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 \( \mathbf{x} \in \mathbb{R}^N \) to an observed vector \( \mathbf{b} \in \mathbb{R}^M \) through an \( M \times N \) matrix \( \mathbf{A} \). We will encourage \( \mathbf{x} \) 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

\[
\min_{\mathbf{x}} \quad \text{Loss}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \text{Regularizer}(\mathbf{x})
\]

where the \( M \times N \) matrix \( \mathbf{A} \) and the \( M \)-vector \( \mathbf{b} \) 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

\[
\text{Loss}(\mathbf{A}\mathbf{x} - \mathbf{b}) = \sum_{i=1}^{B} \ell_i(\mathbf{A}_i\mathbf{x} - \mathbf{b}_i),
\]

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

\[
\text{Regularizer}(\mathbf{x}) = \sum_{i=1}^{C} r_i(\mathbf{x}_i),
\]
where the $x_i \in \mathbb{R}^{N_i}$ partition the vector $x$. 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

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + \tau \|x\|_2^2,$$

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

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + \tau \|x\|_1,$$

(*basis pursuit denoising* or *the LASSO*).

These both clearly fit the separability criteria, as

$$\|Ax - b\|_2^2 = \sum_{m=1}^M (\langle x, a_m \rangle - b[m])^2,$$

$$\|x\|_2^2 = \sum_{n=1}^N (x[n])^2$$

$$\|x\|_1 = \sum_{n=1}^N |x[n]|,$$

where $a_m^T$ is the $m$th row of $A$. 
Example: Support Vector Machines

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

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

With the classifier trained (optimal solution $w^*, z^*$ computed), we can assign a label $y'$ to a new point $x'$ using

$$
y' = \text{sign}(\langle x', w^* \rangle + z^*).
$$

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 $y_m(z - \langle x_m, w \rangle) + 1 \leq 0$, and then have it increase linearly as this quantity exceeds zero. That is, we solve

$$
\min_{w,z} \sum_{m=1}^M \ell(y_m(z - \langle x_m, w \rangle) + 1) + \|w\|_2^2,
$$

where $\ell(\cdot)$ is the hinge loss

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

So “soft margin” SVM fits our model as what is inside the $\ell(\cdot)$ can be written as an affine function of the optimization variables:

$$
y_m(z - \langle x_m, w \rangle) + 1 = [ -y_m x_m \ y_m ] \begin{bmatrix} w \\ z \end{bmatrix} + 1.
$$
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 $A$ and entries of $b$:

$$
A = \begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_B
\end{bmatrix}, \quad b = \begin{bmatrix}
b_1 \\
b_2 \\
\vdots \\
b_B
\end{bmatrix}.
$$

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

$$
\min_{\mathbf{x}} \sum_{i=1}^{B} \ell_i(\mathbf{A}_i \mathbf{x} - \mathbf{b}_i) + r(\mathbf{x}),
$$

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

$$
\min_{\mathbf{x}} \sum_{i=1}^{B} \ell_i(\mathbf{A}_i \mathbf{x} - \mathbf{b}_i) + r(\mathbf{z}) \quad \text{subject to} \quad \mathbf{x} - \mathbf{z} = 0.
$$

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

$$
\min_{\mathbf{x}_1, \ldots, \mathbf{x}_B} \sum_{i=1}^{B} \ell_i(\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i) + r(\mathbf{z}) \quad \text{subject to} \quad \mathbf{x}_i - \mathbf{z} = 0, \ i = 1, \ldots, B.
$$
The augmented Lagrangian for this last problem is

\[ L_\rho(x_1, \ldots, x_B, z, \mu_1, \ldots, \mu_B) = \sum_{i=1}^{B} \ell_i(A_ix_i - b_i) + \frac{\rho}{2} \sum_{i=1}^{B} \|x_i - z + \mu_i\|^2 + r(z), \]

where \( \mu_i \) are the (rescaled) Lagrange multipliers for the constraint \( x_i - z = 0 \).

As the Lagrangian is separable over the \( B \) blocks, each of the primal updates for the \( x_i \) can be performed independently. This makes the ADMM iteration

\[ x_i^{(k+1)} = \arg \min_{x_i} \left( \ell_i(A_ix_i - b_i) + \frac{\rho}{2} \|x_i - z^{(k)} + \mu_i^{(k)}\|^2 \right) \]

\[ i = 1, \ldots, B \]

\[ z^{(k+1)} = \arg \min_{z} \left( r(z) + \frac{\rho}{2} \sum_{i=1}^{B} \|z - x_i^{(k+1)} - \mu_i^{(k)}\|^2 \right) \]

\[ \mu_i^{(k+1)} = \mu_i^{(k)} + x_i^{(k+1)} - z^{(k+1)} \]

\[ i = 1, \ldots, B. \]

The \( z \) update can be written in terms of the average of the \( x_i^{(k+1)} \). To see this, first note that

\[ \sum_{i=1}^{B} \|z - v_i\|^2 = B\|z\|^2 - 2 \left( z, \sum_{i=1}^{B} v_i \right) + \sum_{i=1}^{N} \|v_i\|^2 \]

\[ = B\|z\|^2 - 2B \langle z, \bar{v} \rangle + B\|\bar{v}\|^2 + \left( -B\|\bar{v}\|^2 + \sum_{i=1}^{N} \|v_i\|^2 \right) \]

\[ = B\|z - \bar{v}\|^2 + \left( -B\|\bar{v}\|^2 + \sum_{i=1}^{N} \|v_i\|^2 \right). \]
where \( \bar{v} = \frac{1}{B} \sum_{i=1}^{B} v_i \). Thus

\[
\arg \min_z \left( r(z) + \frac{\rho}{2} \sum_{i=1}^{B} \| z - x_i^{(k+1)} - \mu_i^{(k)} \|_2^2 \right)
\]

\[
= \arg \min_z \left( r(z) + \frac{B \rho}{2} \| z - \bar{x}^{(k+1)} - \bar{\mu}^{(k)} \|_2^2 \right)
\]

### Distributed ADMM (dividing rows of \( A \))

\[
x_i^{(k+1)} = \arg \min_{x_i} \left( \ell_i(A_i x_i - b_i) + \frac{\rho}{2} \| x_i - z^{(k)} + \mu_i^{(k)} \|_2^2 \right)
\]

\[
i = 1, \ldots, B
\]

\[
z^{(k+1)} = \arg \min_z \left( r(z) + \frac{B \rho}{2} \| z - \bar{x}^{(k+1)} - \bar{\mu}^{(k)} \|_2^2 \right)
\]

\[
\mu_i^{(k+1)} = \mu_i^{(k)} + x_i^{(k+1)} - z^{(k+1)}
\]

\[
i = 1, \ldots, B.
\]

where

\[
\bar{x}^{(k+1)} = \frac{1}{B} \sum_{i=1}^{B} x_i^{(k+1)}, \quad \bar{\mu}^{(k)} = \frac{1}{B} \sum_{i=1}^{B} \mu_i^{(k)}.
\]

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

**Example: the LASSO**

With \( \ell_i(A_i x_i - b_i) = \| A_i x_i - b_i \|_2^2 \) and \( r(x) = \tau \| x \|_1 \), the ADMM iteration becomes

\[
\begin{align*}
    x_i^{(k+1)} &= \arg \min_{x_i} \left( \| A_i x_i - b_i \|_2^2 + \frac{\rho}{2} \| x_i - z^{(k)} + \mu_i^{(k)} \|_2^2 \right) \\
    i &= 1, \ldots, B \\
    z^{(k+1)} &= T_{\tau/(B \rho)} \left( \bar{x}^{(k+1)} + \bar{\mu}^{(k)} \right) \\
    \mu_i^{(k+1)} &= \mu_i^{(k)} + x_i^{(k+1)} - z^{(k+1)} \\
    i &= 1, \ldots, B.
\end{align*}
\]

The \( x_i \) updates are all small unconstrained least-squares problems whose solutions can be computed independently; the \( z \) update is a simple soft thresholding, and the \( \mu_i \) updates are computed simply by adding vectors.

**Example: SVM**

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

\[
    v = \begin{bmatrix} w \\ z \end{bmatrix} \in \mathbb{R}^{N+1}
\]

and use

\[
    A_i = \begin{bmatrix} -y_1 x_1 & y_1 \\ \vdots & \vdots \\ -y_N x_N & y_N \end{bmatrix}
\]
Note that the regularization does not include the last term in $\mathbf{v}$:

$$r(\mathbf{v}) = \sum_{n=1}^{N} |v[n]|^2.$$ 

This makes the ADMM iteration

$$\mathbf{v}_i^{(k+1)} = \arg \min_{\mathbf{v}_i} \left( \mathbf{1}^T (\mathbf{A}_i \mathbf{v}_i + \mathbf{1})_+ + \frac{\rho}{2} \| \mathbf{v}_i - \mathbf{z}^{(k)} + \mathbf{\mu}_i^{(k)} \|_2^2 \right)$$

$$\mathbf{z}_{1:N}^{(k+1)} = \frac{\rho}{1 + N\rho} \left( \mathbf{\bar{v}}_{1:N}^{(k+1)} + \mathbf{\bar{\mu}}_{1:N}^{(k)} \right)$$

$$\mathbf{z}^{(k+1)}[N + 1] = \mathbf{\bar{v}}^{(k+1)}[N + 1] + \mathbf{\bar{\mu}}^{(k)}[N + 1]$$

$$\mathbf{\mu}_i^{(k+1)} = \mathbf{\mu}_i^{(k)} + \mathbf{v}_i^{(k+1)} - \mathbf{z}^{(k+1)}.$$

where $\mathbf{x}_{1:N}$ is the first $N$ entries of the vector $\mathbf{x}$, and $\mathbf{x}[N + 1]$ is the last entry.

**Splitting across features**

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