Distributed Recovery/Regression/Classification using ADMM

By being very crafty with how we do the splitting, we can use ADMM to solve certain kinds of optimization programs in a distributed manner.

We consider (this material comes from [BPC+10, Sec. 8]) the general problem of "fitting" a vector $\boldsymbol{x} \in \mathbb{R}^N$ to an observed vector $\boldsymbol{b} \in \mathbb{R}^M$ through an $M \times N$ matrix $\boldsymbol{A}$. We will encourage $\boldsymbol{x}$ to have certain structure using a regularizer. This type of problem is ubiquitous in signal processing and machine learning – the math stays the same, only the words change from area to area.

At a high level, we are interested in solving

$$\underset{\boldsymbol{x}}{\text{minimize}} \;\; \text{Loss}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \; + \; \text{Regularizer}(\boldsymbol{x})$$

where the $M \times N$ matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ are given. Notice that

$$\text{Loss}(\cdot) : \mathbb{R}^M \to \mathbb{R}, \quad \text{and} \quad \text{Regularizer}(\cdot) : \mathbb{R}^N \to \mathbb{R}.$$

We will assume that one or both of these functions are separable, at least at the block level. This means we can write

$$\text{Loss}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = \sum_{i=1}^{B} \ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{b}^{(i)}),$$

where $\boldsymbol{A}^{(i)}$ are $M_i \times N$ matrices formed by partitioning the rows of $\boldsymbol{A}$, and $\boldsymbol{b}^{(i)} \in \mathbb{R}^{M_i}$ is the corresponding part of $\boldsymbol{b}$. For separable regularizers, we can write

$$\text{Regularizer}(\boldsymbol{x}) = \sum_{i=1}^{C} r_i(\boldsymbol{x}^{(i)}),$$

where the $\boldsymbol{x}^{(i)} \in \mathbb{R}^{N_i}$ partition the vector $\boldsymbol{x}$. These two types of separability will allow us to divide up the optimization in two different ways.

Example: Inverse Problems and Regression

Two popular methods for solving linear inverse problems and/or calculating regressors are solving

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \tau\|\boldsymbol{x}\|_2^2,$$

(*Tikhonov regularization* or *ridge regression*), and

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \tau\|\boldsymbol{x}\|_1,$$

(*the LASSO*).

These both clearly fit the separability criteria, as

$$\begin{aligned}
\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 &= \sum_{m=1}^{M} (\boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{x} - b[m])^2, \\
\|\boldsymbol{x}\|_2^2 &= \sum_{n=1}^{N} (x[n])^2 \\
\|\boldsymbol{x}\|_1 &= \sum_{n=1}^{N} |x[n]|.
\end{aligned}$$

where $\boldsymbol{a}_m^{\mathrm{T}}$ is the m^{th} row of $\boldsymbol{A}$.

Example: Support Vector Machines

Previously, we saw how if we are given a set of M training examples $(\boldsymbol{x}_m, y_m)$, where $\boldsymbol{x}_m \in \mathbb{R}^N$ and $y_m \in \{-1, 1\}$, we can find a maximum margin linear classifier by solving

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{subject to} \quad y_m(b - \langle \boldsymbol{x}_m, \boldsymbol{w}\rangle) + 1 \leq 0, \ \ m = 1, \ldots, M.$$

With the classifier trained (optimal solution $\boldsymbol{w}^\star, b^\star$ computed), we can assign a label y' to a new point $\boldsymbol{x}'$ using

$$y' = \operatorname{sign}(\langle \boldsymbol{x}', \boldsymbol{w}^\star\rangle + b^\star).$$

Instead of enforcing the constraints above strictly, we can allow some errors by penalizing mis-classifications on the training data appropriately. One reasonable way to do this is make the loss zero if $y_m(b - \langle \boldsymbol{x}_m, \boldsymbol{w}\rangle) + 1 \leq 0$, and then have it increase linearly as this quantity exceeds zero. That is, we solve

$$\min_{\boldsymbol{w},b} \ \sum_{m=1}^{M} \ell(y_m(b - \langle \boldsymbol{x}_m, \boldsymbol{w}\rangle) + 1) + \frac{1}{2}\|\boldsymbol{w}\|_2^2,$$

where $\ell(\cdot)$ is

$$\ell(u) = (u)_+ = \begin{cases} 0, & u \leq 0, \\ u, & u > 0. \end{cases}$$

This is penalty is often called the **hinge loss**. Note that the argument for $\ell(\cdot)$ is an affine function of the optimization variables:

$$y_m(b - \langle \boldsymbol{x}_m, \boldsymbol{w}\rangle) + 1 = \begin{bmatrix} -y_m\boldsymbol{x}_m^{\mathrm{T}} & y_m \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} + 1.$$

Both the loss function and regularizer in this formulation of the SVM are clearly separable.

Splitting across examples

This framework is useful when we have "many measurements of a small vector" or "large volumes of low-dimensional data".

We partition the rows of $\boldsymbol{A}$ and entries of $\boldsymbol{b}$:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \\ \vdots \\ \boldsymbol{A}^{(B)} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}^{(1)} \\ \boldsymbol{b}^{(2)} \\ \vdots \\ \boldsymbol{b}^{(B)} \end{bmatrix}.$$

If the loss function is separable over this partition, our optimization problem is

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \sum_{i=1}^{B} \ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{b}^{(i)}) + r(\boldsymbol{x}),$$

where $r(\cdot)$ is the regularizer. We start by splitting the optimization variables in the loss function and those in the regularizer, arriving at the equivalent program

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \ \sum_{i=1}^{B} \ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x} - \boldsymbol{b}^{(i)}) + r(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{x} - \boldsymbol{z} = \boldsymbol{0}.$$

This does not make the Lagrangian for the primal update separable, as the $\boldsymbol{A}_i$ are still tying together all of the entries in $\boldsymbol{x}$. The trick is to introduce B different vectors $\boldsymbol{x}^{(i)} \in \mathbb{R}^N$, one for each block, and then use the constraints to make them all agree. This is done with

$$\begin{aligned} \underset{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(B)},\boldsymbol{z}}{\text{minimize}} \ & \sum_{i=1}^{B} \ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{b}^{(i)}) + r(\boldsymbol{z}) \\ \text{subject to} \ & \boldsymbol{x}^{(i)} - \boldsymbol{z} = \boldsymbol{0}, \quad i = 1, \ldots, B. \end{aligned}$$

The augmented Lagrangian for this last problem can be expressed as

$$\mathcal{L}_\rho(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(B)}, \boldsymbol{z}, \boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(B)}) = \sum_{i=1}^{B} \mathcal{L}_i(\boldsymbol{x}^{(i)}, \boldsymbol{z}, \boldsymbol{\mu}^{(i)}),$$

where

$$\mathcal{L}_i(\boldsymbol{x}^{(i)}\boldsymbol{z}, \boldsymbol{\mu}^{(i)}) = \ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{b}^{(i)}) + \frac{r(\boldsymbol{z})}{B} + \frac{\rho}{2}\|\boldsymbol{x}^{(i)} - \boldsymbol{z} + \boldsymbol{\mu}^{(i)}\|_2^2$$

and the $\boldsymbol{\mu}^{(i)}$ are the (rescaled) Lagrange multipliers for the constraint $\boldsymbol{x}^{(i)} - \boldsymbol{z} = \boldsymbol{0}$.

As the Lagrangian is separable over the B blocks, each of the primal updates for the $\boldsymbol{x}_i$ can be performed independently. This makes the ADMM iteration

$$\begin{aligned}
\boldsymbol{x}_{k+1}^{(i)} &= \arg\min_{\boldsymbol{x}^{(i)}} \left(\ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{b}^{(i)}) + \frac{\rho}{2}\|\boldsymbol{x}^{(i)} - \boldsymbol{z}_k + \boldsymbol{\mu}_k^{(i)}\|_2^2 \right) \\
\boldsymbol{z}_{k+1} &= \arg\min_{\boldsymbol{z}} \left(r(\boldsymbol{z}) + \frac{\rho}{2}\sum_{i=1}^{B} \|\boldsymbol{z} - \boldsymbol{x}_{k+1}^{(i)} - \boldsymbol{\mu}_k^{(i)}\|_2^2 \right) \\
\boldsymbol{\mu}_{k+1}^{(i)} &= \boldsymbol{\mu}_k^{(i)} + \boldsymbol{x}_{k+1}^{(i)} - \boldsymbol{z}_{k+1}
\end{aligned}$$

The $\boldsymbol{z}$ update can be written in terms of the average of the $\boldsymbol{x}_{k+1}^{(i)}$ and the $\boldsymbol{\mu}_k^{(i)}$. To see this, first note that

$$\begin{aligned}
\sum_{i=1}^{B} \|\boldsymbol{z} - \boldsymbol{v}_i\|_2^2 &= B\|\boldsymbol{z}\|_2^2 - 2\left\langle \boldsymbol{z}, \sum_{i=1}^{B} \boldsymbol{v}_i \right\rangle + \sum_{i=1}^{N} \|\boldsymbol{v}_i\|_2^2 \\
&= B\|\boldsymbol{z}\|_2^2 - 2B\langle \boldsymbol{z}, \bar{\boldsymbol{v}}\rangle + B\|\bar{\boldsymbol{v}}\|_2^2 + \left(-B\|\bar{\boldsymbol{v}}\|_2^2 + \sum_{i=1}^{N} \|\boldsymbol{v}_i\|_2^2 \right) \\
&= B\|\boldsymbol{z} - \bar{\boldsymbol{v}}\|_2^2 + \left(-B\|\bar{\boldsymbol{v}}\|_2^2 + \sum_{i=1}^{N} \|\boldsymbol{v}_i\|_2^2 \right).
\end{aligned}$$

where $\bar{\boldsymbol{v}} = \frac{1}{B}\sum_{i=1}^{B} \boldsymbol{v}_i$. Thus

$$\begin{aligned}
&\arg\min_{\boldsymbol{z}} \left(r(\boldsymbol{z}) + \frac{\rho}{2}\sum_{i=1}^{B} \|\boldsymbol{z} - \boldsymbol{x}_{k+1}^{(i)} - \boldsymbol{\mu}_k^{(i)}\|_2^2 \right) \\
&= \arg\min_{\boldsymbol{z}} \left(r(\boldsymbol{z}) + \frac{B\rho}{2}\|\boldsymbol{z} - \bar{\boldsymbol{x}}_{k+1} - \bar{\boldsymbol{\mu}}_k\|_2^2 \right)
\end{aligned}$$

Distributed ADMM (dividing rows of $\boldsymbol{A}$)

$$\begin{aligned}
\boldsymbol{x}_{k+1}^{(i)} &= \arg\min_{\boldsymbol{x}^{(i)}} \left(\ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{b}^{(i)}) + \frac{\rho}{2}\|\boldsymbol{x}^{(i)} - \boldsymbol{z}_k + \boldsymbol{\mu}_k^{(i)}\|_2^2 \right) \\
\boldsymbol{z}_{k+1} &= \arg\min_{\boldsymbol{z}} \left(r(\boldsymbol{z}) + \frac{B\rho}{2}\|\boldsymbol{z} - \bar{\boldsymbol{x}}_{k+1} - \bar{\boldsymbol{\mu}}_k\|_2^2 \right) \\
\boldsymbol{\mu}_{k+1}^{(i)} &= \boldsymbol{\mu}_k^{(i)} + \boldsymbol{x}_{k+1}^{(i)} - \boldsymbol{z}_{k+1}
\end{aligned}$$

where

$$\bar{\boldsymbol{x}}_{k+1} = \frac{1}{B}\sum_{i=1}^{B} \boldsymbol{x}_{k+1}^{(i)}, \quad \bar{\boldsymbol{\mu}}_k = \frac{1}{B}\sum_{i=1}^{B} \boldsymbol{\mu}_k^{(i)}.$$

The high-level architecture is that B separate units solve independent optimization programs for the B $\boldsymbol{x}^{(i)}$ updates. These are collected and averaged, and a single optimization program is solved to get the $\boldsymbol{z}$ update. The new $\boldsymbol{z}$ is then communicated back to each of the B units. The Lagrange multiplier update can be easily computed both centrally and at the B units, so these do not have to be communicated.

Example: The LASSO

With $\ell_i(\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{b}^{(i)}) = \|\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{b}^{(i)}\|_2^2$ and $r(\boldsymbol{x}) = \tau\|\boldsymbol{x}\|_1$, the ADMM iteration becomes

$$\begin{aligned}
\boldsymbol{x}_{k+1}^{(i)} &= \arg\min_{\boldsymbol{x}^{(i)}} \left(\|\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{b}^{(i)}\|_2^2 + \frac{\rho}{2}\|\boldsymbol{x}^{(i)} - \boldsymbol{z}_k + \boldsymbol{\mu}_k^{(i)}\|_2^2 \right) \\
\boldsymbol{z}_{k+1} &= T_{\tau/(B\rho)}\left(\bar{\boldsymbol{x}}_{k+1} + \bar{\boldsymbol{\mu}}_k\right) \\
\boldsymbol{\mu}_{k+1}^{(i)} &= \boldsymbol{\mu}_k^{(i)} + \boldsymbol{x}_{k+1}^{(i)} - \boldsymbol{z}_{k+1}.
\end{aligned}$$

The $\boldsymbol{x}^{(i)}$ updates are all small unconstrained least-squares problems whose solutions can be computed independently; the $\boldsymbol{z}$ update is a simple soft thresholding, and the $\boldsymbol{\mu}^{(i)}$ and $\bar{\boldsymbol{\mu}}$ updates are computed simply by adding vectors.

Example: SVMs

For the SVM, we collect the weights and the offset into a single optimization vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} \in \mathbb{R}^{N+1}$$

and set

$$\boldsymbol{A} = \begin{bmatrix} -y_1\boldsymbol{x}_1^{\mathrm{T}} & y_1 \\ \vdots & \vdots \\ -y_M\boldsymbol{x}_M^{\mathrm{T}} & y_M. \end{bmatrix}$$

If we partition the data ($\boldsymbol{A}$) into B blocks ($\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(B)}$) then we can express the i^{th} component of the augmented Lagrangian as

$$\mathcal{L}_i(\boldsymbol{x}^{(i)}, \boldsymbol{z}, \boldsymbol{\mu}^{(i)}) = \mathbf{1}^{\mathrm{T}}(\boldsymbol{A}^{(i)}\boldsymbol{x}^{(i)} + \mathbf{1})_+ + \frac{r(\boldsymbol{z})}{B} + \frac{\rho}{2}\|\boldsymbol{x}^{(i)} - \boldsymbol{z} + \boldsymbol{\mu}^{(i)}\|_2^2.$$

Note that the regularization does not include the last term in $\boldsymbol{z}$:

$$r(\boldsymbol{z}) = \frac{1}{2}\sum_{n=1}^{N} |z[n]|^2$$

This results in the ADMM iteration

$$\boldsymbol{x}_{k+1}^{(i)} = \arg\min_{\boldsymbol{x}^{(i)}} \left(\mathbf{1}^{\mathrm{T}} (\boldsymbol{A}^{(i)} \boldsymbol{x}^{(i)} + \mathbf{1})_+ + \frac{\rho}{2} \| \boldsymbol{x}^{(i)} - \boldsymbol{z}_k + \boldsymbol{\mu}_k^{(i)} \|_2^2 \right),$$

$$\boldsymbol{z}_{k+1}[n] = \begin{cases} \frac{B\rho}{1+B\rho} \left(\bar{\boldsymbol{x}}_{k+1}[n] + \bar{\boldsymbol{\mu}}_k[n] \right), & n = 1, \ldots, N, \\ \bar{\boldsymbol{x}}_{k+1}[n] + \bar{\boldsymbol{\mu}}_k[n], & n = N+1, \end{cases}$$

$$\boldsymbol{\mu}_{k+1}^{(i)} = \boldsymbol{\mu}_k^{(i)} + \boldsymbol{x}_{k+1}^{(i)} - \boldsymbol{z}_{k+1}.$$

Splitting across features

Similarly, we can divide up the *columns* of $\boldsymbol{A}$. This is described in [BPC+10, Section 8.3].

References

[BPC+10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.