ADMM AND DSO

-By Ayush Bhatnagar

Pre-requisites for ADMM

- Dual Ascent
- Dual decomposition
- Augmented Lagrangian

Dual Ascent

- Constraint convex optimization of the form:

$$\min \ f(x)$$
$$subject \ to \ Ax = b$$

- Lagrangian:

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

- Dual function:

$$g(y) = \inf_x L(x, y)$$

- Issue:
  - Slow convergence
  - No distributedness.

Algorithm 1 Dual Ascent

Initialize dual variable: y^0
repeat
    for each iteration **do**
        $x^{(t+1)} \leftarrow arg \min_x L(x, y^t)$
        $\nabla g(y) = Ax^{(t+1)} - b$
        $y^{t+1} \leftarrow y^t + \alpha^t \nabla g(y)$
    end for
until happy

Dual decomposition

- Same constraint convex optimization problem but separable.

$$f(x) = \sum_{i=1}^{G} f_i(x_i) \qquad Ax = \sum_{i=1}^{G} A_i x_i$$

- Lagrangian:

$$L(x,y) = \sum_{i=1}^{G} L_i(x_i, y) = \sum_{i=1}^{G} (f_i(x_i) + y^T A_i x_i - \frac{1}{G} y^T b)$$

- Parallelization is possible.
- Issue:
  - Still slow convergence.

Algorithm 2 Dual Decomposition

```
Initialize dual variable: y^0
repeat
    for each machine i ∈ {1, ..., G} in parallel do
        x_i^(t+1) ← arg min_{x_i} L_i(x_i, y^t)
    end for
    Collect x_i^(t+1) from all machines to make x^(t+1)
    Now compute:
        ∇g(y) = Ax^(t+1) − b
        y^(t+1) ← y^t + α^t ∇g(y)
    Distribute y^(t+1) to all the machines
until happy
```

- xi updates of t+1 iteration are send to a central hub which calculates y(t+1) and then again propagates it to different machines.

Augmented Lagrangian method

- Constraint convex optimization : Updated objective function

$$\min \ f(x) + \frac{\rho}{2} \| Ax - b \|_2^2$$

$$subject\ to\ Ax = b$$

- So the Lagrangian would look like:

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

- Updates would look like:

$$x^{k+1} := \operatorname*{argmin}_x L_\rho(x, y^k)$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} - b),$$

- Issue:
  - Due to this new term we lost decomposability but improved convergence.

ADMM (Alternating Direction Method of multipliers)

- Standard ADMM
- Scaled ADMM

Standard ADMM

- Constraint convex optimization :

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

- Augmented Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2.$$

- **AL** updates would be like:

$$(x^{t+1}, z^{t+1}) := arg \min_{x,z} L(x, z, y^t)$$

$$y^{t+1} := y^t + \rho(Ax^{t+1} + Bz^{t+1} - c)$$

Standard ADMM

- Blend of dual decomposition and augmented Lagrangian method(AL).

- **ADMM** updates would be:

$$x^{t+1} := arg\min_x L(x, z^t, y^t)$$

$$z^{t+1} := arg\min_z L(x^{t+1}, z, y^t)$$

$$y^{t+1} := y^t + \rho(Ax^{t+1} + Bz^{t+1} - c)$$

Scaled ADMM

- Scale the dual variable: u=y/ρ
- The standard ADMM updates would look like:

$$x^{t+1} := arg\min_x \left(f(x) + (\rho/2) \parallel Ax + Bz^t - c + u^t \parallel_2^2 \right)$$

$$z^{t+1} := arg\min_z \left(g(z) + (\rho/2) \parallel Ax^{t+1} + Bz - c + u^t \parallel_2^2 \right)$$

$$u^{t+1} := u^t + Ax^{t+1} + Bz^{t+1} - c$$

- The formulas are shorter in this version.
- This version is widely used.

Least square problem

- Consider the method of least-square where we minimize the sum of square of errors for regression purpose:

$$\min_x \| Ax - Y \|^2$$

- For standard ADMM to work, we will reformulate the problem as:

$$\min_z \| z \|_2^2$$
$$s.t. Ax - z = Y$$

DSO (Distributed Stochastic Optimization)

Regularized Risk Minimization

- RRM :

$$\min_{w \in \mathbb{R}^d} \lambda \sum_j \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(\langle w, x_i \rangle)$$

- Introducing constraints:

$$\min_{w,u} \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(u_i)$$
$$s.t. \quad u_i = \langle w, x_i \rangle \quad \forall i = 1, \ldots, m$$

Lagrangian

- Lagrangian:

$$\min_{w,u} \max_{\alpha} \ \lambda \sum_{j=1}^{d} \phi_j(w_j) + \frac{1}{m} \sum_{i=1}^{m} l_i(u_i) + \frac{1}{m} \sum_{i=1}^{m} \alpha_i(u_i - \langle w, x_i \rangle)$$

- Fenchel-Legendre conjugate :

$$f^*(x) = \max_{y} \ \langle x, y \rangle - f(y) \quad \Rightarrow \quad -l_i^*(-\alpha_i) = \min_{u_i} \alpha_i u_i + l_i(u_i)$$

- Lagrangian can be rewritten as:

$$\min_{w} \max_{\alpha} \ \lambda \sum_{j=1}^{d} \phi_j(w_j) - \frac{1}{m} \sum_{i=1}^{m} \alpha_i \langle w, x_i \rangle - \frac{1}{m} \sum_{i=1}^{m} l_i^*(-\alpha_i)$$

DSO

- Again rewriting the previous equation but only for non-zero features.

$$\min_{w} \max_{\alpha} \sum_{(i,j)\in\Omega} \frac{\lambda\phi_j(w_j)}{|\,\overline{\Omega}_j\,|} - \frac{l_i^*(-\alpha_i)}{m\,|\,\Omega_i\,|} - \frac{\alpha_i w_j x_{ij}}{m}$$

- Now applying stochastic gradient descent for w and ascent for α :

$$w_j \leftarrow w_j - \eta\Big(\frac{\lambda\nabla\phi_j(w_j)}{|\,\overline{\Omega}_j\,|} - \frac{\alpha_i x_{ij}}{m}\Big) \qquad \alpha_i \leftarrow \alpha_i + \eta\Big(\frac{\nabla l_i^*(-\alpha_i)}{m\,|\,\Omega_i\,|} - \frac{w_j x_{ij}}{m}\Big)$$

- Note that we can parallelize this stochastic optimization algorithm.

Algorithm 1 Distributed Stochastic Optimization

Each processor $q \in \{1, 2, \ldots, p\}$ initializes $w^{(q)}, \alpha^{(q)}$

$t \leftarrow 1$

repeat

  $\eta_t \leftarrow \eta_0 / \sqrt{t}$

  for all $r \in \{1, 2, \ldots, p\}$ **do**

    for all processors $q \in \{1, 2, \ldots, p\}$ **in parallel do**

      for (i, j) non zero feature in q^{th} processor **do**

$$w_j \leftarrow w_j - \eta \left(\frac{\lambda \nabla \phi_j(w_j)}{|\bar{\Omega}_j|} - \frac{\alpha_i x_{ij}}{m} \right)$$

$$\alpha_i \leftarrow \alpha_i + \eta \left(\frac{\nabla l_i^*(-\alpha_i)}{m|\Omega_i|} - \frac{w_j x_{ij}}{m} \right)$$

      end for

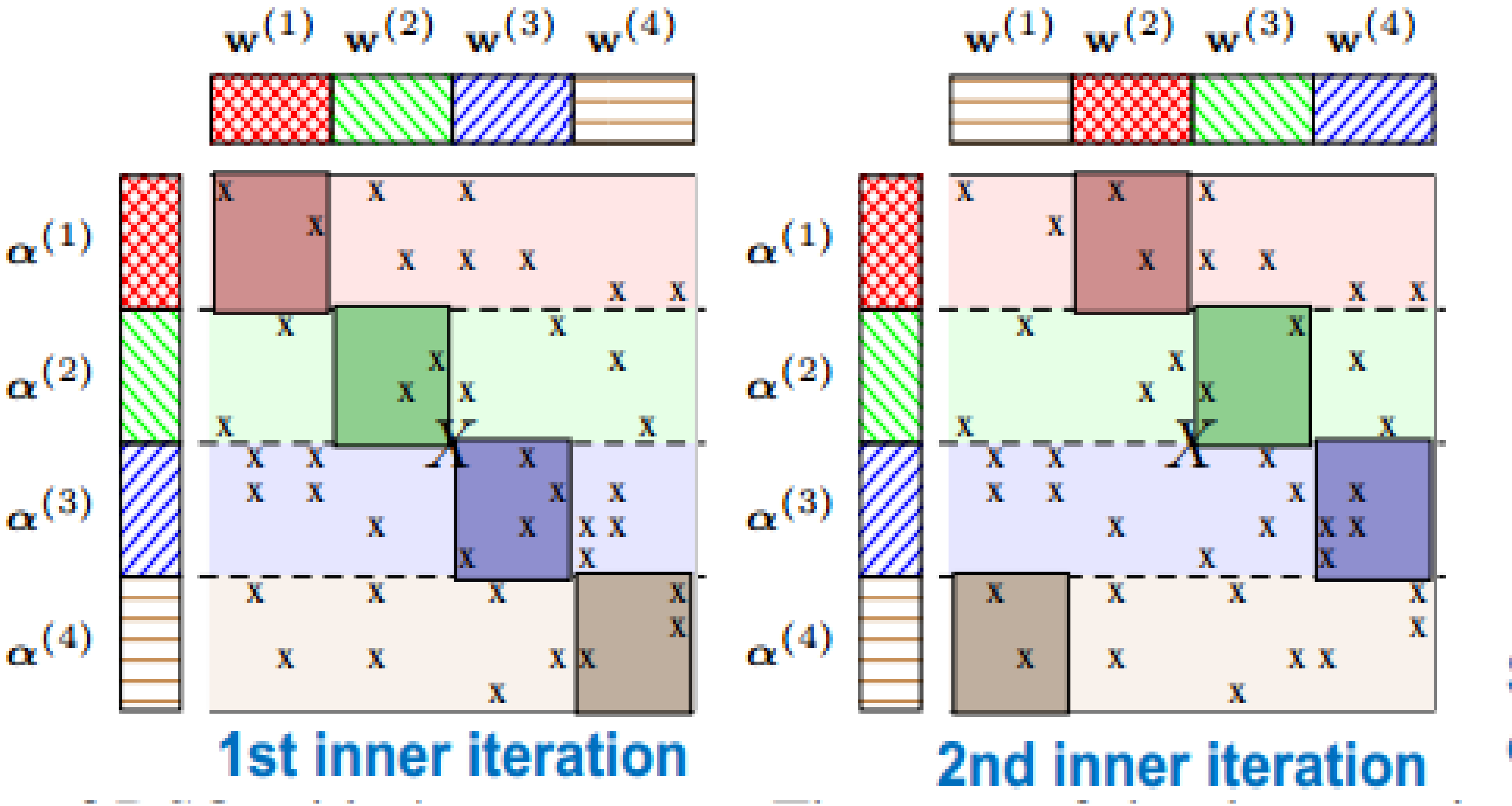
      send these w_j's to next machine and receive from previous.

    end for

  end for

  $t \leftarrow t + 1$

until convergence



Working of DSO

Thank-you ☺