Lecture 26 — March 9

Lecturer: Emmanuel Candes *Scribe: Hamid Javadi and Emmanuel Candes*

Warning: *These notes may contain factual and/or typographic errors. Some portions of lecture may have been omitted.*

26.1 Overview

In this lecture we will discuss

1. examples of ADMM, and
2. consensus optimization.

Our interest is on parallel solvers that can run on 'big data' problems.

26.2 Solving the Lasso via ADMM

The Lasso problem is given by

$$\text{minimize} \quad \tfrac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1 \tag{26.1}$$

In order to apply ADMM to this problem we rewrite (26.1) as

$$\begin{array}{ll} \text{minimize} & \tfrac{1}{2}\|Ax-b\|_2^2 + \lambda\|z\|_1 \\ \text{subject to} & x - z = 0. \end{array} \tag{26.2}$$

The augmented Lagrangian with penalty parameter $(1/\tau) > 0$ for (26.2) is

$$\mathcal{L}_{\frac{1}{\tau}}(x,z,y) = \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|z\|_1 + \frac{1}{\tau}\langle y, x-z\rangle + \frac{1}{2\tau}\|x-z\|_2^2.$$

Now we derive the update rules of the ADMM for this problem. We have

$$\begin{aligned} x_k &= \arg\min_x \mathcal{L}_{\frac{1}{\tau}}(x, z_{k-1}, y_{k-1}) \\ &= \arg\min_x \left\{ \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|z_{k-1}\|_1 + \frac{1}{\tau}\langle y_{k-1}, x - z_{k-1}\rangle + \frac{1}{2\tau}\|x - z_{k-1}\|_2^2 \right\} \\ &= \arg\min_x \left\{ \frac{1}{2}\left\langle x, \left(A^\mathsf{T}A + \frac{1}{\tau}I\right)x\right\rangle - \left\langle x, A^\mathsf{T}b + \frac{1}{\tau}(z_{k-1} - y_{k-1})\right\rangle \right\} \\ &= \left(A^\mathsf{T}A + \frac{1}{\tau}I\right)^{-1}\left(A^\mathsf{T}b + \frac{1}{\tau}(z_{k-1} - y_{k-1})\right). \end{aligned}$$

We also have

$$\begin{aligned}
z_k &= \arg\min_z \mathcal{L}_{\frac{1}{\tau}}(x_k, z, y_{k-1}) \\
&= \arg\min_z \left\{ \frac{1}{2}\|Ax_k - b\|_2^2 + \lambda\|z\|_1 + \frac{1}{\tau}\langle y_{k-1}, x_k - z\rangle + \frac{1}{2\tau}\|x_k - z\|_2^2 \right\} \\
&= \arg\min_z \left\{ \frac{1}{2\tau}\|x_k + y_{k-1} - z\|_2^2 + \lambda\|z\|_1 \right\} \\
&= \mathrm{S}_{\lambda\tau}(x_k + y_{k-1}).
\end{aligned}$$

Where $\mathrm{S}_{\lambda\tau}$ is the soft-thresholding operator. The dual update rule is

$$y_k = y_{k-1} + \frac{1}{\tau}(x_k - z_k).$$

Again we can see that all the steps can be done very efficiently. The ADMM steps for solving Lasso can be seen in Algorithm 1.

Algorithm 1 ADMM for solving the Lasso problem

$z_0 \leftarrow \tilde{z}$, $y_0 \leftarrow \tilde{y}$, $k \leftarrow 1$ `//initialize`
$\tau \leftarrow \tilde{\tau} > 0$
while convergence criterion is not satisfied **do**
  $x_k \leftarrow \left(A^\top A + \frac{1}{\tau}I\right)^{-1}\left(A^\top b + \frac{1}{\tau}(z_{k-1} - y_{k-1})\right)$
  $z_k \leftarrow \mathrm{S}_{\lambda\tau}(x_k + y_{k-1})$
  $y_k \leftarrow y_{k-1} + \frac{1}{\tau}(x_k - z_k)$
  $k \leftarrow k + 1$
end while

26.3 Consensus optimization [BPC+11]

Consider the problem of the form

$$\text{minimize} \quad \textstyle\sum_{i=1}^N f_i(x), \tag{26.3}$$

where $f_i(x)$ are given convex functions. f_i's can be seen as loss function for the i'th block the training data. In order to apply ADMM we rewrite (26.3) as

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x_i), \\ \text{subject to} & x_i - z = 0. \end{array} \tag{26.4}$$

ADMM can be used to solve (26.4) in parallel. Augmented Lagrangian with penalty parameter $t > 0$ for (26.4) is

$$\mathcal{L}_t(x_i, y_i, z) = \sum_{i=1}^N \left[f_i(x_i) + \langle y_i, x_i - z\rangle + \frac{t}{2}\|x_i - z\|_2^2 \right].$$

Based on this, ADMM steps for solving this problem can be seen in Algorithm 2. The ADMM steps for this problem can be seen as the following

- Solve N independent subproblems in parallel to compute x_i for $i = 1, 2, \ldots, N$.
- Collect computed x_i's in the central unit and update z by averaging.
- Broadcast computed z to N parallel units.
- Update y_i at each unit using the received z.

Algorithm 2 ADMM for consensus optimization

$z^{(0)} \leftarrow \tilde{z}$, $y^{(0)} \leftarrow \tilde{y}$, $k \leftarrow 1$ `//initialize`
$t \leftarrow \tilde{t} > 0$
while convergence criterion is not satisfied **do**
  $x_i^{(k)} \leftarrow \arg\min_{x_i} \left\{ f_i(x_i) + \left\langle y_i^{(k-1)}, x_i - z^{(k-1)} \right\rangle + \frac{t}{2}\|x_i - z^{(k-1)}\|_2^2 \right\}$
  $z^{(k)} \leftarrow \frac{1}{N}\sum_{i=1}^N \left(x_i^{(k)} + \frac{1}{t} y_i^{(k-1)} \right)$
  $y_i^{(k)} \leftarrow y_i^{(k-1)} + t(x_i^{(k)} - z^{(k)})$
  $k \leftarrow k + 1$
end while

Note that the algorithm converges because we are alternating the minimization of the augmented Lagrangian over only two variables. Letting $\mathbf{x}$ be the vector $\{x_i\}_{i=1}^N$, Algorithm 2 is of the form

1. $\mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, z^{(k-1)}; y^{(k-1)})$
2. $z^{(k)} = \arg\min_z \mathcal{L}(\mathbf{x}^{(k)}, z; y^{(k-1)})$
3. $y_i^{(k)} = y_i^{(k-1)} + t(x_i^{(k)} - z^{(k)})$

The point is that the first step (1) decomposes into N independent subproblems, corresponding to the update $x_i^{(k)} \leftarrow \ldots$ for $i = 1, \ldots, N$ in Algorithm 2. Hence, general ADMM theory ensures convergence since there are only 'two blocks'.

26.3.1 Examples

We return to our lasso example and assume we are dealing with a very large problem in the sense that only a small fraction of the data matrix A can be held in fast memory. To see how the ADMM can help in this situation, we can rewrite the residual sum of squares as

$$\|Ax - b\|^2 = \sum_{i=1}^N \|A_i x - b_i\|^2$$

where $A_1, A_2, \ldots, A_N$ is a partition of the rows of the data matrix by cases. One way to reformulate the Lasso problem is this:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N \left\{ \frac{1}{2}\|A_i x_i - b_i\|_2^2 + \lambda_i \|x_i\|_1 \right\} \\ \text{subject to} & x_i = z \quad i = 1, \ldots N, \end{array} \tag{26.5}$$

where $\lambda_i \geq 0$ and $\sum_i \lambda_i = \lambda$. We can now work out the $x_i^{(k)}$ update in Algorithm 2. This update asks for the solution to a (small) Lasso problem of the form

$$\arg\min_{x_i} \left\{ \frac{1}{2}\|C_i x_i - d_i\|^2 + \lambda_i \|x_i\|_1 \right\}$$

where $C_i^\mathsf{T} C_i = A_i^\mathsf{T} A_i + tI$ (this does not change through iterations) and d_i depends on b_i, $z^{(k)}$ and $y_i^{(k-1)}$. Hence, each unit solves a Lasso problem and communicates the result.

A perhaps better way to work is not to separate the ℓ_1 norm and apply ADMM to

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N \frac{1}{2}\|A_i x_i - b_i\|_2^2 + \lambda \|z\|_1 \\ \text{subject to} & x_i = z \quad i = 1, \dots N, \end{array} \tag{26.6}$$

In this case the update for x_i is the solution to a Least-squares problem as we saw in Section 26.2: this asks for the solution to

$$\arg\min_{x_i} \frac{1}{2}\|C_i x_i - d_i\|_2^2$$

where $C_i^\mathsf{T} C_i = A_i^\mathsf{T} A_i + tI$ as before (this does not change through iterations) and d_i depends on b_i, $z^{(k)}$ and $y_i^{(k-1)}$. Then the update for z is of the form

$$z^{(k)} = \mathrm{S}_{\lambda\tau/N}\Big(\mathrm{Ave}_i(x_i^{(k)}) + t^{-1}\mathrm{Ave}(y_i^{(k-1)})\Big).$$

The update for the dual parameter is as in Algorithm 2, namely,

$$y_i^{(k)} \leftarrow y_i^{(k-1)} + t(x_i^{(k)} - z^{(k)}).$$

Bibliography

[BPC+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine Learning **3** (2011), no. 1, 1–122.