**Primal problem formulation**

\[
\min_{w \in \mathbb{R}^n} \frac{\lambda}{2} |w|^2 + \frac{1}{n} \sum_{i=1}^{n} \ell_i(w^T x_i)
\]

data partitioned by examples \( x_i \) and \( w_i \)

**Dual problem**

\[
\max_{\alpha \in \mathbb{R}^n} -\frac{\lambda}{2} |A\alpha|^2 - \frac{1}{n} \sum_{i=1}^{n} \ell_i^T (-\alpha_i)
\]

Information: local, shared

\( \ell_i^T (\cdot) = \max_{r \in \mathbb{R}} \{ r - \ell_i(\cdot) \} \)

**Main Idea**

Safely and efficiently combine updates from different machines by employing the primal-dual structure

**Algorithm:** Communication-Efficient Coordinate Ascent (CoCoA)

Input: \( T \geq 1 \), scaling \( 1 \leq \gamma_k \leq K \) (default: \( \gamma_k = 1 \)),

Data: \( \{ (x_i, y_i) \}_{i=1}^{n} \) distributed over \( K \) machines

Initialize: \( \alpha_i^{(0)} = 0 \) for all machines \( k \), and \( w_k^{(0)} = 0 \)

for \( t = 1, 2, \ldots, T \)

for all machines \( k = 1, 2, \ldots, K \) in parallel

\[
\begin{align*}
\Delta w_k^{(t)} &= \text{LOCAL-DUAL-METHOD} (\alpha_i^{(t)}, w_k^{(t-1)}) \\
\end{align*}
\]

reduce \( w_k^{(t)} \leftarrow w_k^{(t-1)} + \frac{\Delta w_k^{(t)}}{K} \)

\( R_k = a_k \Delta w_k^{(t)} \)

apply updates immediately

average \( \Delta w_k^{(t)} \) across all machines

**Convergence result**

Linear convergence rate on both inner and outer level

\[
E \left[ D(\alpha^* - D(\alpha^{(T)})) \leq \frac{\lambda \gamma}{1 - (1 - \theta) \frac{1}{1 + \lambda K}} \sum_{i=1}^{T} R_i \right]
\]

\( \theta = \left( 1 - \frac{1}{1 + \lambda K} \right)^n \)

\( R_k = a_k \Delta w_k^{(t)} \)

\( \Delta w_k^{(t)} \) is the update at machine \( k \)

**Applications**

- classic supervised ML
- SVM, Logistic Regr.
- Lasso
- Ridge Regression
- structured SVMs

**Cost of Communication**

Commodity hardware: often 10^8 to 10^9 times slower than reading from local memory

**Related methods**

<table>
<thead>
<tr>
<th>Properties</th>
<th>mini-batch SGD</th>
<th>mini-batch SDCA</th>
<th>Local-SGD</th>
<th>CoCoA</th>
</tr>
</thead>
<tbody>
<tr>
<td>convergence</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>primal-dual</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>apply updates immediately</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>average over ( K )</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Future work**

- optimal scaling between averaging & adding
- similar rates for local-SGD?
- integration into Spark MLlib