COORDINATE DESCENT ALGORITHMS WITH COUPLING CONSTRAINTS: LESSONS LEARNED

Ahmed Hefny, Sashank Reddi, Suvrit Sra
• Why is this class of algorithms important?

• What is special about these algorithms in terms of software design?

• A proposed framework.
Problem

\[
\min_x f(x) + \sum_{i=1}^{N} h_i(x_i)
\]

s.t. \[\sum_{i}^{N} A_i x_i = 0\]
Problem

\[
\begin{align*}
\min_x f(x) + \sum_{i=1}^{N} h_i(x_i) \\
\text{s.t. } \sum_{i}^{N} A_i x_i = 0
\end{align*}
\]

Why do we care?

- Support Vector Machines
- Consensus Optimization
- Resource Allocation
Random (Block) Coordinate Descent

• Repeat until convergence:
  • Select a pair of blocks $i$ and $j$
  • Fixing $x_k$ for $k \notin \{i, j\}$, find new values for $x_i$ and $x_j$ that minimize a quadratic approximation of the objective function while retaining feasibility.
Random (Block) Coordinate Descent

• Repeat until convergence:
  • Select pair $i$ and $j$
  • Fixing $x_k$ for $k \notin \{i, j\}$, find new values for $x_i$ and $x_j$ that minimize a quadratic approximation of the objective function while retaining feasibility.

Why do we care?
• Can be more efficient than gradient descent methods
• Feasible intermediate solution
Core Operation: Pairwise Update

- \( d_i, d_j = \arg \min_{A_i d_i + A_j d_j = 0} \tilde{f}(x_i + d_i, x_j + d_j) + h_i(x_i + d_i) + h_j(x_j + d_j) \)

- \( x_i \leftarrow x_i + d_i, \quad x_j \leftarrow x_j + d_j \)
Core Operation: Pairwise Update

- \( d_i, d_j = \arg \min_{A_i d_i + A_j d_j = 0} \tilde{f}(x_i + d_i, x_j + d_j) + h_i(x_i + d_i) + h_j(x_j + d_j) \)

- \( x_i \leftarrow x_i + d_i, \ x_j \leftarrow x_j + d_j \)

Updates are incremental.

```
updateFunction(in i, in j, out newXi, out newXj) {
    ...
}
```

//Usage
updateFunction(i, j, xi, xj)
Core Operation: Pairwise Update

• \(d_i, d_j = \arg \min_{A_id_i + A_jd_j = 0} \tilde{f}(x_i + d_i, x_j + d_j) + h_i(x_i + d_i) + h_j(x_j + d_j)\)

• \(x_i \leftarrow x_i + d_i, \ x_j \leftarrow x_j + d_j\)

Updates are incremental.

```
updateFunction(in i, in j, out di, out dj) {
  
  }

//Usage
updateFunction(i, j, di, dj);
x_i = add(x_i, di)
x_j = add(x_j, dj)
```
Core Operation: Pairwise Update

- \( d_i, d_j = \arg \min_{A_i d_i + A_j d_j = 0} \tilde{f}(x_i + d_i, x_j + d_j) + h_i(x_i + d_i) + h_j(x_j + d_j) \)

- \( x_i \leftarrow x_i + d_i, \quad x_j \leftarrow x_j + d_j \)

Requires communication between \( x_i \) and \( x_j \)
Master/Slave Pairwise Update

1. Master node reads required parameters

2. Master node computes **master message**

3. Slave node reads required parameters and receives master message

4. Slave node computes **slave message and slave update**

5. Master node receives slave message

6. Master node computes **master update**
Problem and Environment Adaptation

Generic Framework

Problem Adaptation

Environment Adaptation

Runnable Program

- e.g. for SVM
- e.g. parallel SVM trainer
- e.g. for multicore
Problem and Environment Adaptation

- Parameter Read Client Wrapper (optional)
- Scheduler
- Worker Nodes
- Parameter Read Client
- Parameter Storage
- Parameter Update Client Wrapper (optional)
- Parameter Update Client
THANK YOU