi_{\mathcal{C}}(z^k - u^k)$ %x-minimization
2. $z^{k+1} := \Pi_{\mathcal{D}}\left(x^{k+1} + u^k\right)$ %z-minimization
3. $u^{k+1} := u^k + (x^{k+1} - z^{k+1})$ %dual update

This is like the classical alternating projections method, but now with a dual variable u (much more efficient).

Nonnegative Least squares

Nonnegative Least Squares (NNLS): Given $Q \in \mathbb{R}_+^{m \times n}$, $b \in \mathbb{R}_+^m$, we are interested in solving:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2}\|Qx - b\|_2^2 \\ \text{subject to} \quad & x \geqslant 0, \end{aligned} \tag{6}$$

where x is required to be componentwise nonnegative, in other words, x must belong to $\mathbb{R}_+^n$ (nonnegative orthant of dimension n). This set is convex hence Problem (6) is convex. Why ?...

What else can we say ?...

Nonnegative Least squares

- **ADMM** form:
  - let f be the objective function of (6) (convex quadratic),
  - let g be the indicator function of the set $\mathbb{R}^n_+$,

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{subject to} \quad & x - z = 0, \end{aligned}$$

- Build the **augmented Lagrangian** (scaled form):

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

- **ADMM** steps:

1. $x^{k+1} := \operatorname*{argmin}_x \left(f(x) + \frac{\rho}{2}\|x - z^k + u^k\|_2^2\right)$ %x-minimization
2. $z^{k+1} := \operatorname*{argmin}_z \left(g(z) + \frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2\right)$ %z-minimization
3. $u^{k+1} := u^k + (x^{k+1} - z^{k+1})$ %dual update

Nonnegative Least squares: Steps 1 and 2

Step 1
$f(x) = \frac{1}{2}\|Qx - b\|_2^2$ is a quadratic (expand the squared l-2 norm). As explained previously, by setting the gradient to zero, we obtain:

$$x^{k+1} := \left(Q^T Q + \rho I\right)^{-1}\left(\rho v^k + Q^T b\right)$$

where $v^k = z^k - u^k$. Note that in this particular case, $Q^T Q + \rho I$ is always positive definite for $\rho > 0$, thus it is invertible.

Step 2
Since g is an indicator function for a set, namely $\mathbb{R}^n_+$, we have seen previously that z-update is:

$$z^{k+1} := \Pi_{\mathbb{R}^n_+}\left(x^{k+1} + u^k\right) = [x^{k+1} + u^k]_+$$

where $[a]_+ = \max(a, 0)$.

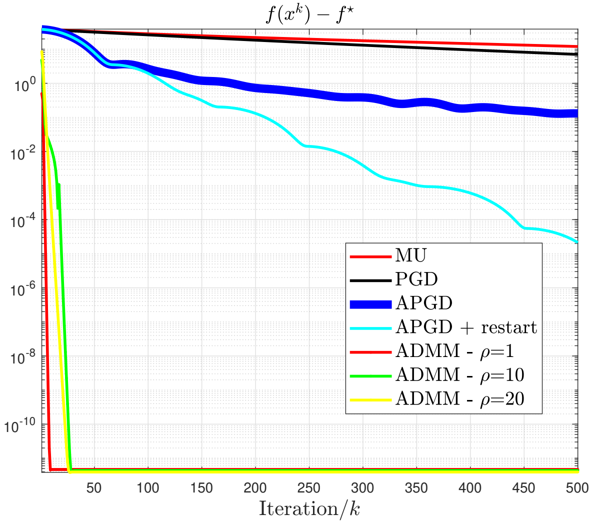
Nonnegative Least squares

Both Steps 1 and 2 have closed form solution.
Finally, **ADMM** steps:

1. $x^{k+1} := \left(Q^TQ + \rho I\right)^{-1}\left(\rho v^k + Q^Tb\right)$ %x-minimization
2. $z^{k+1} := [x^{k+1} + u^k]_+$ %z-minimization
3. $u^{k+1} := u^k + (x^{k+1} - z^{k+1})$ %dual update

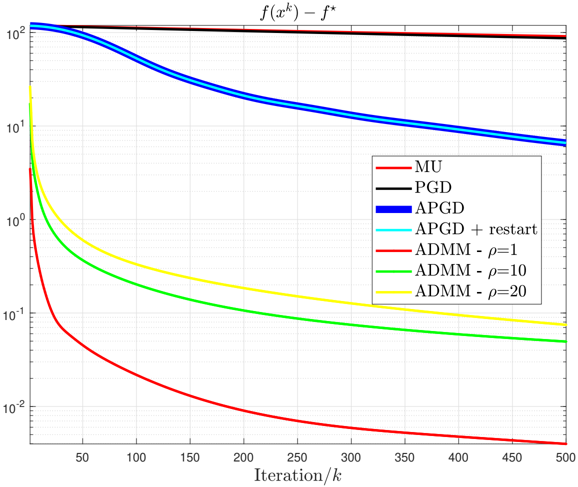
Nonnegative Least squares: numerical experiments

Comparison of various algorithms for Nonnegative Least squares: instance with $m = 1000$, $n = 100$



Nonnegative Least squares: numerical experiments

Comparison of various algorithms for Nonnegative Least squares: instance with $m = 1000$, $n = 1000$



▸ MatLab Code

Least Absolute Shrinkage and Selection Operator (Lasso)

- Given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, we are interested in solving:

$$\min_{\beta \in \mathbb{R}^p} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \tag{7}$$

  where $\lambda \geqslant 0$ is a given regularization parameter and $\|\beta\|_1 = \sum_i^n |x_i|_1$ is the ℓ^1-norm of β.

- This problem is referred to as the **Lasso** problem.
- **ADMM** form:

$$\begin{aligned} \min_{\beta, \alpha \in \mathbb{R}^p} \quad & \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\alpha\|_1 \\ \text{subject to} \quad & \beta - \alpha = 0, \end{aligned}$$

Least Absolute Shrinkage and Selection Operator (Lasso)

- **ADMM** steps:

1. $\beta^{k+1} := \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}\|\beta - \alpha^k + u^k\|_2^2\right)$ %β-minimization
2. $\alpha^{k+1} := \underset{\alpha}{\operatorname{argmin}} \left(\lambda\|\alpha\|_1 + \frac{\rho}{2}\|\beta^{k+1} - \alpha + u^k\|_2^2\right)$ %α-minimization
3. $u^{k+1} := u^k + (\beta^{k+1} - \alpha^{k+1})$ %dual update

Lasso: Steps 1 and 2

Step 1
$f(x) = \frac{1}{2}\|y - X\beta\|_2^2$ is a quadratic, we obtain

$$\beta^{k+1} := \left(X^T X + \rho I\right)^{-1} \left(\rho v^k + X^T y\right)$$

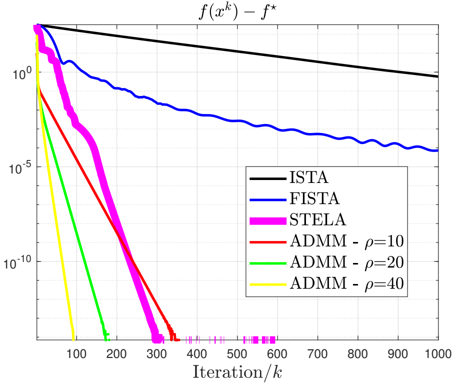
where $v^k = \alpha^k - u^k$ and $X^T X + \rho I$ is always invertible, regardless of X. If we compute a factorization (say Cholesky) in $O(p^3)$ flops, then each β update takes $O(p^2)$ flops (complexity of forward/backward substitution).

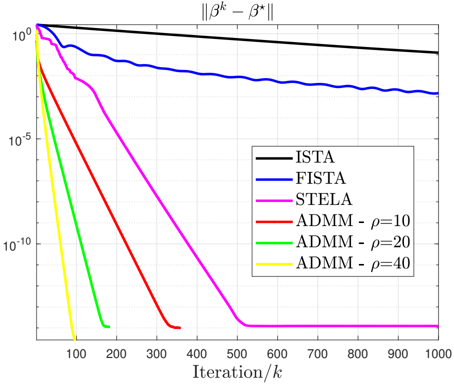
Step 2
The z-update corresponds to the proximal operator associated to the l-1 norm of parameter $\frac{\lambda}{\rho}$, therefore:

$$\begin{aligned} \alpha^{k+1} &:= \mathbf{prox}_{\lambda/\rho, \|.\|_1}(\beta^{k+1} + u^k) \\ &= S_{\lambda/\rho}(\beta^{k+1} + u^k) \end{aligned}$$

Lasso: numerical experiments

Comparison of various algorithms: instances with $n = 1000$, $p = 100$

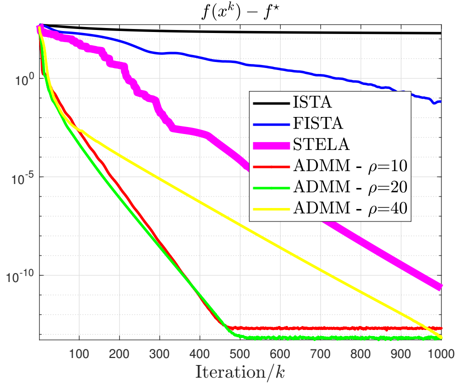

$f(x^k) - f^\star$

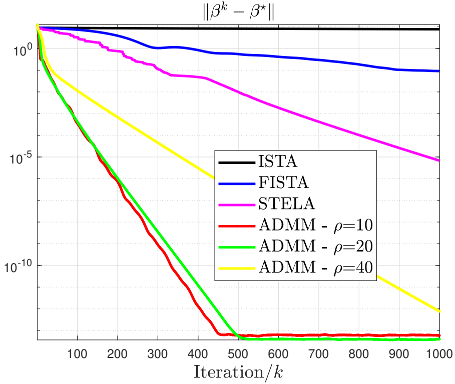

$\|\beta^k - \beta^\star\|$

▸ MatLab Code

Lasso: numerical experiments

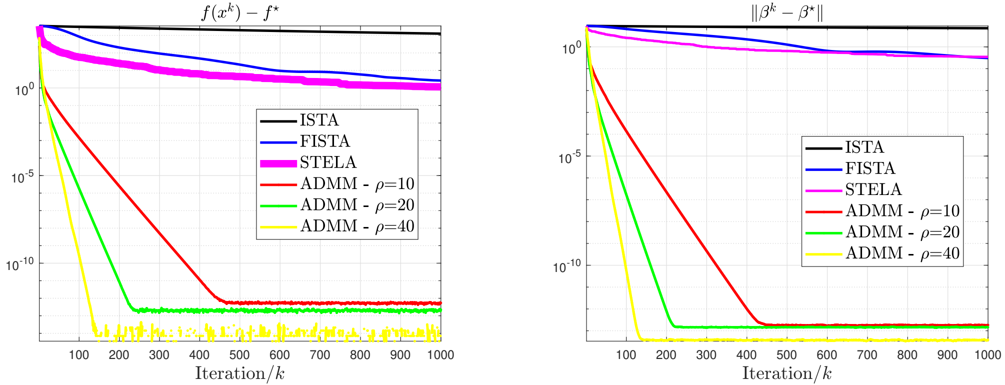
Comparison of various algorithms: instances with $n = 100$, $p = 1000$





Lasso: numerical experiments

Comparison of various algorithms: instances with $n = 1000$, $p = 1000$



Remark: **STELA** algorithm (Yang and Pesavento, 2017) seems more sensible to initialization than other methods. Therefore random and various initializations should be done to compare more accurately and fairly the algorithms.

Lasso: numerical experiments

Remarks:

- Other methods exist for solving Lasso problem such as Coordinate descent-based methods that showed high effectiveness (for small size problems), see (Tibshirani, 2015) - page 11 for a numerical comparison.
- The proper choice of λ is crucial for good performance of this algorithm, but this is not an easy task. Unfortunately we are not in the place here to give you a rule of thumb what to do, since it highly depends on the application at hand. Again, consult (Beck et al 2009) for any further considerations of this matter.

Some many famous examples...

but so little time :)...

- Generalized lasso, revisited
- QP with double-sided constraints,
- Consensus optimization,
- Sparse inverse covariance selection,
- Nonnegative least squares with ℓ^1-regularization,
- Matrix Decompositions:
  1. Sparse low rank decomposition
  2. Nonnegative low rank approximation
  3. Denoising: total variation
  4. Linear Regression with Khatri-Rao structured matrix
- Tensor Decompositions:
  1. Low-rank Tensor Decomposition for Incomplete data
  2. Two-Factor Update Algorithm for CPD
  3. Regularized Matrix-Tensor Factorizations with Linear Couplings

Conclusions

Summary

We have seen

- the motivation behind ADMM: combine advantages from two algorithms: *Dual decomposition* (dual ascent) and *Method of Multipliers* (AKA augmented Lagrangian methods)
- The *problem formulation* *tailored* for ADMM : $\min_{x,z} f(x) + g(z)$ s.t. $Ax + Bz = c$
- The ADMM *steps*:
  - Build the *Augmented Lagrangian* (and the scaled dual form): $L_\rho(x, z, u)$
  - the *iterative scheme*:
    1. x-minimization with z and u fixed: $x^{k+1} = \text{argmin}_x L_\rho(x, z^k, u^k)$
    2. z-minimization with x and u fixed: $z^{k+1} = \text{argmin}_z L_\rho(x^{k+1}, z, u^k)$
    3. dual ascent step with step size ρ: $u^{k+1} := u^k + (Ax^{k+1} + Bz^{k+1} - c)$
- Some common patterns: quadratic objectives, connection to *proximal operators*,
- Applications: alternating projections, NNLS, Lasso, etc.

Appendix A - Proximal operators associated to ℓ^1-norm

When f is the l-1 norm of x: $f(x) = \|x\|_1 = \sum_i^n |x_i|$ (non-differentiable). The proximal operator becomes:

$$\mathbf{prox}_{\lambda,f}(v) := \underset{x}{\operatorname{argmin}} \left(\|x\|_1 + \frac{1}{2\lambda}\|x - v\|_2^2 \right)$$

Recall $\|x\|_2^2 = \sum_i^n x_i^2$, hence the function to minimize is separable w.r.t. each x_i component. Indeed:

$$\mathbf{prox}_{\lambda,f}(v) := \underset{x}{\operatorname{argmin}} \left(\sum_i^n \left(|x_i| + \frac{1}{2\lambda}(x_i - v_i)^2 \right) \right)$$

x-minimization splits into n independent problems:

$$\left[\mathbf{prox}_{\lambda,f}(v)\right]_i := \underset{x_i \in \mathbb{R}}{\operatorname{argmin}} \quad |x_i| + \frac{1}{2\lambda}(x_i - v_i)^2$$

$$\iff \quad := \underset{x_i \in \mathbb{R}}{\operatorname{argmin}} \quad \lambda|x_i| + \frac{1}{2}(x_i - v_i)^2$$

i.e. given v_i, find the minimizer x_i of the scalar function.

Appendix A - Proximal operators associated to ℓ^1-norm

Consider the scalar function:

$$l(x_i) = \lambda |x_i| + \frac{1}{2}(x_i - v_i)^2. \tag{8}$$

It is convex :

1. $|x_i|$ is not differentiable but convex
2. $(x_i - v_i)^2$ is differentiable and convex

The minimizer of f, denoted as $x_i^\star$, can be obtained by considering the first-order optimality condition. As $|x_i|$ is not differentiable, we use the optimality condition for non-smooth functions (no constraints).

Minimum of a non-smooth function

A point $x_i^\star$ is a minimizer of a convex function l if and only if l is subdifferentiable at $x_i^\star$ and

$$0 \in \partial l(x_i^\star). \tag{9}$$

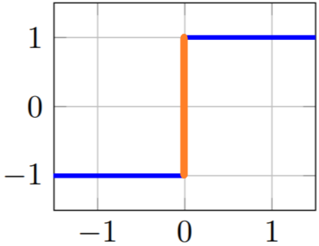
Appendix A - Proximal operators associated to ℓ^1-norm

Few subdifferential calculus rules:

1. if $l(x) = l_1(x) + l_2(x)$, then $\partial l(x) = \partial l_1(x) + \partial l_2(x)$
2. if l_2 is differentiable, then $\partial l_2(x) = \{\nabla l_2(x)\}$

First-order optimality condition for $l(x_i)$:

1. The subdifferential of $|x_i|$ is $\mathrm{sgn}(x_i)$ for $x_i \neq 0$ and $[-1, 1]$ for $x_i = 0$:



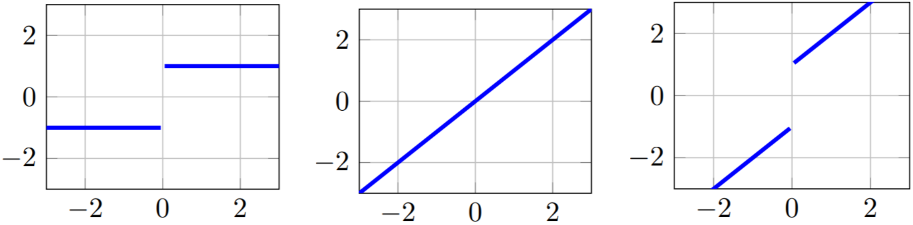
2. The gradient of $\frac{1}{2}(x_i - v_i)^2$ is $(x_i - v_i)$.

Hence (to ease the reading, we consider the case $x_i \neq 0$):

$$0 \in \partial l(x_i^\star) \iff 0 \in \lambda \mathrm{sgn}(x_i^\star) + (x_i^\star - v_i) \iff v_i = x_i^\star + \lambda \mathrm{sgn}(x_i^\star)$$

Appendix A - Proximal operators associated to ℓ^1-norm

Let us know express $x_i^\star$ as a function of v_i. For ease of notation, we drop the subscript i in the **two** following slides. This can be done by swapping the xv axes of the plot of $v = x + \lambda \text{sgn}(x)$ (or more formally, computing its inverse).
As a simple illustration for the case $\lambda = 1$:



(a) $\text{sgn}(x)$ (b) x (c) $x + \text{sgn}(x)$

Appendix A - Proximal operators associated to ℓ^1-norm

Swapping the axes and including the case $x_i = 0$, we get the soft thresholding operator S:

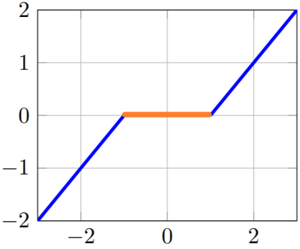


Figure: $S(v) = \text{sgn}(v)\left(|v| - 1\right)_+ = \text{sgn}(v)\max\left(|v| - 1, 0\right)$

In general case with threshold λ, we have:

$$S_\lambda(v) = \text{sgn}(v)\left(|v| - \lambda\right)_+ \tag{10}$$

In MATLAB : sign(v).*(max(abs(v)-lambda,0));

Appendix A - Proximal operators associated to ℓ^1-norm

Finally, the proximal operator [4]:

$$\mathbf{prox}_{\lambda,f}(v) := \underset{x}{\operatorname{argmin}} \left(\|x\|_1 + \frac{1}{2\lambda}\|x - v\|_2^2 \right)$$

can be computed by the following update rule:

$$\mathbf{prox}_{\lambda,f}(v) = S_\lambda(v)$$

where S_λ is the soft thresholding operator applied on v componentwise:

$$\left[\mathbf{prox}_{\lambda,f}(v)\right]_i = \operatorname{sgn}(v_i)\left(|v_i| - \lambda\right)_+$$

Remark: the proximal operator has been computed in closed form !

[4] **Attention**, we are back to the original notations: $x, v \in \mathbb{R}^n$

Goodbye, So Soon

THANKS FOR THE ATTENTION

- v.leplat@innopolis.ru
- sites.google.com/view/valentinleplat/