CoCoA @ Amazon

Simone Forte

ETH Zurich

fortesi@student.ethz.ch

1 Generalized CoCoA

1.1 Setup

As presented in the CoCoA paper [1] the algorithm currently has support for problems having the following primal form:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \quad \left[\; P(\boldsymbol{w}) := \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) \; \right]. \tag{1}$$

It is though possible, in a very similar theoretical and implementative framework, to solve problems of the following more general form:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \quad \left[\; P(\boldsymbol{w}) := \lambda g(\boldsymbol{w}) + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) \; \right]. \tag{2}$$

where g is a 1-convex function with respect to the L2 norm.

Similarly as for the L2 regularized problem solved by CoCoA, and as done in [2], we can define a dual problem:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad \left[\; D(\boldsymbol{\alpha}) := -\lambda g^\star(\boldsymbol{v}(\boldsymbol{\alpha})) - \frac{1}{n}\sum_{i=1}^{n} \ell_i^*(-\alpha_i) \; \right], \tag{3}$$

where:

$$\boldsymbol{v}(\boldsymbol{\alpha}) = \frac{1}{\lambda n}\sum_{i=1}^{n} X_i\alpha_i \qquad \boldsymbol{w}(\boldsymbol{\alpha}) = \nabla g^\star(\boldsymbol{\alpha}). \tag{4}$$

A primal-dual relation also holds for these two problems, with the duality gap

$$P(\boldsymbol{w}(\boldsymbol{\alpha})) - D(\boldsymbol{\alpha}) \tag{5}$$

acting as an upperbound to the suboptimality of the primal and the dual.

1.2 Method description

In this section we present a generalized CoCoA algorithm to solve this more general problem. The high level structure of the algorithm is very similar (if not almost identical) to the original algorithm and it is as follows:

```
Input: $T \geq 1$,
Data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ distributed over $K$ machines
Initialize: $\boldsymbol{\alpha}_{[k]}^{(0)} \leftarrow \mathbf{0}$ for all machines $k$, and $\boldsymbol{v}^{(0)} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots, T$
    for all machines $k = 1, 2, \ldots, K$ in parallel
        $(\Delta\boldsymbol{\alpha}_{[k]}, \Delta\boldsymbol{v}_k) \leftarrow$ GenLocalDualMethod$(\boldsymbol{\alpha}_{[k]}^{(t-1)}, \boldsymbol{v}^{(t-1)})$
        $\boldsymbol{\alpha}_{[k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[k]}^{(t-1)} + \frac{1}{K}\Delta\boldsymbol{\alpha}_{[k]}$
    end
    reduce $\boldsymbol{v}^{(t)} \leftarrow \boldsymbol{v}^{(t-1)} + \frac{1}{K}\sum_{k=1}^K \Delta\boldsymbol{v}_k$
end
Output: $\left(\boldsymbol{\alpha}^{(T)}, \nabla g^\star(\boldsymbol{v}^{(T)})\right)$
```

The only difference to the original CoCoA is the use of the $\boldsymbol{v}(\boldsymbol{\alpha})$ vectors instead of the $\boldsymbol{w}(\boldsymbol{\alpha})$. This vector is computed at the end, using the gradient of the dual conjugate of the regularizer. Another difference is that the local method is now required to optimize the generalized dual form as in 3 (for its local coordinated) instead of the original CoCoA problem. Same as in [1] we'll also make the following assumption on the local solver:

Assumption 1 (Local Geometric Improvement of GenLocalDualMethod)**.** *We assume that there exists $\Theta \in [0, 1)$ such that for any given $\boldsymbol{\alpha}$,* GenLocalDualMethod *when run on block k alone returns a (possibly random) update $\Delta\boldsymbol{\alpha}_{[k]}$ such that*

$$\mathbf{E}[\epsilon_{D,k}((\boldsymbol{\alpha}_{[1]}, \ldots, \boldsymbol{\alpha}_{[k-1]}, \boldsymbol{\alpha}_{[k]} + \Delta\boldsymbol{\alpha}_{[k]}, \boldsymbol{\alpha}_{[k+1]}, \ldots, \boldsymbol{\alpha}_{[K]}))] \leq \Theta \cdot \epsilon_{D,k}(\boldsymbol{\alpha}). \tag{6}$$

where:

$$\varepsilon_{D,k}(\boldsymbol{\alpha}) := \max_{\hat{\boldsymbol{\alpha}}_{[k]} \in \mathbb{R}^{n_k}} D((\boldsymbol{\alpha}_{[1]}, \ldots, \hat{\boldsymbol{\alpha}}_{[k]}, \ldots, \boldsymbol{\alpha}_{[K]})) - D((\boldsymbol{\alpha}_{[1]}, \ldots, \boldsymbol{\alpha}_{[k]}, \ldots, \boldsymbol{\alpha}_{[K]})). \tag{7}$$

Given this we can state an theorem equivalent to Theorem 2 in [1] (the proof can be found in the appendix):

Theorem 1. *Assume that Algorithm 1 is run for T outer iterations on K worker machines, with the procedure* GenLocalDualMethod *having local geometric improvement Θ. Further, assume the loss functions ℓ_i are $(1/\gamma)$-smooth and that the regularizer* g *is $(1/\delta)$-smooth and μ-strongly convex. Then the following geometric convergence rate holds for the global (dual) objective:*

$$\mathbf{E}[D(\boldsymbol{\alpha}^*) - D(\boldsymbol{\alpha}^{(T)})] \leq \left(1 - (1 - \Theta)\frac{1}{K}\frac{\lambda n \gamma}{\sigma + \lambda n \gamma}\right)^T \left(D(\boldsymbol{\alpha}^*) - D(\boldsymbol{\alpha}^{(0)})\right). \tag{8}$$

Here σ is any real number satisfying

$$\sigma \geq \sigma_{\min} := \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \lambda^2 n^2 \frac{\mu^{-1}\sum_{k=1}^K \|\boldsymbol{v}(\boldsymbol{\alpha}_{[k]})\|^2 - \delta\|\boldsymbol{v}(\boldsymbol{\alpha})\|^2}{\|\boldsymbol{\alpha}\|^2} \geq 0. \tag{9}$$

1.3 Local SDCA

In this section we'll present an SDCA method, very similar to the one in [2] used to solve the generalized local dual problem. The algorithm is as follows and it's also very similar to the

CoCoA one:

```
Input: $H \geq 1$, $\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^{n_k}$, and $\boldsymbol{v} \in \mathbb{R}^d$ consistent with other coordinate blocks of $\boldsymbol{\alpha}$ s.t.
        $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{\alpha})$
Data: Local $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_k}$
Initialize: $\boldsymbol{v}^{(0)} \leftarrow \boldsymbol{v},\ \Delta\boldsymbol{\alpha}_{[k]} \leftarrow \mathbf{0} \in \mathbb{R}^{n_k}$
for $h = 1, 2, \ldots, H$
    choose $i \in \{1, 2, \ldots, n_k\}$ uniformly at random
    find $\Delta\alpha$ maximizing $-\lambda\nabla g^\star(\boldsymbol{v}^{(h-1)} + \frac{1}{\lambda n}\Delta\alpha\, \boldsymbol{x}_i) - \frac{1}{n}\ell_i^*\big( -(\alpha_i^{(h-1)} + \Delta\alpha)\big)$
    $\alpha_i^{(h)} \leftarrow \alpha_i^{(h-1)} + \Delta\alpha$
    $(\Delta\alpha_{[k]})_i \leftarrow (\Delta\alpha_{[k]})_i + \Delta\alpha$
    $\boldsymbol{v}^{(h)} \leftarrow \boldsymbol{v}^{(h-1)} + \frac{1}{\lambda n}\Delta\alpha\, \boldsymbol{x}_i$
end
Output: $\Delta\boldsymbol{\alpha}_{[k]}$ and $\Delta\boldsymbol{v} := A_{[k]}\Delta\boldsymbol{\alpha}_{[k]}$
```

Unfortunately solving the problem:

$$-\lambda\nabla g^\star(\boldsymbol{v}^{(h-1)} + \frac{1}{\lambda n}\Delta\alpha\, \boldsymbol{x}_i) - \frac{1}{n}\ell_i^*\big(-(\alpha_i^{(h-1)} + \Delta\alpha)\big)$$

can be very hard for complex regularizers and it's replaced in [2] by the following relaxed lowerbound problem (which still allows for finding the exact solution):

$$-\lambda\nabla g^\star(\boldsymbol{v}^{(h-1)})^T\Delta\alpha_i\boldsymbol{x}_i - \frac{1}{2\lambda n}\|\Delta\alpha\boldsymbol{x}_i\|^2 - \frac{1}{n}\ell_i^*\big(-(\alpha_i^{(h-1)} + \Delta\alpha)\big)$$

that is much simpler to solve since it only requires us to be able to compute the gradient $\nabla g^\star$ and after that it can easily be solved by line search. It is also possible to notice that this lowerbound problem is actually strict in the case of L2 regularization.

1.3.1 Single coordinate optimizer

For the sake of modularity, the second line in the SDCA loop has been abstracted in the code with the concept of single coordinate optimizer and a couple of them have been implemented; the 3 main ones are:

- BrentMethodOptimizer: this optimizer uses a derivative free line search method called Brent's method to solve the single coordinate problem. This allows from great generality since it easily allows to construct a solver for a new loss function by simply pluggin in the conjugate of the loss function. More common and maybe fast method will possibly be explored later on, but it's unlikely they will give any benefit since this optimization is not the bottle neck of the algorithm.
- Ad-hoc solver for SVM: this is basically the same as in the standard CoCoA paper. It is worth noticing that replacing this solver with the brent's method solver does not bring any noticeable loss of precision.
- Ad-hoc solver for Ridge regression: similarly as for svm, the ridge regression single coordinate problem is solvable in closed form.

1.3.2 Supported regularizers

In addition to L2 regularization elastic net has also been implemented and it works really well, provided that the L1 term is not more than three orders of magnitude smaller than the L1 term. An approximation scheme for the L1, taken from [2] has also been implemented (it consists in setting the L2 term to a very small value); unfortunately from some recent experiments it seems that it does not perform quite as well as liblinear (on logistic regression). In any case it still seems kind of practical.

Any combination of this losses and regularizers is possible.

1.3.3 Passcode

Another local solver provided as an option is passcode wild, taken from the Passcode paper (arxiv: 1504.01365v1). The algorithm consists of a parallel version of SDCA where t threads work in parallel on the same w vector (kept in shared memory) with no synchronization whatsoever (in the sense that they read and write from it with no locking or synchronization). This can result in losing the relationship between the α and the w (but I'm reconstructing the w from scratch from the alpha once in a while) but it gives a basically linear speedup. The correctness of the method in the paper has only been proven for L2 regularized losses but from my experiments it works just as fine for elastic net (even with a very small L2 term).

2 Appendix

3 Proof of Theorem 1 – Main Convergence Theorem

Proof. (The first part of the proof is identical (and thus copypasted) from [1])
From the definition of the update performed by Algorithm 1, we have $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \frac{1}{K}\sum_{k=1}^{K}\Delta\boldsymbol{\alpha}_{\langle[k]\rangle}$. Let us estimate the change of objective function after one outer iteration. Then using concavity of D we have

$$\begin{aligned} D(\boldsymbol{\alpha}^{(t+1)}) &= D\left(\boldsymbol{\alpha}^{(t)} + \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\Delta\boldsymbol{\alpha}_{\langle[k]\rangle}\right) = D\left(\tfrac{1}{K}\textstyle\sum_{k=1}^{K}(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}_{\langle[k]\rangle})\right) \\ &\geq \tfrac{1}{K}\textstyle\sum_{k=1}^{K} D(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}_{\langle[k]\rangle}). \end{aligned}$$

Subtracting $D(\boldsymbol{\alpha}^{(t)})$ from both sides and denoting by $\hat{\boldsymbol{\alpha}}^{*}_{[k]}$ the local maximizer as in (7) we obtain

$$\begin{aligned} D(\boldsymbol{\alpha}^{(t+1)}) - D(\boldsymbol{\alpha}^{(t)}) &\geq \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\left[D(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}_{\langle[k]\rangle}) - D(\boldsymbol{\alpha}^{(t)})\right] \\ &= \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\Big[D(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}_{\langle[k]\rangle}) - D((\boldsymbol{\alpha}^{(1)}_{[t]},\ldots,\hat{\boldsymbol{\alpha}}^{*}_{[k]},\ldots,\boldsymbol{\alpha}^{(K)}_{[t]})) \\ &\quad + D((\boldsymbol{\alpha}^{(1)}_{[t]},\ldots,\hat{\boldsymbol{\alpha}}^{*}_{[k]},\ldots,\boldsymbol{\alpha}^{(K)}_{[t]})) - D(\boldsymbol{\alpha}^{(t)})\Big] \\ &\overset{(7)}{=} \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\left[\varepsilon_{D,k}(\boldsymbol{\alpha}^{(t)}) - \varepsilon_{D,k}(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}_{\langle[k]\rangle})\right]. \end{aligned}$$

Considering the expectation of this quantity, we are now ready to use Assumption 1 on the *local geometric improvement* of the inner procedure. We have

$$\begin{aligned} \mathbf{E}[D(\boldsymbol{\alpha}^{(t+1)}) - D(\boldsymbol{\alpha}^{(t)}) \,|\, \boldsymbol{\alpha}^{(t)}] &\geq \tfrac{1}{K}\textstyle\sum_{k=1}^{K}\mathbf{E}[\varepsilon_{D,k}(\boldsymbol{\alpha}^{(t)}) - \varepsilon_{D,k}(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}_{\langle[k]\rangle}) \,|\, \boldsymbol{\alpha}^{(t)}] \\ &\overset{(6)}{\geq} \tfrac{1}{K}(1-\Theta)\textstyle\sum_{k=1}^{K}\varepsilon_{D,k}(\boldsymbol{\alpha}^{(t)}). \end{aligned}$$

(Copypasted part ends here, the proof now slightly diverges)
It remains to bound $\sum_{k=1}^{K}\varepsilon_{D,k}(\boldsymbol{\alpha}^{(t)})$.

$$
\begin{aligned}
\textstyle\sum_{k=1}^{K}\varepsilon_{D,k}(\boldsymbol{\alpha}^{(t)}) &\overset{(7)}{=} \max_{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^n}\Big\{\textstyle\sum_{k=1}^{K}\big[D((\boldsymbol{\alpha}_{[1]},\dots,\hat{\boldsymbol{\alpha}}_{[k]},\dots,\boldsymbol{\alpha}_{[K]})) - D((\boldsymbol{\alpha}_{[1]},\dots,\boldsymbol{\alpha}_{[k]},\dots,\boldsymbol{\alpha}_{[K]}))\big]\Big\}\\
&\overset{(3)}{=} \max_{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^n}\Big\{\tfrac{1}{n}\textstyle\sum_{i=1}^{n}(-\ell_i^*(-\hat{\alpha}_i)+\ell_i^*(-\alpha_i^{(t)}))\\
&\qquad +\lambda\textstyle\sum_{k=1}^{K}\Big[-g^\star\big(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})+\boldsymbol{v}(\hat{\boldsymbol{\alpha}}_{[k]}-\boldsymbol{\alpha}_{[k]}^{(t)})\big)+g^\star\big(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})\big)\Big]\Big\}\\
&\overset{smoothness}{>=} \max_{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^n}\Big\{\tfrac{1}{n}\textstyle\sum_{i=1}^{n}(-\ell_i^*(-\hat{\alpha}_i)+\ell_i^*(-\alpha_i^{(t)}))\\
&\qquad +\lambda\textstyle\sum_{k=1}^{K}\Big[-g^\star\big(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})\big)-\nabla g^\star\big(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})\big)^T\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}_{[k]}-\boldsymbol{\alpha}_{[k]}^{(t)}\big)-\tfrac{1}{2\mu}\|\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}_{[k]}-\boldsymbol{\alpha}_{[k]}^{(t)}\big)\|^2+g^\star\big(\boldsymbol{v}(\boldsymbol{\alpha}\\
&= \max_{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^n}\Big\{D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t)})+\lambda(g^\star(\boldsymbol{v}(\hat{\boldsymbol{\alpha}}))-g^\star(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})))\\
&\qquad +\lambda\textstyle\sum_{k=1}^{K}\Big[-\nabla g^\star\big(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})\big)^T\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}_{[k]}-\boldsymbol{\alpha}_{[k]}^{(t)}\big)-\tfrac{1}{2\mu}\|\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}_{[k]}-\boldsymbol{\alpha}_{[k]}^{(t)}\big)\|^2\Big]\Big\}\\
&= \max_{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^n}\Big\{D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t)})+\lambda(g^\star(\boldsymbol{v}(\hat{\boldsymbol{\alpha}}))-g^\star(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})))\\
&\qquad -\lambda\nabla g^\star\big(\boldsymbol{v}(\boldsymbol{\alpha}^{(t)})\big)^T\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}-\boldsymbol{\alpha}^{(t)}\big)-\frac{\lambda}{2\mu}\textstyle\sum_{k=1}^{K}\Big[\|\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}_{[k]}-\boldsymbol{\alpha}_{[k]}^{(t)}\big)\|^2\Big]\Big\}\\
&\overset{delta-stronglyconvexassumption}{>=} \max_{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^n}\Big\{D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t)})+\frac{\lambda\delta}{2}\|\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}-\boldsymbol{\alpha}^{(t)}\big)\|^2\\
&\qquad -\frac{\lambda}{2\mu}\textstyle\sum_{k=1}^{K}\Big[\|\boldsymbol{v}\big(\hat{\boldsymbol{\alpha}}_{[k]}-\boldsymbol{\alpha}_{[k]}^{(t)}\big)\|^2\Big]\Big\}\\
&= \max_{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^n}\Big\{D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t)})-\frac{\sigma}{2\lambda n^2}\|\hat{\boldsymbol{\alpha}}-\boldsymbol{\alpha}^{(t)}\|^2\ \Big\}
\end{aligned}
$$

And then the proof continues just the same. □

References

[1] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I. Jordan. Communication-efficient distributed dual coordinate ascent. *CoRR*, abs/1409.1458, 2014.

[2] Shai Shalev-Shwartz and Tong Zhang. Proximal Stochastic Dual Coordinate Ascent. *arXiv.org*, November 2012.