10-725/36-725: Convex Optimization **Fall 2016**

Lecture 21: November 14

Lecturer: Ryan Tibshirani *Scribes: Haichen Li, Mo Li, Xiongtao Ruan*

21.1 Review of dual methods and Augmented Lagrangian method

21.1.1 Dual methods

Consider the problem,

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

where f is strictly convex and closed. The lagrangian of the problem is

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

where u is the dual variable. Then dual gradient ascent repeats, for $k = 1, 2, 3, ...$

$$x^{(k)} = \underset{x}{\text{argmin}} \ L(x, u^{(k-1)})$$

$$u^k = u^{(k-1)} + t_k(Ax^{(k)} - b)$$

Here t_k is the step size in k-th iteration.

In dual method, the primal variable x is updated by minimize the Lagrangian, and the dual variable u is updated by the way of gradient ascent, as $Ax - b$ is the gradient of u.

The advantage of the dual method is that when f is decomposable, the update of x can be decomposed, which means the problem can be easily parallelized or scalable. And the disadvantage is that it need stringent assumptions to ensure convergence, i.e. f must be strong convex, because it is not clear the primal is feasible without the assumptions.

21.1.2 Augmented Lagrangian method

Augmented Lagrangian method is also called method of multipliers. Compared to dual methods, it doesn't change the solution but improves the convergence property. Consider the modified problem, for a parameter $\rho > 0$,

$$\begin{aligned} \min_x \quad & f(x) + \frac{\rho}{2}\|Ax - b\|_2^2 \\ \text{subject to} \quad & Ax = b \end{aligned}$$

The lagrangian of the problem is

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

And the gradient ascent repeats, for $k = 1, 2, 3, ...$

$$x^{(k)} = \underset{x}{\operatorname{argmin}}\ L_\rho(x, u^{(k-1)})$$

$$u^k = u^{(k-1)} + \rho(Ax^{(k)} - b)$$

The good point of the method is that it has better convergence properties than dual method in general. Because the added term adds some smoothness of the problem. And the bad point is that it loses decomposability, because the introduced term $\frac{\rho}{2}\|Ax - b\|_2^2$ is not decomposable.

21.2 Review of Alternating direction method of multipliers (ADMM)

21.2.1 Description of the algorithm

ADMM combines the best of both dual method and augumented Lagrangian method. Consider a problem of the form:

$$\min_x \quad f(x) + g(z) \quad \text{subject to } Ax + Bz = c$$

We define augmented Lagrangian, for a parameter $\rho > 0$,

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

The updates are, for $k = 1, 2, 3...$

$$x^{(k)} = \underset{x}{\operatorname{argmin}}\ L_\rho(x, z^{(k-1)}, u^{(k-1)})$$

$$z^{(k)} = \underset{z}{\operatorname{argmin}}\ L_\rho(x^{(k)}, z, u^{(k-1)})$$

$$u^k = u^{(k-1)} + \rho(Ax^{(k)} + Bz^{(k)} - c)$$

In ADMM, instead of jointly update the primal variables x and z, it sequentially update x, z. That means we use the more recent x i.e. $x^{(k)}$ instead of $x^{(k-1)}$ to update z, and the more recent x, z to update u.

21.2.2 Convergence Guarantees

Under modest assumptions on f, g, where A, B are not required to be full rank, the ADMM iterates satisfy, for any $\rho > 0$:

- Residual convergence: $r^{(k)} = Ax^{(k)} + Bz^{(k)} - c \to 0$ as $k \to \infty$, i.e., primal iterates approach feasibility.
- Objective convergence: $f(x^{(k)}) + g(z^{(k)}) \to f^\star + g^\star$, where $f^\star + g^\star$ is the optimal primal objective value.
- Dual convergence: $u^{(k)} \to u^\star$, where $u^\star$ is a dual solution.

The details for the convergence analysis is in Boyd et al. (2011) [BPCPE11]. And the primal convergence cannot be get generically, but this is true under more assumptions.

Convergence rate of ADMM is not know in general and is currently being developed. Some papers, i.e., Hong and Luo (2012) [HL12], Deng and Yin (2012) [DY16], Iutzeler et al. (2014) [IBCH14] Nishihara et al. (2015) [NLRPJ15] showed some work for the convergence rate. It gets linear convergence when one of the $f(x)$ or $g(x)$ is strongly convex. Roughly, ADMM behaves like a first-order method or a bit better.

21.2.3 ADMM in scaled form

Beside the original ADMM algorithm, we can also express the ADMM algorithm in a scaled form, where the dual variable u is replaced by a scaled variable $w = u/\rho$. After parametrization, the ADMM steps are:

$$\begin{aligned}
x^{(k)} &= \underset{x}{\text{argmin}}\ f(x) + \frac{\rho}{2}\|Ax + Bz^{(k-1)} - c + w^{(k-1)}\|_2^2 \\
z^{(k)} &= \underset{z}{\text{argmin}}\ g(z) + \frac{\rho}{2}\|Ax^{(k)} + Bz - c + w^{(k-1)}\|_2^2 \\
w^k &= w^{(k-1)} + Ax^{(k)} + Bz^{(k)} - c
\end{aligned}$$

Here $w^{(k)}$ can be treated as the running sum of residuals with the expression:

$$w^(k) = w^{(0)} + \sum_{i=1}^{k}(Ax^{(i)} + Bz^{(i)} - c)$$

21.3 Connection to proximal operators

Consider the problem

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

It is equivalent to

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

ADMM steps here are

$$\begin{aligned}
x^{(k)} &= \text{prox}_{f,1/\rho}(z^{k-1} - w^{(k-1)}) \\
z^{(k)} &= \text{prox}_{g,1/\rho}(x^k + w^{(k-1)}) \\
w^k &= w^{(k-1)} + x^{(k)} - z^{(k)}
\end{aligned}$$

Where $\text{prox}_{f,1/\rho}$ and $\text{prox}_{f,1/\rho}$ are the proximal operator at parameter $1/\rho$ for f and g respectively.

The steps can be easily derived as follows:

Let x^+, z^+ and w^+ represented as the one step update of x, z and w. By the general formula of ADMM update, we have:

$$\begin{aligned}
x^+ &= \underset{x}{\text{argmin}}\ f(x) + \frac{\rho}{2}\|x - z + w\|_2^2 \\
&= \underset{x}{\text{argmin}}\ \frac{1}{2 \cdot 1/\rho}\|(z - w) - z\|_2^2 + f(x) \\
&= \text{prox}_{f,1/\rho}(z - w)
\end{aligned}$$

Similarly, we can easily derive

$$\begin{aligned}
z^+ &= \underset{z}{\text{argmin}}\ g(z) + \frac{\rho}{2}\|x^+ - z + w\|_2^2 \\
&= \underset{z}{\text{argmin}}\ \frac{1}{2 \cdot 1/\rho}\|(x^+ + w) - z\|_2^2 + g(z) \\
&= \text{prox}_{g,1/\rho}(x^+ + w)
\end{aligned}$$

And then

$$w^+ = w + x^+ - z^+$$

In general, the update for block of variables reduces to prox update whenever the corresponding linear transformation is the identity.

21.4 Examples: Lasso regression and group lasso Regression

21.4.1 Lasso regression

Given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, recall the lasso problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

In previous classes and homework, we've known how to solve lasso problems in multiple ways such as proximal gradient descent (ISTA), accelerated proximal gradient descent (FISTA), barrier method and primal-dual interior-point method. Like deriving duals, how you set up the auxiliary variables is going to affect the algorithm. Among different choices for how to introduce the auxiliary variables, the following form is in some sense the most efficient option.
We can rewrite the problem as:

$$\min_{\beta,\alpha} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\alpha\|_1 \quad \text{subject to} \quad \beta - \alpha = 0$$

The ADMM updates can be derived as follows:

$$\begin{aligned}
\beta^+ &= \operatorname*{argmin}_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\rho}{2}\|\beta - \alpha + w\|_2^2 && \text{(quadratic in } \beta\text{)} \\
&= (X^T X + \rho I)^{-1}(X^T y + \rho(\alpha - w)) && \text{(set gradient equals to zero and solve for } \beta\text{)} \\
\alpha^+ &= \operatorname*{argmin}_{\alpha} \lambda\|\alpha\|_1 + \frac{\rho}{2}\|\beta^+ - \alpha + w\|_2^2 \\
&= S_{\lambda/\rho}(\beta^+ + w) && \text{(Soft-thresholding of } \beta^+ + w\text{)} \\
w^+ &= w + \beta^+ - \alpha^+
\end{aligned}$$

Notes:

- The matrix $X^T X + \rho I$ 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.
- The α update applies the soft-thresholding operator S_t, defined as

$$[S_t(x)]_j = \begin{cases} x_j - t & x > t \\ 0 & -t \le x \le t, j = 1, \dots p \\ x_j + t & x < -t \end{cases}$$

  which is cheap in $O(p)$.

- ADMM steps are "almost" like repeated soft-thresholding of ridge regression coefficients.
- Different values of ρ can give different results.

In Fig. 21.1, we're comparing the convergence behaviors of various algorithms for lasso regression. One thing to notice is that all these algorithms have the same computational complexity per iteration. What can be seen is that ADMM is actually fairly close to proximal gradient descent for this problem in the sense that it's converging at a very similar rate as compared to proximal gradient descent. Accelerated proximal gradient descent is a little faster with the characteristic "Nesterov ripples". Different values of ρ result in different convergence behaviors. Coordinate descent uses a lot more information of the problem and is much faster than all other methods for this problem. One drawback is that it's not always applicable. More details about coordinate descent will be covered in the upcoming lecture.

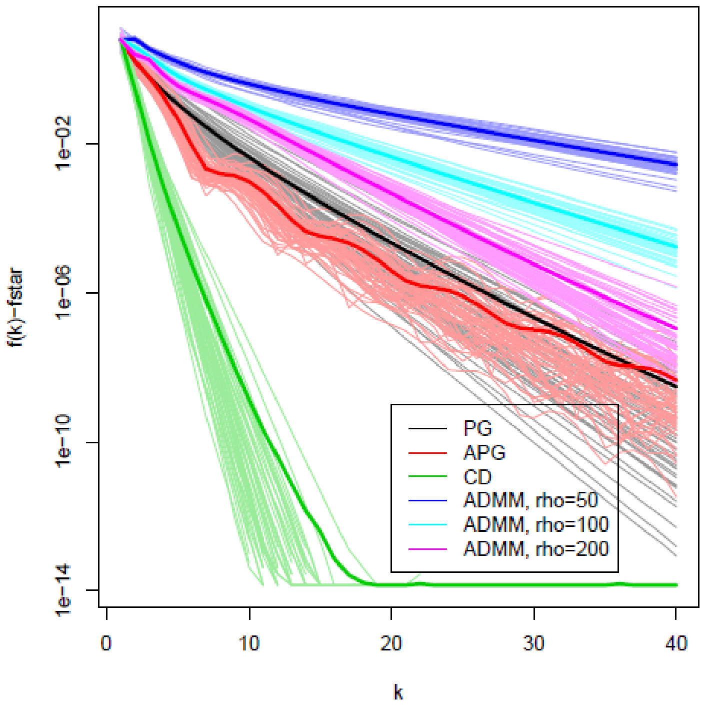

Figure 21.1: Comparison of various algorithms for lasso regression (50 instances with $n = 100$, $p = 20$, the dark lines are the average convergence curves).

In practice, ADMM usually obtains a relatively accurate solution in a handful of iterations, but it requires a large number of iterations for a highly accurate solution (generally behaves like a first-order method). The choice of ρ can greatly influence practical convergence of ADMM. If ρ is too large, there's not enough emphasis on minimizing $f + g$. On the other hand, if ρ is too small, there's not enough emphasis on feasibility. Thus, it's not always a good idea to take a too huge or too tiny ρ in any given problem. Boyd et al. (2011) [BPCPE] give a strategy for varying ρ basically adaptively. At each iteration, you check something like the primal residual and the dual residual, essentially corresponding to how well you are doing on minimizing the criterion and how well you are doing at getting feasibility preliminaries and adjust for ρ accordingly. This strategy can be useful, however, it does not have convergence guarantees. Besides, 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.

21.4.2 Group lasso regression

Given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n\times p}$, recall the group lasso problem:

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

Similar to the lass problem, we can rewrite the problem as

$$\min_{\beta,\alpha} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{g=1}^{G} c_g \|\alpha_{(g)}\|_2 \quad \text{subject to} \quad \beta - \alpha = 0$$

The ADMM steps are

$$\begin{aligned} \beta^+ &= (X^TX + \rho I)^{-1}(X^Ty + \rho(\alpha - w)) && \text{(like the form of ridge regression)} \\ \alpha^+_{(g)} &= R_{c_g\lambda/\rho}(\beta^+_{(g)} + w_{(g)}) \quad g = 1, \dots G \\ w^+ &= w + \beta^+ - \alpha^+ \end{aligned}$$

Notes:

- The matrix $X^TX + \rho I$ 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.
- The α update applies the group soft-thresholding operator R_t, defined as

$$R_t(x) = \left(1 - \frac{t}{\|x\|_2}\right)_+ x$$

- Similar ADMM steps follow for a sum of arbitrary norms of as regularizer, provided we know prox operator of each norm.
- ADMM algorithm can be rederived when groups have overlap, which in general is a very hard problem to optimize. (Boyd et al. (2011) [BPCPE]).

21.5 Examples: Sparse subspace estimation and sparse plus low rank decomposition

21.5.1 Sparse subspace estimation

Given $S = X^TX$ where $X \in \mathbb{R}^{n\times p}$, recall the problem of finding a projection matrix so that the different between the original X and the projected X is as small as possible, which can be formalized as

$$\min_{P} \|X - XP\|_F^2 \quad \text{subject to rank}(P) = k \text{ where } P \text{ is a projection matrix}$$

This problem is a non-convex problem since the space of the projection matrices is not a convex set. As explained previously, it is equivalent to PCA because the solution is to do an eigendecomposition of X^TX

and take the first k eigenvectors.
This non-convex problem, however, is equivalent to the following convex problem

$$\max_{Y} \; tr(SY) \quad \text{subject to} \quad Y \in \mathcal{F}_k = \{Y : 0 \preceq Y \preceq I, tr(Y) = k\}$$

which is often called as the subspace estimation problem.

Vu et al.(2013) [VCLR13] consider the sparse version of subspace estimation problem (a convex problem)

$$\max_{Y} \; tr(SY) - \lambda \|Y\|_1 \quad \text{subject to} \quad Y \in \mathcal{F}_k$$

where $\mathcal{F}_k$ is the Fantope of order k, namely

$$\mathcal{F}_k = \{Y \in \mathbb{S}^p : 0 \preceq Y \preceq I, tr(Y) = k\}$$

Note that when $\lambda = 0$, the above problem is equivalent to ordinary PCA.

The above problem is an SDP and in principle is solvable with interior point methods, though these can be complicated to implement and quite slow for large problem sizes.

To solve the problem in ADMM algorithm, we rewrite the problem as:

$$\min_{Y,Z} -tr(SY) + I_{\mathcal{F}_k}(Y) + \lambda \|Z\|_1 \quad \text{subject to} \quad Y = Z$$

Then the ADMM steps are as follows:

$$\begin{aligned}
Y^+ &= \underset{Y}{\operatorname{argmin}} -tr(SY) + I_{\mathcal{F}_k}(Y) + \frac{\rho}{2}\|Y - Z + W\|_F^2 \\
&= \underset{Y \in \mathcal{F}_k}{\operatorname{argmin}} \frac{1}{2}\|Y - Z + W - S/\rho\|_F^2 \\
&= P_{\mathcal{F}_k}(Z - W + S/\rho) \\
Z^+ &= \underset{Z}{\operatorname{argmin}} \lambda \|Z\|_1 + \frac{\rho}{2}\|Y^+ - Z + W\|_F^2 \\
&= S_{\lambda/\rho}(Y^+ + W) \\
W^+ &= W + Y^+ - Z^+
\end{aligned}$$

Here $P_{\mathcal{F}_k}$ is Fantope projection operator, computed by clipping the eigendecomposition $A = U\Sigma U^T$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p)$:

$$P_{\mathcal{F}_k}(A) = U\Sigma_\theta U^T, \quad \Sigma_\theta = \text{diag}(\sigma_1(\theta), \ldots, \sigma_p(\theta))$$

where each $\sigma_i(\theta) = \min\{\max\{\sigma_i - \theta, 0\}, 1\}$, and $\sum_{i=1}^{p} \sigma_i(\theta) = k$.

21.5.2 Sparse plus low rank decomposition

Given $M \in \mathbb{R}^{n \times m}$, consider the sparse plus low rank decomposition problem (Candes et al., 2009) [CLMW09]:

$$\min_{L,S} \|L\|_{tr} + \lambda \|S\|_1 \quad \text{subject to} \quad L + S = M$$

Here, the goal of the problem is to decompose the observed matrix M into a low rank matrix L and a sparse matrix S. In the above formed convex optimization problem, the trace penalty on L, the sum of the singular values of matrix L, forces a convex relaxation of the rank function applied on L, and the ℓ_1 norm on S induces sparsity of matrix S. λ is the tuning parameter to trade off the importance of low rankness of L and the sparsity in S. Neither the trace norm nor the ℓ_1 norm is smooth, and trace norm is quite complicated as a penalty. Although this problem is actually a SDP and can be solved using interior point methods, it is extremely expensive and complicated. ADMM, on the other hand, offers a rather simple algorithm, and the update steps are as follows:

$$\begin{aligned} L^+ &= S^{tr}_{1/\rho}(M - S + W) \\ S^+ &= S^{\ell_1}_{\lambda/\rho}(M - L^+ + W) \\ W^+ &= W + M - L^+ - S^+ \end{aligned}$$

here, we use $S^{tr}_{1/\rho}$ for matrix soft-thresholding and $S^{\ell_1}_{\lambda/\rho}$ for elementwise soft-thresholding.

Figure 21.2 shows a neat application of sparse plus low rank decomposition in parsing surveillance video clips. From many video frames coming from a surveillance camera, it is possible to decompose each frame into a low rank part, which encodes commonalities shared between all frames, and a sparse part, which carries characteristic features corresponding to this particular frame. For example, three original frames are shown on the left column in Figure 21.2, and after the decomposition, the corresponding low rank parts are shown in the middle column, and the sparse parts are shown in the right column. While the low rank parts are composed of background information in all of the frames, each of the sparse parts captures the appearance of noticeable figures that are specific to the frame. Across a long run of surveillance camera, the appearance of certain figures are clearly sparse and so may be extracted out using this technique.

21.6 Consensus ADMM

21.6.1 Consensus ADMM

Consider a problem of the form:

$$\min_x \sum_{i=1}^{B} f_i(x)$$

To formulate the corresponding ADMM updates we must introduce constraints, in a way that the updates are easy to parallel. The *consensus ADMM* approach begins by reparametrizing:

$$\min_{x_1,\ldots x_B,x} \sum_{i=1}^{B} f_i(x_i) \quad \text{subject to} \quad x_i = x, \; i = 1,\ldots B$$

This yields the *decomposable* ADMM steps:

$$\begin{aligned} x_i^{(k)} &= \operatorname*{argmin}_{x_i} f_i(x_i) + \frac{\rho}{2}\|x_i - x^{(k-1)} + w_i^{(k-1)}\|_2^2, \quad i = 1,\ldots B && \text{(parallel)} \\ x^{(k)} &= \frac{1}{B}\sum_{i=1}^{B}\left(x_i^{(k)} + w_i^{(k-1)}\right) && \text{(simple average)} \\ w_i^{(k)} &= w_i^{(k-1)} + x_i^{(k)} - x^{(k)}, \quad i = 1,\ldots B && \text{(as usual)} \end{aligned}$$

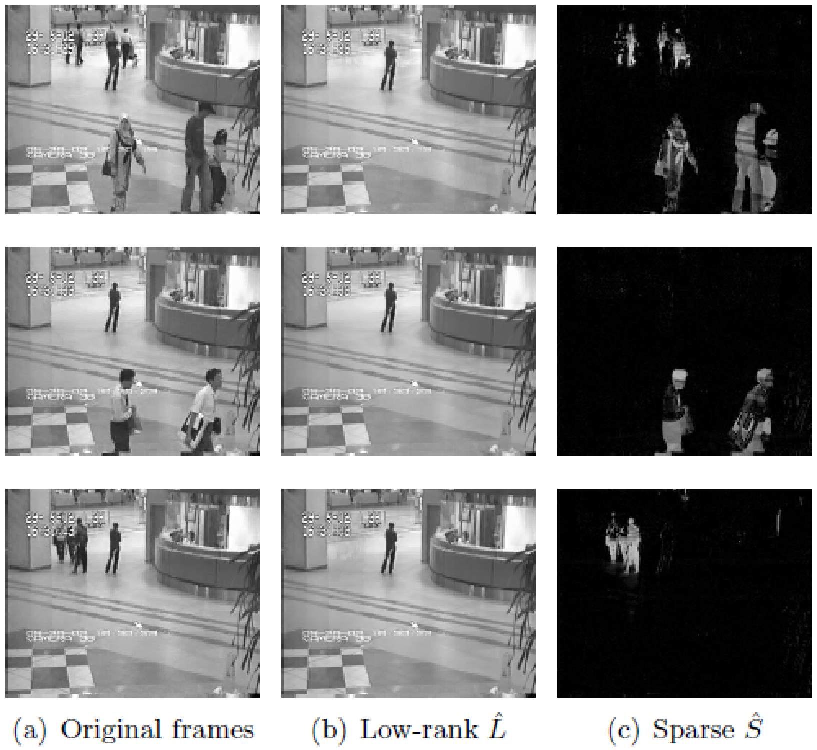

Figure 21.2: Applications of sparse plus low rank decomposition on video frames (blind deconvolution) [CLMW09]

An additional trick is to write $\bar{x} = \frac{1}{B}\sum_{i=1}^{B} x_i$ and $\bar{w} = \frac{1}{B}\sum_{i=1}^{B} w_i$. It is not hard to see that $\bar{w}^{(k)} = 0$ for all iterations $k \geq 1$. As a result, the second ADMM update is equivalent to $x^{(k)} = \bar{x}^{(k)}$, and hence ADMM steps can be simplified into:

$$\begin{aligned}
x_i^{(k)} &= \underset{x_i}{\operatorname{argmin}}\, f_i(x_i) + \frac{\rho}{2}\|x_i - \bar{x}^{(k-1)} + w_i^{(k-1)}\|_2^2, \quad i = 1, \ldots B, \\
w_i^{(k)} &= w_i^{(k-1)} + x_i^{(k)} - \bar{x}^{(k)}, \quad i = 1, \ldots B.
\end{aligned}$$

To reiterate, the $x_i,\ i = 1, \ldots B$ updates here are done in *parallel*

The intuition behind consensus ADMM is the following. It tries to minimize each $f_i(x_i)$, use (squared) ℓ_2 regularization to pull each x_i towards the average $\bar{x}$. If a variable x_i is bigger than the average, then w_i is increased. So the regularization in the next step pulls x_i even closer.

21.6.2 General consensus ADMM with regularization

Consensus ADMM can actually be made more general, for problems with an affine transform on x and an arbitrary regularizer g. Consider a problem of the form:

$$\min_x \sum_{i=1}^{B} f_i(a_i^T x + b_i) + g(x)$$

For *consensus ADMM*, we again reparametrize:

$$\min_{x_1,\dots x_B,x} \sum_{i=1}^{B} f_i(a_i^T x + b_i) + g(x) \ \text{ subject to } \ x_i = x,\ i = 1,\dots B$$

This yields the *decomposable* ADMM updates:

$$\begin{aligned} x_i^{(k)} &= \underset{x_i}{\operatorname{argmin}}\, f_i(a_i^T x + b_i) + \frac{\rho}{2}\|x_i - x^{(k-1)} + w_i^{(k-1)}\|_2^2, \quad i = 1,\dots B \\ x^{(k)} &= \underset{x}{\operatorname{argmin}}\, \frac{B\rho}{2}\|x - \bar{x}^{(k)} - \bar{w}^{(k-1)}\|_2^2 + g(x) \\ w_i^{(k)} &= w_i^{(k-1)} + x_i^{(k)} - x^{(k)}, \quad i = 1,\dots B \end{aligned}$$

Properties of generalized consensus ADMM and its difference compared with consensus ADMM may be summarized as the following. First, it is no longer true that $\bar{w}^{(k)} = 0$ at a general iteration k, and unfortunately ADMM steps do not simplify as before. However, the $x_i,\ i = 1,\dots B$ updates are still done in parallel. Second, for generalized consensus ADMM, each x_i update can be thought of as a loss minimization on part of the data, with ℓ_2 regularization. The x update is a proximal operation in regularizer g. The w update drives the individual variables into consensus. Last, a different initial reparametrization will give rise to a different ADMM algorithm.

Please see Boyd et al. (2011) [BPCPE11], Parikh and Boyd (2013) [NB13] for more details on consensus ADMM, strategies for splitting up into subproblems, and implementation tips.

21.7 Faster convergence with subprogram parametrization: example of the 2d fused lasso problem

An interesting property of ADMM is that it can exhibit much faster convergence than usual, when we parametrize subproblems in a "special way". ADMM updates relate closely to block coordinate descent, in which we optimize a criterion in an alternating fashion across blocks of variables. With this in mind, it is possible to get significantly faster convergence when minimizing over blocks of variables leads to updates in nearly orthogonal directions. As being shown in the following example, ADMM form (auxiliary constraints) can be designed in a way that the primal updates de-correlate as best as possible. This is done in, e.g., Ramdas and Tibshirani (2014) [RT14], Wytock et al. (2014) [WSK14], Barbero and Sra (2014) [BS14].

21.7.1 Problem statement

Given an image $Y \in \mathbb{R}^{d\times d}$, recall the *2d fused lasso* or *2d total variation denoising* problem, in its matrix form:

$$\min_{\Theta} \frac{1}{2}\|Y - \Theta\|_F^2 + \lambda \sum_{i,j} \Big(|\Theta_{i,j} - \Theta_{i+1,j}| + |\Theta_{i,j} - \Theta_{i,j+1}| \Big),$$

where we have a parameter for each pixel in the image and the parameter matrix is represented as $\Theta \in \mathbb{R}^{d\times d}$. Figure 21.3 illustrates the penalty terms in this minimization problem, where every pair of horizontally or vertically neighboring pixels introduces a penalty. The intuition behind is that a lot of neighboring pixels should be assigned to the same value, in order to avoid fusions in both the horizontal and vertical directions.

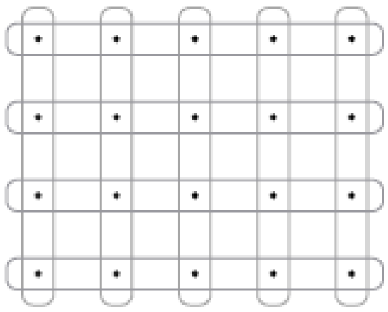

Figure 21.3: Interpretation of the penalty term in 2d fused lasso.

Equivalently, the *2d fused lasso* problem may be stated as the following vector form, with $y \in \mathbb{R}^n$:

$$\min_{\theta} \frac{1}{2}\|y-\theta\|_F^2 + \lambda\|D\theta\|_1.$$

Here $D \in \mathbb{R}^{m\times n}$ is a 2d difference operator giving the appropriate differences (across horizontally and vertically adjacent positions).

21.7.2 Forms of ADMM updates for the 2d fused lasso problem

The first way to setup ADMM is to follow its vector form:

$$\min_{\theta,z} \frac{1}{2}\|y-\theta\|_2^2 + \lambda\|z\|_1 \text{ subject to } z = D\theta,$$

which leads to the following ADMM steps:

$$\begin{aligned}
\theta^{(k)} &= (I+\rho D^TD)^{-1}\big(y+\rho D^T(z^{(k-1)}+w^{(k-1)})\big),\\
z^{(k)} &= S_{\lambda/\rho}(D\theta^{(k)} - w^{(k-1)}),\\
w^{(k)} &= w^{(k-1)} + z^{(k-1)} - D\theta^{(k)}.
\end{aligned}$$

The vector form ADMM updates achieves $O(n)$ time complexity, for the following reasons. On one hand, the θ update solves linear system in $I+\rho L$, with $L = D^TD$ the graph Laplacian matrix of the 2d grid, so this can be done efficiently, in roughly $O(n)$ operations. On the other hand, The z update applies soft thresholding operator S_t. Hence, one entire ADMM cycle uses roughly $O(n)$ operations.

The second way to setup ADMM comes from its matrix form:

$$\begin{aligned}
\min_{\Theta,Z} \quad & \frac{1}{2}\|Y-\Theta\|_F^2 + \lambda\sum_{i,j}\Big(|\Theta_{i,j}-\Theta_{i+1,j}| + |Z_{i,j}-Z_{i,j+1}|\Big)\\
\text{subject to } \quad & \Theta = Z
\end{aligned}$$

which leads to ADMM steps:

$$\begin{aligned}
\Theta^{(k)}_{\cdot,j} &= \mathrm{FL}^{1\mathrm{d}}_{\lambda/(1+\rho)}\left(\frac{Y+\rho(Z^{(k-1)}_{\cdot,j}-W^{(k-1)}_{\cdot,j})}{1+\rho}\right), \quad j=1,\dots,d \\
Z^{(k)}_{i,\cdot} &= \mathrm{FL}^{1\mathrm{d}}_{\lambda/\rho}(\Theta^{(k)}_{i,\cdot}+W^{(k-1)}_{i,\cdot}), \quad i=1,\dots,d \\
W^{(k)} &= W^{(k-1)}+\Theta^{(k)}-Z^{(k-1)}
\end{aligned}$$

The matrix form ADMM updates also achieves $O(n)$ time complexity. Both Θ, Z updates solve (sequence of) 1d fused lassos, where we write $\mathrm{FL}^{1\mathrm{d}}_{\tau}(a) = \mathrm{argmin}_x \frac{1}{2}\|a-x\|_2^2 + \tau\sum_{i=1}^{d-1}|x_i - x_{i+1}|$. On critical observations is that each 1d fused lasso solution can be computed exactly in $O(d)$ operations with specialized algorithms (e.g., Johnson, 2013; Davies and Kovac, 2001 [DK01]). As a result, one entire ADMM cycle again uses $O(n)$ operations.

Figure 21.4 provides a practical view on the matrix form ADMM updates. Note that in the penalty terms, Θ corresponds to all the vertical penalties, and Z corresponds to all the horizontal penalties. The matrix form ADMM update operates on Θ and Z alternatively. In this way, both the Θ update and the Z update can be decomposed into a bunch of separate 1d fused lasso problems (corresponding to rows/columns of the image), where each 1d fused lasso can be solved efficiently in linear time. Therefore, a full ADMM cycle also uses $O(n)$ operations. Interestingly, the Θ update and Z update actually pushes the θ variable in the vector form in opposite directions, and so achieves a much higher convergence rate.

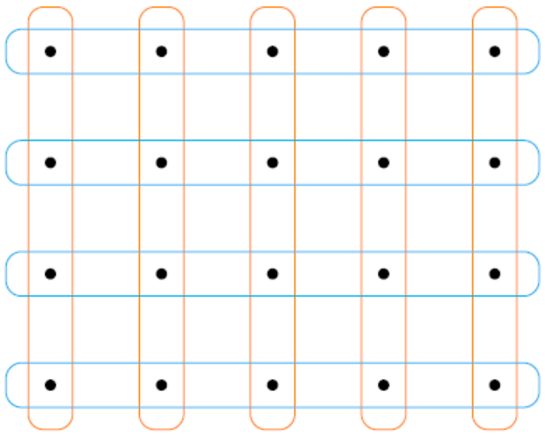

Figure 21.4: Interpretation of the matrix form ADMM updates for 2d fused lasso.

21.7.3 Image denoising experiments

Figure 21.5 shows the image to denoise and the exact solution of the denoised image.

Figure 21.6 shows the difference in terms of convergence properties for the two forms of ADMM updates. The "specialized" ADMM, which decomposes updates into vertical/horizontal strips, achieves a much higher convergence rate compared with the "standard" ADMM.

Figure 21.7 shows the image quality after certain ADMM steps. It is quite obvious that the "specialized" ADMM updates approaches the exact solution much faster, compared with the "standard" ADMM updates.

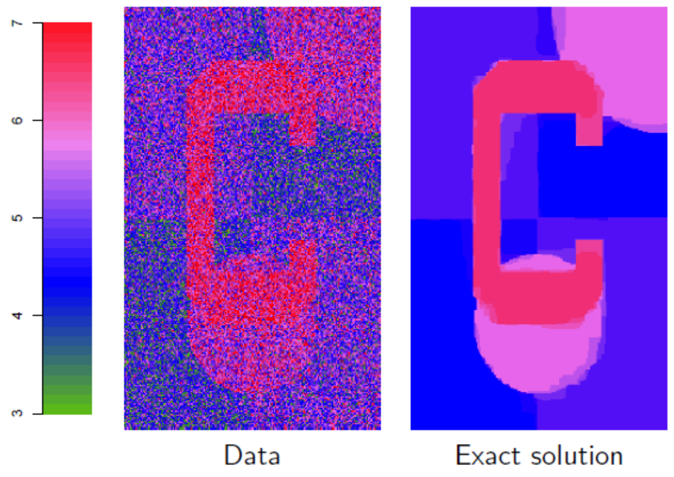

Figure 21.5: Comparison of 2d fused lasso algorithms: an image of dimension 300×200 (so $n = 60,000$). Left: the original image before denoising. Right: the exact solution of the denoised image.

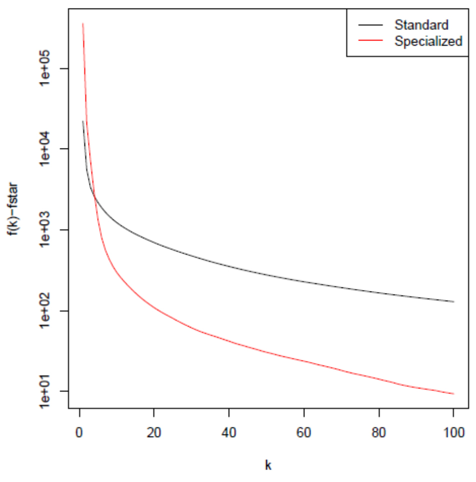

Figure 21.6: Convergence curves of two ADMM algorithms. "standard" corresponds to the vector form ADMM updates, and "specialized" corresponds to the matrix form updates.

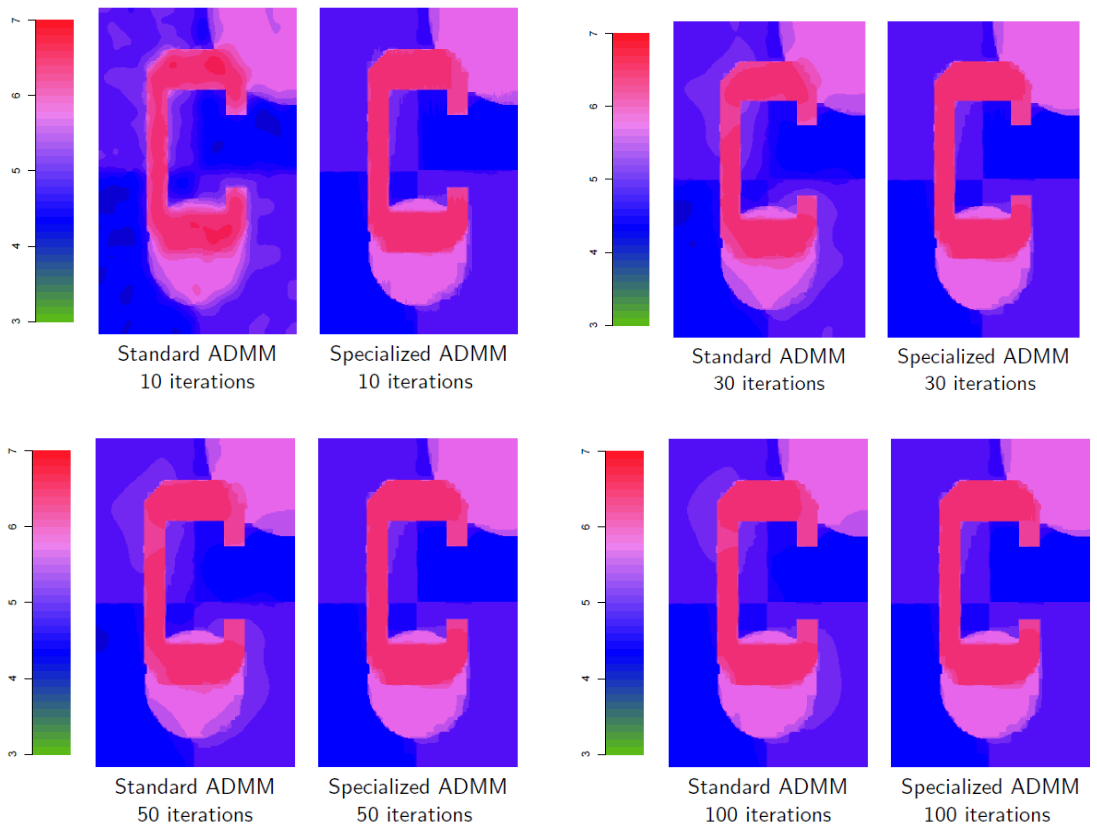

Figure 21.7: ADMM iterates visualized after $k = 10,\ 30,\ 50,\ 100$ iterations.

References

[HL12] M. Hong and Z. Luo, "On the linear convergence of the alternating direction method of multipliers," *arXiv preprint arXiv:1208.3922*, 2012

[DY16] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *Journal of Scientific Computing*, 2016, pp. 889–916

[NLRPJ15] R. Nishihara, L. Lessard, B. Recht, A. Packard and M.I. Jordan, "A General Analysis of the Convergence of ADMM," *arXiv preprint*, 2015

[IBCH14] F. Iutzeler, P. Bianchi, P. Ciblat and W. Hachem, "Linear convergence rate for distributed optimization with the alternating direction method of multipliers," *53rd IEEE Conference on Decision and Control*, 2014, pp. 5046–5051

[CW87] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 1–6.

[DK01] P. L. Davies and A. Kovac, "Local extremes, runs, strings and multiresolution," *Annals of Statistics*, 2001, pp. 1–48.

[BPCPE11] S. Boyd and N. Parikh and E. Chu and B. Peleato and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, 2011, pp. 1-122.

[RT14] A. Ramdas and R. J. Tibshirani, "Fast and flexible admm algorithms for trend filtering," *Journal of Computational and Graphical Statistics*, 2014, just-accepted.

[WSK14] M. Wytock and S. Sra and J. Z. Kolter, "Fast Newton methods for the group fused lasso," *In Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 2014.

[BS14] Á. Barbero and S. Sra, "Modular proximal optimization for multidimensional total-variation regularization," *arXiv preprint*, 2014, arXiv:1411.0589.

[NB13] N. Parikh and S. P. Boyd, "Proximal Algorithms," *Foundations and Trends in optimization*, 2013, 1(3), 127-239.

[VCLR13] V. Vu and J. Cho and J. Lei and K. Rohe, "Fantope projection and selection: a near-optimal convex relaxation of sparse PCA," *Advances in neural information processing systems*, 2013, pp. 2670-2678.

[CLMW09] E. Candes and X. Li and Y. Ma and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, 2009, 58(3), 11.