Convex Optimization in Machine Learning and Inverse Problems

Part 3: Augmented Lagrangian Methods

Mário A. T. Figueiredo[1] and Stephen J. Wright[2]

[1]Instituto de Telecomunicações,
Instituto Superior Técnico, Lisboa, Portugal

[2]Computer Sciences Department,
University of Wisconsin,
Madison, WI, USA

Condensed version of ICCOPT tutorial, Lisbon, Portugal, 2013

Augmented Lagrangian Methods

- Consider a linearly constrained problem,

$$\min\ f(x)\ \text{ s.t. }\ Ax = b.$$

  where f is a proper, lower semi-continuous, convex function.

- The augmented Lagrangian is (with $\rho > 0$)

$$\mathcal{L}(x, \lambda; \rho) := \underbrace{f(x) + \lambda^T(Ax - b)}_{\text{Lagrangian}} + \underbrace{\frac{\rho}{2}\|Ax - b\|_2^2}_{\text{“augmentation”}}$$

Augmented Lagrangian Methods

- Consider a linearly constrained problem,

$$\min f(x) \ \text{ s.t. } \ Ax = b.$$

  where f is a proper, lower semi-continuous, convex function.

- The augmented Lagrangian is (with $\rho > 0$)

$$\mathcal{L}(x, \lambda; \rho) := \underbrace{f(x) + \lambda^T(Ax - b)}_{\text{Lagrangian}} + \underbrace{\frac{\rho}{2}\|Ax - b\|_2^2}_{\text{“augmentation”}}$$

- Basic augmented Lagrangian (a.k.a. method of multipliers) is

$$x_k = \arg\min_x \mathcal{L}(x, \lambda_{k-1}; \rho);$$
$$\lambda_k = \lambda_{k-1} + \rho(Ax_k - b);$$

(Hestenes, 1969; Powell, 1969)

A Favorite Derivation

...more or less rigorous for convex f.

- Write the problem as

$$\min_x \max_\lambda f(x) + \lambda^T (Ax - b).$$

  Obviously, the max w.r.t. λ will be $+\infty$, unless $Ax = b$, so this is equivalent to the original problem.

A Favorite Derivation

...more or less rigorous for convex f.

- Write the problem as

$$\min_x \max_\lambda f(x) + \lambda^T(Ax - b).$$

  Obviously, the max w.r.t. λ will be $+\infty$, unless $Ax = b$, so this is equivalent to the original problem.

- This equivalence is not very useful, computationally: the $\max_\lambda$ function is highly nonsmooth w.r.t. x. Smooth it by adding a "proximal point" term, penalizing deviations from a prior estimate $\bar{\lambda}$:

$$\min_x \left\{ \max_\lambda f(x) + \lambda^T(Ax - b) - \frac{1}{2\rho}\|\lambda - \bar{\lambda}\|^2 \right\}.$$

A Favorite Derivation

...more or less rigorous for convex f.

- Write the problem as

$$\min_x \max_\lambda f(x) + \lambda^T (Ax - b).$$

  Obviously, the max w.r.t. λ will be $+\infty$, unless $Ax = b$, so this is equivalent to the original problem.

- This equivalence is not very useful, computationally: the $\max_\lambda$ function is highly nonsmooth w.r.t. x. Smooth it by adding a "proximal point" term, penalizing deviations from a prior estimate $\bar{\lambda}$:

$$\min_x \left\{ \max_\lambda f(x) + \lambda^T (Ax - b) - \frac{1}{2\rho} \|\lambda - \bar{\lambda}\|^2 \right\}.$$

- Maximization w.r.t. λ is now trivial (a concave quadratic), yielding

$$\lambda = \bar{\lambda} + \rho(Ax - b).$$

A Favorite Derivation (Cont.)

- Inserting $\lambda = \bar{\lambda} + \rho(Ax - b)$ leads to

$$\min_x f(x) + \bar{\lambda}^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|^2 = \mathcal{L}(x, \bar{\lambda}; \rho).$$

A Favorite Derivation (Cont.)

- Inserting $\lambda = \bar{\lambda} + \rho(Ax - b)$ leads to

$$\min_x f(x) + \bar{\lambda}^T (Ax - b) + \frac{\rho}{2}\|Ax - b\|^2 = \mathcal{L}(x, \bar{\lambda}; \rho).$$

- Hence can view the augmented Lagrangian process as:
  - $\min_x \mathcal{L}(x, \bar{\lambda}; \rho)$ to get new x;
  - Shift the "prior" on λ by updating to the latest max: $\bar{\lambda} + \rho(Ax - b)$.
  - repeat until convergence.

A Favorite Derivation (Cont.)

- Inserting $\lambda = \bar{\lambda} + \rho(Ax - b)$ leads to

$$\min_x f(x) + \bar{\lambda}^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|^2 = \mathcal{L}(x, \bar{\lambda}; \rho).$$

- Hence can view the augmented Lagrangian process as:
  - ◇ $\min_x \mathcal{L}(x, \bar{\lambda}; \rho)$ to get new x;
  - ◇ Shift the "prior" on λ by updating to the latest max: $\bar{\lambda} + \rho(Ax - b)$.
  - ◇ repeat until convergence.

- Add subscripts, and we recover the augmented Lagrangian algorithm of the first slide!

- Can also increase ρ (to sharpen the effect of the prox term), if needed.

Inequality Constraints, Nonlinear Constraints

- The same derivation can be used for inequality constraints:

$$\min f(x) \;\text{ s.t. }\; Ax \geq b.$$

- Apply the same reasoning to the constrained min-max formulation:

$$\min_{x} \max_{\lambda \geq 0} f(x) - \lambda^T (Ax - b).$$

Inequality Constraints, Nonlinear Constraints

- The same derivation can be used for inequality constraints:

$$\min f(x) \ \text{ s.t. } \ Ax \geq b.$$

- Apply the same reasoning to the constrained min-max formulation:

$$\min_x \max_{\lambda \geq 0} f(x) - \lambda^T (Ax - b).$$

- After the prox-term is added, can find the minimizing λ in closed form (as for prox-operators). Leads to update formula:

$$\max\left(\bar{\lambda} + \rho(Ax - b), 0\right).$$

- This derivation extends immediately to nonlinear constraints $c(x) = 0$ or $c(x) \geq 0$.

"Explicit" Constraints, Inequality Constraints

- There may be other constraints on x (such as $x \in \Omega$) that we prefer to handle explicitly in the subproblem.
- For the formulation $\min_x f(x)$, s.t. $Ax = b$, $x \in \Omega$, the $\min_x$ step can enforce $x \in \Omega$ explicitly:

$$x_k = \arg\min_{x \in \Omega} \mathcal{L}(x, \lambda_{k-1}; \rho);$$
$$\lambda_k = \lambda_{k-1} + \rho(Ax_k - b);$$

"Explicit" Constraints, Inequality Constraints

- There may be other constraints on x (such as $x \in \Omega$) that we prefer to handle explicitly in the subproblem.

- For the formulation $\min_x f(x)$, s.t. $Ax = b$, $x \in \Omega$, the $\min_x$ step can enforce $x \in \Omega$ explicitly:

$$x_k = \arg\min_{x \in \Omega} \mathcal{L}(x, \lambda_{k-1}; \rho);$$
$$\lambda_k = \lambda_{k-1} + \rho(Ax_k - b);$$

- This gives an alternative way to handle inequality constraints: introduce slacks s, and enforce them explicitly. That is, replace

$$\min_x f(x) \text{ s.t. } c(x) \geq 0,$$

by

$$\min_x f(x) \text{ s.t. } c(x) = s, \quad s \geq 0.$$

"Explicit" Constraints, Inequality Constraints (Cont.)

- The augmented Lagrangian is now

$$\mathcal{L}(x, s, \lambda; \rho) := f(x) + \lambda^T (c(x) - s) + \frac{\rho}{2} \|c(x) - s\|_2^2.$$

- Enforce $s \geq 0$ explicitly in the subproblem:

$$(x_k, s_k) = \arg\min_{x,s} \mathcal{L}(x, s, \lambda_{k-1}; \rho), \quad \text{s.t.} \quad s \geq 0;$$

$$\lambda_k = \lambda_{k-1} + \rho(c(x_k) - s_k)$$

"Explicit" Constraints, Inequality Constraints (Cont.)

- The augmented Lagrangian is now

$$\mathcal{L}(x, s, \lambda; \rho) := f(x) + \lambda^T (c(x) - s) + \frac{\rho}{2}\|c(x) - s\|_2^2.$$

- Enforce $s \geq 0$ explicitly in the subproblem:

$$(x_k, s_k) = \arg\min_{x,s} \mathcal{L}(x, s, \lambda_{k-1}; \rho), \quad \text{s.t.} \quad s \geq 0;$$
$$\lambda_k = \lambda_{k-1} + \rho(c(x_k) - s_k)$$

- There are good algorithmic options for dealing with bound constraints $s \geq 0$ (gradient projection and its enhancements). This is used in the Lancelot code (Conn et al., 1992).

Quick History of Augmented Lagrangian

- Dates from at least 1969: Hestenes, Powell.
- Developments in 1970s, early 1980s by Rockafellar, Bertsekas, and others.
- Lancelot code for nonlinear programming: Conn, Gould, Toint, around 1992 (Conn et al., 1992).
- Lost favor somewhat as an approach for general nonlinear programming during the next 15 years.
- Recent revival in the context of sparse optimization and its many applications, in conjunction with splitting / coordinate descent.

Alternating Direction Method of Multipliers (ADMM)

- Consider now problems with a separable objective of the form

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

for which the augmented Lagrangian is

$$\mathcal{L}(x, z, \lambda; \rho) := f(x) + h(z) + \lambda^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax - Bz - c\|_2^2.$$

Alternating Direction Method of Multipliers (ADMM)

- Consider now problems with a separable objective of the form

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

for which the augmented Lagrangian is

$$\mathcal{L}(x, z, \lambda; \rho) := f(x) + h(z) + \lambda^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax - Bz - c\|_2^2.$$

- Standard AL would minimize $\mathcal{L}(x, z, \lambda; \rho)$ w.r.t. (x, z) jointly. However, these are coupled in the quadratic term, separability is lost

Alternating Direction Method of Multipliers (ADMM)

- Consider now problems with a separable objective of the form

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

for which the augmented Lagrangian is

$$\mathcal{L}(x, z, \lambda; \rho) := f(x) + h(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2}\|Ax - Bz - c\|_2^2.$$

- Standard AL would minimize $\mathcal{L}(x, z, \lambda; \rho)$ w.r.t. (x, z) jointly. However, these are coupled in the quadratic term, separability is lost

- In ADMM, minimize over x and z separately and sequentially:

$$x_k = \arg\min_x \mathcal{L}(x, z_{k-1}, \lambda_{k-1}; \rho);$$
$$z_k = \arg\min_z \mathcal{L}(x_k, z, \lambda_{k-1}; \rho);$$
$$\lambda_k = \lambda_{k-1} + \rho(Ax_k + Bz_k - c).$$

ADMM

Main features of ADMM:

- Does one cycle of block-coordinate descent in (x, z).
- The minimizations over x and z add only a quadratic term to f and h, respectively. Usually does not alter the cost much.
- Can perform the (x, z) minimizations inexactly.
- Can add explicit (separated) constraints: $x \in \Omega_x$, $z \in \Omega_z$.
- Many (many!) recent applications to compressed sensing, image processing, matrix completion, sparse principal components analysis....

ADMM has a rich collection of antecendents, dating even to the 1950s (operator splitting).

For an comprehensive recent survey, including a diverse collection of machine learning applications, see Boyd et al. (2011).

ADMM: A Simpler Form

- Often, a simpler version is enough: $\min_{(x,z)} f(x) + h(z)$ s.t. $Ax = z$, equivalent to $\min_x f(x) + h(Ax)$, often the one of interest.

ADMM: A Simpler Form

- Often, a simpler version is enough: $\min_{(x,z)} f(x) + h(z)$ s.t. $Ax = z$, equivalent to $\min_x f(x) + h(Ax)$, often the one of interest.

- In this case, the ADMM can be written as

$$
\begin{aligned}
x_k &= \arg\min_x \; f(x) + \tfrac{\rho}{2}\|A\,x - z_{k-1} - d_{k-1}\|_2^2 \\
z_k &= \arg\min_z \; h(z) + \tfrac{\rho}{2}\|A\,x_k - z - d_{k-1}\|_2^2 \\
d_k &= d_{k-1} - (A\,x_k - z_k)
\end{aligned}
$$

the so-called "scaled version" (Boyd et al., 2011).

ADMM: A Simpler Form

- Often, a simpler version is enough: $\min_{(x,z)} f(x) + h(z)$ s.t. $Ax = z$, equivalent to $\min_x f(x) + h(Ax)$, often the one of interest.

- In this case, the ADMM can be written as

$$x_k = \arg\min_x \; f(x) + \frac{\rho}{2}\|A\,x - z_{k-1} - d_{k-1}\|_2^2$$
$$z_k = \arg\min_z \; h(z) + \frac{\rho}{2}\|A\,x_k - z - d_{k-1}\|_2^2$$
$$d_k = d_{k-1} - (A\,x_k - z_k)$$

the so-called "scaled version" (Boyd et al., 2011).

- Updating z_k is a proximity computation: $z_k = \operatorname{prox}_{h/\rho}\big(A\,x_{k-1} - d_{k-1}\big)$

ADMM: A Simpler Form

- Often, a simpler version is enough: $\min_{(x,z)} f(x) + h(z)$ s.t. $Ax = z$,
  equivalent to $\min_x f(x) + h(Ax)$, often the one of interest.

- In this case, the ADMM can be written as

$$
\begin{aligned}
x_k &= \arg\min_x \; f(x) + \tfrac{\rho}{2}\|A\,x - z_{k-1} - d_{k-1}\|_2^2 \\
z_k &= \arg\min_z \; h(z) + \tfrac{\rho}{2}\|A\,x_k - z - d_{k-1}\|_2^2 \\
d_k &= d_{k-1} - (A\,x_k - z_k)
\end{aligned}
$$

  the so-called "scaled version" (Boyd et al., 2011).

- Updating z_k is a proximity computation: $z_k = \mathrm{prox}_{h/\rho}\big(A\,x_{k-1} - d_{k-1}\big)$

- Updating x_k may be hard: if f is quadratic, involves matrix inversion; if f is not quadratic, may be as hard as the original problem.

ADMM: Convergence

- Consider the problem $\min_x f(x) + h(Ax)$, where f and h are lower semi-continuous, proper, convex functions and A has full column rank.

ADMM: Convergence

- Consider the problem $\min_x f(x) + h(Ax)$, where f and h are lower semi-continuous, proper, convex functions and A has full column rank.

- The ADMM algorithm presented in the previous slide converges (for any $\rho > 0$) to a solution x^*, if one exists, otherwise it diverges.

  This is a cornerstone result by Eckstein and Bertsekas (1992).

ADMM: Convergence

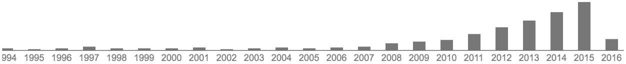
- Consider the problem $\min_x f(x) + h(Ax)$, where f and h are lower semi-continuous, proper, convex functions and A has full column rank.
- The ADMM algorithm presented in the previous slide converges (for any $\rho > 0$) to a solution x^*, if one exists, otherwise it diverges.

  This is a cornerstone result by Eckstein and Bertsekas (1992).
- As in IST/FBS/PGA, convergence is still guaranteed with inexactly solved subproblems, as long as the errors are absolutely summable.

ADMM: Convergence

- Consider the problem $\min_x f(x) + h(Ax)$, where f and h are lower semi-continuous, proper, convex functions and A has full column rank.
- The ADMM algorithm presented in the previous slide converges (for any $\rho > 0$) to a solution x^*, if one exists, otherwise it diverges.

  This is a cornerstone result by Eckstein and Bertsekas (1992).
- As in IST/FBS/PGA, convergence is still guaranteed with inexactly solved subproblems, as long as the errors are absolutely summable.
- The recent explosion of interest in ADMM is quite clear in the citation record of the paper by Eckstein and Bertsekas (1992).



ADMM for a More General Problem

- Consider the problem $\min_{x \in \mathbb{R}^n} \sum_{i=1}^{J} g_j(H^{(j)}x)$, where $H^{(j)} \in \mathbb{R}^{p_j \times n}$, and $g_1, ..., g_J$ are l.s.c proper convex fuctions.

- Map it into $\min_x f(x) + h(Ax)$ as follows (with $p = p1 + \cdots + p_J$):
  - ◇ $f(x) = 0$
  - ◇ $A = \begin{bmatrix} H^{(1)} \\ \vdots \\ H^{(J)} \end{bmatrix} \in \mathbb{R}^{p \times n}$,
  - ◇ $h : \mathbb{R}^{p_1 + \cdots + p_J} \to \bar{\mathbb{R}}, \quad h\left(\begin{bmatrix} z^{(1)} \\ \vdots \\ z^{(J)} \end{bmatrix}\right) = \sum_{j=1}^{J} g_j(z^{(j)})$

- We'll see next that this leads to a very convenient version of ADMM.

ADMM for a More General Problem (Cont.)

Resulting instance of

$$x_k = \arg\min_x \|Ax - z_{k-1} - d_{k-1}\|_2^2$$

ADMM for a More General Problem (Cont.)

Resulting instance of

$$x_k = \arg\min_x \|Ax - z_{k-1} - d_{k-1}\|_2^2 = \left(\sum_{j=1}^{J} (H^{(j)})^T H^{(j)}\right)^{-1} \left(\sum_{j=1}^{J} (H^{(j)})^T (z_{k-1}^{(j)} + d_{k-1}^{(j)})\right)$$

ADMM for a More General Problem (Cont.)

Resulting instance of

$$x_k = \arg\min_x \|Ax - z_{k-1} - d_{k-1}\|_2^2 = \left(\sum_{j=1}^{J} (H^{(j)})^T H^{(j)}\right)^{-1} \left(\sum_{j=1}^{J} (H^{(j)})^T (z_{k-1}^{(j)} + d_{k-1}^{(j)})\right)$$

$$z_k^{(1)} = \arg\min_u g_1 + \frac{\rho}{2}\|u - H^{(1)} x_{k-1} + d_{k-1}^{(1)}\|_2^2$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$z_k^{(J)} = \arg\min_u g_J + \frac{\rho}{2}\|u - H^{(J)} x_{k-1} + d_{k-1}^{(J)}\|_2^2$$

ADMM for a More General Problem (Cont.)

Resulting instance of

$$
\begin{aligned}
x_k &= \arg\min_x \|Ax - z_{k-1} - d_{k-1}\|_2^2 = \Big(\sum_{j=1}^{J} (H^{(j)})^T H^{(j)}\Big)^{-1}\Big(\sum_{j=1}^{J} (H^{(j)})^T (z_{k-1}^{(j)} + d_{k-1}^{(j)})\Big) \\
z_k^{(1)} &= \arg\min_u g_1 + \tfrac{\rho}{2}\|u - H^{(1)}x_{k-1} + d_{k-1}^{(1)}\|_2^2 = \mathrm{prox}_{g_1/\rho}\big(H^{(1)}x_{k-1} - d_{k-1}^{(1)}\big) \\
&\vdots \\
z_k^{(J)} &= \arg\min_u g_J + \tfrac{\rho}{2}\|u - H^{(J)}x_{k-1} + d_{k-1}^{(J)}\|_2^2 = \mathrm{prox}_{g_J/\rho}\big(H^{(J)}x_{k-1} - d_{k-1}^{(J)}\big)
\end{aligned}
$$

ADMM for a More General Problem (Cont.)

Resulting instance of

$$
\begin{aligned}
x_k &= \arg\min_x \|Ax - z_{k-1} - d_{k-1}\|_2^2 = \left(\sum_{j=1}^{J} (H^{(j)})^T H^{(j)}\right)^{-1} \left(\sum_{j=1}^{J} (H^{(j)})^T (z_{k-1}^{(j)} + d_{k-1}^{(j)})\right) \\
z_k^{(1)} &= \arg\min_u g_1 + \tfrac{\rho}{2}\|u - H^{(1)} x_{k-1} + d_{k-1}^{(1)}\|_2^2 = \operatorname{prox}_{g_1/\rho}\big(H^{(1)} x_{k-1} - d_{k-1}^{(1)}\big) \\
\vdots \quad & \qquad \vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\
z_k^{(J)} &= \arg\min_u g_J + \tfrac{\rho}{2}\|u - H^{(J)} x_{k-1} + d_{k-1}^{(J)}\|_2^2 = \operatorname{prox}_{g_J/\rho}\big(H^{(J)} x_{k-1} - d_{k-1}^{(J)}\big) \\
d_k &= d_{k-1} - Ax_k + z_k
\end{aligned}
$$

ADMM for a More General Problem (Cont.)

Resulting instance of

$$
\begin{aligned}
x_k &= \arg\min_x \|Ax - z_{k-1} - d_{k-1}\|_2^2 = \Big(\sum_{j=1}^{J} (H^{(j)})^T H^{(j)}\Big)^{-1} \Big(\sum_{j=1}^{J} (H^{(j)})^T (z_{k-1}^{(j)} + d_{k-1}^{(j)})\Big) \\
z_k^{(1)} &= \arg\min_u g_1 + \tfrac{\rho}{2}\|u - H^{(1)}x_{k-1} + d_{k-1}^{(1)}\|_2^2 = \mathrm{prox}_{g_1/\rho}\big(H^{(1)}x_{k-1} - d_{k-1}^{(1)}\big) \\
\vdots \quad & \qquad \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\
z_k^{(J)} &= \arg\min_u g_J + \tfrac{\rho}{2}\|u - H^{(J)}x_{k-1} + d_{k-1}^{(J)}\|_2^2 = \mathrm{prox}_{g_J/\rho}\big(H^{(J)}x_{k-1} - d_{k-1}^{(J)}\big) \\
d_k &= d_{k-1} - Ax_k + z_k
\end{aligned}
$$

Key features: matrices are handled separately from the prox operators; the prox operators are decoupled (can be computed in parallel); requires a matrix inversion (can be a curse or a blessing).

(Afonso et al., 2010; Setzer et al., 2010; Combettes and Pesquet, 2011)

Example: Image Restoration using SALSA

Problem: $\widehat{\mathbf{x}} \in \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \tau \, \|\mathbf{P}\mathbf{x}\|_1$

Template: $\min_{\mathbf{z} \in \mathbb{R}^d} \sum_{j=1}^{J} g_j(\mathbf{H}^{(j)}\mathbf{z})$

Mapping: $J = 2, \;\; g_1(\mathbf{z}) = \frac{1}{2}\|\mathbf{z} - \mathbf{y}\|_2^2, \qquad g_2(\mathbf{z}) = \tau \, \|\mathbf{z}\|_1$

$$\mathbf{H}^{(1)} = \mathbf{A}, \qquad\qquad \mathbf{H}^{(2)} = \mathbf{P},$$

Convergence conditions: g_1 and g_2 are closed, proper, and convex.

$\mathbf{G} = \begin{bmatrix} \mathbf{A} \\ \mathbf{P} \end{bmatrix}$ has full column rank.

Resulting algorithm: SALSA

(*split augmented Lagrangian shrinkage algorithm*) [Afonso, Bioucas-Dias, F, 2009, 2010]

Example: Image Restoration using SALSA

Key steps of SALSA:

Moreau proximity operator of $g_1(\mathbf{z}) = \frac{1}{2}\|\mathbf{z}-\mathbf{y}\|_2^2,$

$$\text{prox}_{g_1/\mu}(\mathbf{u}) = \arg\min_{\mathbf{z}} \frac{1}{2\mu}\|\mathbf{z}-\mathbf{y}\|_2^2 + \frac{1}{2}\|\mathbf{z}-\mathbf{u}\|_2^2 = \frac{\mathbf{y}+\mu\,\mathbf{u}}{1+\mu}$$

Moreau proximity operator of $g_2(\mathbf{z}) = \tau\|\mathbf{z}\|_1,$

$$\text{prox}_{g_2/\mu}(\mathbf{u}) = \text{soft}\Big(\mathbf{u}, \tau/\mu\Big)$$

Matrix inversion:

$$\mathbf{z}_{k+1} = \left[\mathbf{A}^*\mathbf{A} + \mathbf{P}^*\mathbf{P}\right]^{-1}\left(\mathbf{A}^*\left(\mathbf{u}_k^{(1)} + \mathbf{d}_k^{(1)}\right) + \mathbf{P}^*\left(\mathbf{u}_k^{(2)} + \mathbf{d}_k^{(2)}\right)\right)$$

...next slide!

Example: Image Restoration using SALSA

Problem: $\widehat{\mathbf{x}} \in \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \tau\, \|\mathbf{x}\|_1$

observation matrix

$\mathbf{A} = \mathbf{B}\mathbf{W}$

synthesis matrix

Template: $\min_{\mathbf{z}\in\mathbb{R}^d} \sum_{j=1}^{J} g_j(\mathbf{H}^{(j)}\mathbf{z})$

Mapping: $J = 2$, $\quad g_1(\mathbf{z}) = \frac{1}{2}\|\mathbf{z} - \mathbf{y}\|_2^2, \quad g_2(\mathbf{z}) = \tau\, \|\mathbf{z}\|_1$

$$\mathbf{H}^{(1)} = \mathbf{A} = \mathbf{B}\mathbf{W} \qquad \mathbf{H}^{(2)} = \mathbf{I},$$

Convergence conditions: g_1 and g_2 are closed, proper, and convex.

$$\mathbf{G} = \begin{bmatrix} \mathbf{B}\,\mathbf{W} \\ \mathbf{I} \end{bmatrix}$$ has full column rank.

Example: Image Restoration using SALSA

Frame-based analysis: $\left[\sum_{j=1}^{J} (\mathbf{H}^{(j)})^* \, \mathbf{H}^{(j)}\right]^{-1} = \left[\mathbf{W}^*\mathbf{B}^*\mathbf{B}\mathbf{W} + \mathbf{I}\right]^{-1}$

DFT

Periodic deconvolution: $\mathbf{B} = \mathbf{U}^*\mathbf{D}\mathbf{U}$

diagonal matrix

$O(n \log n)$

$$\left[\mathbf{W}^*\mathbf{B}^*\mathbf{B}\mathbf{W} + \mathbf{I}\right]^{-1} = \mathbf{I} - \mathbf{W}^*\mathbf{U}^*\boxed{\mathbf{D}^*\left[|\mathbf{D}|^2 + \mathbf{I}\right]^{-1}\mathbf{D}}\mathbf{U}\mathbf{W}$$

matrix inversion lemma + $\mathbf{W}\mathbf{W}^* = \mathbf{I}$

subsampling matrix: $\mathbf{M}\mathbf{M}^* = \mathbf{I}$

Compressive imaging (MRI): $\mathbf{B} = \mathbf{M}\mathbf{U}$

$O(n \log n)$

$$\left[\mathbf{W}^*\mathbf{U}^*\mathbf{M}^*\mathbf{M}\mathbf{U}\mathbf{W} + \mathbf{I}\right]^{-1} = \mathbf{I} - \frac{1}{2}\mathbf{W}^*\mathbf{U}^*\mathbf{M}^*\mathbf{M}\mathbf{U}\mathbf{W}$$

subsampling matrix: $\mathbf{S}\mathbf{S}^* = \mathbf{I}$

Inpainting (recovery of lost pixels): $\mathbf{B} = \mathbf{S}$

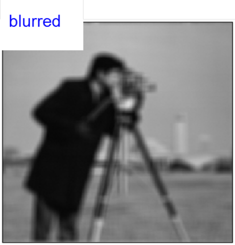
$O(n \log n)$

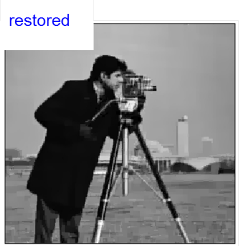
$$\left[\mathbf{W}^*\mathbf{S}^*\mathbf{S}\mathbf{W} + \mathbf{I}\right]^{-1} = \mathbf{I} - \frac{1}{2}\mathbf{W}^*\mathbf{S}^*\mathbf{S}\mathbf{W}^*$$

Example: Image Restoration using SALSA

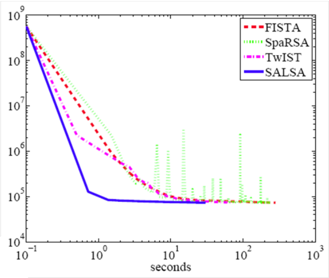
9x9 uniform blur,

40dB BSNR

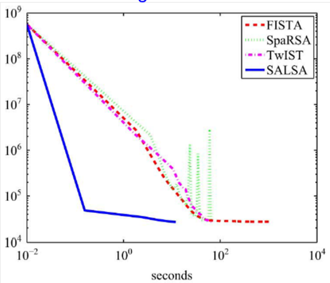




undecimated Haar frame, ℓ_1 regularization.



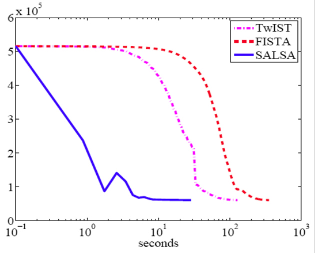
TV regularization



Example: Image Restoration using SALSA

Image inpainting

(50% missing)



Alg.	Calls to $\mathbf{B}, \mathbf{B}^H$	Iter.	CPU time (sec.)	MSE MSE	ISNR (dB)
FISTA	1022	340	263.8	92.01	18.96
TwIST	271	124	112.7	100.92	18.54
SALSA	84	28	20.88	77.61	19.68

Conjecture: SALSA is fast because it's ***blessed*** by the matrix inversion

The inverted matrix (e.g., $\mathbf{A}^*\mathbf{A} + \mathbf{I}$) is (almost) the Hessian of the data term;

...second-order (curvature) information (as Newton's method)

ADMM for the Morozov Formulation

Unconstrained optimization formulation: $\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 + \tau c(\mathbf{x})$

Constrained optimization (Morozov) formulation:

basis pursuit denoising, if $c(\mathbf{x}) = \|\mathbf{x}\|_1$

[Chen, Donoho, Saunders, 1998]

$$\min_{\mathbf{x}} \quad c(\mathbf{x})$$
$$\text{s.t.} \quad \|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 \leq \varepsilon$$

Both analysis and synthesis can be used:

- frame-based analysis, $c(\mathbf{x}) = \|\mathbf{P}\mathbf{x}\|_1$
- frame-based synthesis $c(\mathbf{x}) = \|\mathbf{x}\|_1$

$$\mathbf{A} = \mathbf{B}\,\mathbf{W}$$

ADMM for the Morozov Formulation

Constrained problem: $$\min_{\mathbf{x}} \; c(\mathbf{x}) \quad \text{s.t.} \; \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 \leq \varepsilon$$

...can be written as $$\min_{\mathbf{x}} \; c(\mathbf{x}) + \iota_{\mathcal{B}(\varepsilon,\mathbf{y})}(\mathbf{A}\,\mathbf{x})$$

$$\mathcal{B}(\varepsilon,\mathbf{y}) = \{\mathbf{x} \in \mathbb{R}^n \, : \, \|\mathbf{x} - \mathbf{y}\|_2 \leq \varepsilon\}$$

...which has the form $$\min_{\mathbf{u}\in\mathbb{R}^d} \sum_{j=1}^{J} g_j(\mathbf{H}^{(j)}\mathbf{u}) \qquad (P1)$$

with $J = 2, \quad g_1(\mathbf{z}) = c(\mathbf{z}), \qquad \mathbf{H}^{(1)} = \mathbf{I}$

$g_2(\mathbf{z}) = \iota_{E(\varepsilon,\mathbf{y})}(\mathbf{z}), \qquad \mathbf{H}^{(2)} = \mathbf{A}$

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} \\ \mathbf{A} \end{bmatrix}$$

full column rank

Resulting algorithm: C-SALSA (constrained-SALSA)

[Afonso, Bioucas-Dias, F, 2010,2011]

ADMM for the Morozov Formulation

Moreau proximity operator of $\iota_{\mathcal{B}(\varepsilon,\mathbf{y})}$ is simply a projection on an ℓ_2 ball:

$$\text{prox}_{\iota_{\mathcal{B}(\varepsilon,\mathbf{y})}}(\mathbf{u}) = \arg\min_{\mathbf{z}} \iota_{\mathcal{B}(\varepsilon,\mathbf{y})} + \frac{1}{2}\|\mathbf{z}-\mathbf{u}\|_2^2$$

$$= \begin{cases} \mathbf{u} & \Leftarrow \quad \|\mathbf{u}-\mathbf{y}\|_2 \leq \varepsilon \\ \mathbf{y} + \frac{\varepsilon(\mathbf{u}-\mathbf{y})}{\|\mathbf{u}-\mathbf{y}\|_2} & \Leftarrow \quad \|\mathbf{u}-\mathbf{y}\|_2 > \varepsilon \end{cases}$$

As SALSA, also C-SALSA involves inversion of the form

$$\left[\mathbf{W}^*\mathbf{B}^*\mathbf{B}\mathbf{W} + \mathbf{I}\right]^{-1} \quad \text{or} \quad \left[\mathbf{B}^*\mathbf{B} + \mathbf{P}^*\mathbf{P}\right]^{-1}$$

…all the same tricks as above.

ADMM for the Morozov Formulation

Image deconvolution benchmark problems:

Experiment	blur kernel	σ^2
1	9×9 uniform	0.56^2
2A	Gaussian	2
2B	Gaussian	8
3A	$h_{ij} = 1/(1+i^2+j^2)$	2
3B	$h_{ij} = 1/(1+i^2+j^2)$	8

NESTA: [Becker, Bobin, Candès, 2011]

SPGL1: [van den Berg, Friedlander, 2009]

Frame-synthesis

Expt.	Avg. calls to $\mathbf{B}, \mathbf{B}^H$ (min/max)			Iterations			CPU time (seconds)		
	SPGL1	NESTA	C-SALSA	SPGL1	NESTA	C-SALSA	SPGL1	NESTA	C-SALSA
1	1029 (659/1290)	3520 (3501/3541)	398 (388/406)	340	880	134	441.16	590.79	100.72
2A	511 (279/663)	4897 (4777/4981)	451 (442/460)	160	1224	136	202.67	798.81	98.85
2B	377 (141/532)	3397 (3345/3473)	362 (355/370)	98	849	109	120.50	557.02	81.69
3A	675 (378/772)	2622 (2589/2661)	172 (166/175)	235	656	58	266.41	423.41	42.56
3B	404 (300/475)	2446 (2401/2485)	134 (130/136)	147	551	41	161.17	354.59	29.57

Frame-analysis

Expt.	Avg. calls to $\mathbf{B}, \mathbf{B}^H$ (min/max)		Iterations		CPU time (seconds)	
	NESTA	C-SALSA	NESTA	C-SALSA	NESTA	C-SALSA
1	2881 (2861/2889)	413 (404/419)	720	138	353.88	80.32
2A	2451 (2377/2505)	362 (344/371)	613	109	291.14	62.65
2B	2139 (2065/2197)	290 (278/299)	535	87	254.94	50.14
3A	2203 (2181/2217)	137 (134/143)	551	42	261.89	23.83
3B	1967 (1949/1985)	116 (113/119)	492	39	236.45	22.38

Total-variation

Expt.	Avg. calls to $\mathbf{B}, \mathbf{B}^H$ (min/max)		Iterations		CPU time (seconds)	
	NESTA	C-SALSA	NESTA	C-SALSA	NESTA	C-SALSA
1	7783 (7767/7795)	695 (680/710)	1945	232	311.98	62.56
2A	7323 (7291/7351)	559 (536/578)	1830	150	279.36	38.63
2B	6828 (6775/6883)	299 (269/329)	1707	100	265.35	25.47
3A	6594 (6513/6661)	176 (98/209)	1649	59	250.37	15.08
3B	5514 (5417/5585)	108 (104/110)	1379	37	210.94	9.23

References I

Afonso, M., Bioucas-Dias, J., and Figueiredo, M. (2010). Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19:2345–2356.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Combettes, P. and Pesquet, J.-C. (2011). Signal recovery by proximal forward-backward splitting. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer.

Conn, A., Gould, N., and Toint, P. (1992). *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*. Springer Verlag, Heidelberg.

Eckstein, J. and Bertsekas, D. (1992). On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 5:293–318.

Hestenes, M. (1969). Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320.

Powell, M. (1969). A method for nonlinear constraints in minimization problems. In Fletcher, R., editor, *Optimization*, pages 283–298. Academic Press, New York.

Setzer, S., Steidl, G., and Teuber, T. (2010). Deblurring poissonian images by split bregman techniques. *Journal of Visual Communication and Image Representation*, 21:193–199.