Outline

- The convex-concave min-max paradigm
- Instances / special cases
- Algorithms
  - basics
  - enhancements
  - nature of convergence / complexity analysis
- Algorithms with state-of-art complexity
  - PURE-CD: Dense variant
  - PURE-CD: Sparse variant
  - VRPDA^2
  - CLVR
- Computations

Collaborators: Ahmet Alacaoglu, Jelena Diakonikolas, Chaobing Song, Eric Lin.

Sources: Recent Conference Papers / Submissions

1. Alacaoglu, A., Cevher, V., and Wright, S. J., *On the Complexity of a Practical Primal-Dual Coordinate Method*, arXiv preprint arXiv:2201.07684 (2022).
2. Song, C., Lin, C. Y., Wright, S. J., and Diakonikolas, J., *Coordinate Linear Variance Reduction for Generalized Linear Programming*, arXiv preprint arXiv:2111.01842 (2021).
3. Song, C., Wright, S. J. and Diakonikolas, J., *Variance reduction via primal-dual accelerated dual averaging for nonsmooth convex finite-sums*, In International Conference on Machine Learning, pp. 9824-9834. PMLR (2021).

Convex-Concave Min-Max

$$\min_{x \in \mathbb{R}^d} \max_{y \in \mathbb{R}^n} L(x, y) \qquad \text{(Min-Max)}$$

where

$$\begin{aligned} L(x, y) &= \sum_{i=1}^{n} \left[\langle A_i x, y^{(i)} \rangle - h_i^*(y^{(i)}) \right] + g(x) \\ &= \langle Ax, y \rangle - h^*(y) + g(x), \end{aligned}$$

- $h_i^* : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is convex conjugate of h_i defined by $h_i^*(t) := \sup_s(st - h_i(s))$ (convex and extended-valued);
- $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ (convex and extended-valued);
- $h^*(y) = \sum_{i=1}^n h_i^*(y^{(i)})$ (separable);
- $A_i \in \mathbb{R}^d$ is a row vector;
- A is the $n \times d$ matrix with rows A_i.

More Specs

We consider cases in which A is dense and A is sparse.

In the case of sparse A, we assume for analysis that g is separable, that is,

$$g(x) = \sum_{j=1}^{d} g_j(x^{(j)}).$$

All algorithms make use of the prox-operator denoted for diagonal weighting matrix $\mathrm{T} \succ 0$ and function g by $\mathrm{prox}_{\mathrm{T},g}$ and defined

$$\begin{aligned}\mathrm{prox}_{\mathrm{T},g}(x) &:= \arg\min_u \ \tfrac{1}{2}\|u - x\|^2_{\mathrm{T}^{-1}} + g(u) \\ &= \arg\min_u \ \frac{1}{2}\sum_{i=1}^{d} \frac{(x^{(i)} - u^{(i)})^2}{\mathrm{T}_{ii}} + g(u).\end{aligned}$$

Assume that we can compute prox-operators for g and h_i^* "easily."

Case I: Empirical Risk Minimization (ERM)

Ubiquitous in statistics and machine learning.

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^{n} h_i(A_i x) + g(x). \tag{ERM}$$

- Least squares: $h_i(t) = \frac{1}{2}t^2$;
- ℓ_1 regression: $h_i(t) = |t|$;
- Hinge loss: $h_i(t) = \max(t, 0)$ (used in SVM, neural nets);
- Regularization: Tikhonov $g(x) = \lambda \|x\|_2^2$, $\quad \ell_1$: $g(x) = \|x\|_1$;
- TV regularization: $h_i(t) = \|t\|_2$, $A_i \in \mathbb{R}^{2 \times d}$;
- logistic regression, least absolute deviation, ...

Case II: Linearly Constrained Optimization

$$\min g(x) \text{ s.t. } A_i x \in C_i, \ i = 1, 2, \ldots, n, \qquad \text{(LinOpt)}$$

where $C_i = \{b_i\}$ (equality constraints) or $C_i = \{t : t \geq b_i\}$ (inequality).

Recall that we need to compute prox-operators involving convex g.

- Trivial if g is linear.
- Can be arranged (possibly via reformulation) if g is convex quadratic.

For algorithm output x_{out}, we could bound the number of iterates to attain

- $\mathbb{E}|g(x_{\text{out}}) - g(x_\star)| \leq \varepsilon$, and
- $\mathbb{E}\,\text{dist}(Ax_{\text{out}}, C) \leq \varepsilon$.

Case III: Generalized LP

$$\min\ c^T x + r(x)\ \text{ s.t. } Ax = b,\ x \in \mathcal{X}, \tag{GLP}$$

which can be written in min-max form as

$$\min_{x\in\mathcal{X}}\ \max_{y\in\mathbb{R}^n}\ L(x,y) = \langle Ax, y\rangle + c^T x + r(x) - b^T y.$$

$\mathcal{X} \subset \mathbb{R}^d$ is closed and convex, r is convex. We assume that the following modified prox-operator is easy to compute:

$$\text{prox}_{\mathcal{X},r}(\hat{x}) := \arg\min_{z\in\mathcal{X}}\ \tfrac{1}{2}\|z-\hat{x}\|_2^2 + r(z).$$

- Ordinary LP: $\mathcal{X} = \mathbb{R}^d_{\geq 0}$ and regularized LP: $\mathcal{X} = \mathbb{R}^d_{\geq 0}$, $r(x) = \lambda\|x\|_2^2$;
- Reinforcement Learning [De Farias and Van Roy, 2003]
- Optimal Transport [Villani, 2009]
- DRO (f-divergence, Wasserstein) (see below)
- relaxed Neural Net verification [Liu et al., 2020].

Case IIIa: Distributionally Robust Optimization (DRO)

Setup: sample vectors $\{a_1, a_2, \ldots, a_n\}$ in $\mathbb{R}^d$ with labels $\{b_1, b_2, \ldots, b_n\}$, where $b_i \in \{1, -1\}$. Usual ERM problem is

$$\min_w \frac{1}{n} \sum_{i=1}^{n} h(b_i a_i^T w)$$

where $h : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is convex (e.g. hinge loss).

- Wasserstein metric defines a distance between distributions $\mathbb{P}$ and $\mathbb{Q}$ over $\mathbb{R}^d \times \{-1, 1\}$, based on cost

$$\zeta((a, b), (a', b')) = \|a - a'\|_1 + \kappa |b - b'|$$

  for some $\kappa > 0$;
- $\mathbb{P}_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{(a_i, b_i)}$ is the empirical distribution defined by the data;
- Seek sup of the objective over the ball of radius ρ around $\mathbb{P}_n$ (in space of distributions over (a, b)) defined by the Wasserstein metric:

$$\min_{w \in \mathbb{R}^d} \sup_{\text{dist}(\mathbb{P}, \mathbb{P}_n) \le \rho} \mathbb{E}^{\mathbb{P}}[h(ba^T w)].$$

GLP formulation of the DRO problem above is

$$\begin{aligned}
\min_{w,\lambda,u,v,s,t} \quad & \rho\lambda + \frac{1}{n}\sum_{i=1}^{n} s_i \\
\text{s.t.} \quad & u_i = b_i a_i^T w, && i = 1,2,\ldots,n, \\
& v_i = -u_i, && i = 1,2,\ldots,n, \\
& t_i = 2\kappa\lambda + s_i, && i = 1,2,\ldots,n, \\
& h(u_i) \le s_i, && i = 1,2,\ldots,n, \\
& h(v_i) \le t_i, && i = 1,2,\ldots,n, \\
& \|w\|_\infty \le \lambda/M.
\end{aligned}$$

$\mathcal{X}$ is defined by the last 3 constraints. The corresponding prox operation is separable so can be implemented easily.

Algorithms: The Basics

Formulation (reminder):

$$\min_{x\in\mathbb{R}^d}\max_{y\in\mathbb{R}^n} L(x,y) \qquad \text{(Min-Max)}$$

$$\begin{aligned} L(x,y) &= \sum_{i=1}^{n}\left[\langle A_i x, y^{(i)}\rangle - h_i^*(y^{(i)})\right] + g(x) \\ &= \langle Ax, y\rangle - h^*(y) + g(x), \end{aligned}$$

Gradient Ascent-Descent (GDA):

$$\begin{aligned} \bar{x}_{k+1} &= \text{prox}_{\tau,g}(\bar{x}_k - \tau A^\top \bar{y}_k) \\ \bar{y}_{k+1} &= \text{prox}_{\sigma,h^*}(\bar{y}_k + \sigma A \bar{x}_{k+1}), \end{aligned} \qquad \text{(GDA)}$$

for positive step sizes τ and σ. [1]

[1] $\mathrm{T} = \tau I$ in our earlier definition of prox.

Algorithms: The Basics

Primal-Dual Hybrid Gradient (PDHG) [Chambolle and Pock, 2011] uses extrapolation in the x step:

$$\begin{aligned} \bar{x}_{k+1} &= \text{prox}_{\tau,g}(\bar{x}_k - \tau A^\top (2\bar{y}_k - \bar{y}_{k-1})) \\ \bar{y}_{k+1} &= \text{prox}_{\sigma,h^*}(\bar{y}_k + \sigma A \bar{x}_{k+1}), \end{aligned} \tag{PDHG}$$

Equivalent form of PDHG:

$$\bar{x}_{k+1} = \text{prox}_{\tau,g}(\hat{x}_k - \tau A^\top \bar{y}_k) \tag{1a}$$

$$\bar{y}_{k+1} = \text{prox}_{\sigma,h^*}(\bar{y}_k + \sigma A \bar{x}_{k+1}) \tag{1b}$$

$$\hat{x}_{k+1} = \bar{x}_{k+1} - \tau A^\top (\bar{y}_{k+1} - \bar{y}_k). \tag{1c}$$

[Chambolle and Pock, 2011] discusses connections to Douglas-Rachford, Extrapolated gradient, ADMM.

[Aragón-Artacho et al., 2020] show application of DR to finding intersection of sets, pictures show the benefits of extrapolation.

Algorithms: Additional Features

Theoretical convergence / complexity properties of these algorithms can be improved (in some cases, including strong convexity / concavity and sparsity) by adding extra features.

- Coordinate descent: e.g. update random element(s) of y in (1b) instead of the whole vector.
- Variance Reduction: Adjust the update formula for x to account for noise arising from coordinate update of y.
- Dual Averaging: At step k, use a gradient term that is a weighted average over all previous iterations.
- Importance sampling: Apply different weights to different components of each update (e.g. weight matrix T in definition of prox).
- Iterate averaging: Output a weighted average of iterates, rather than the final iterate for x.

Some are used by PURE-CD and VRPDA[2].

Measuring Approximate Optimality

Algorithms can be compared using rate of reduction of a gap function that measures near-optimality. Define G as follows:

$$\begin{aligned} G(x', y', x, y) &:= L(x', y) - L(x, y') \\ &= [g(x') + \langle Ax', y\rangle - h^*(y)] - [g(x) + \langle Ax, y'\rangle - h^*(y')], \end{aligned}$$

and for compact set $\mathcal{Z} \subset \mathbb{R}^d \times \mathbb{R}^n$ define the gap function

$$\text{Gap}(x', y') := \max_{(x,y)\in\mathcal{Z}} G(x', y', x, y).$$

Suppose output (x^K, y^K) is generated by an algorithm after K iterations.

If the algorithm is *random*, can measure:

- "max of expectation" $\max_{(x,y)\in\mathcal{Z}} \mathbb{E} G(x^K, y^K, x, y)$;
- "expected gap" (stronger)

$$\mathbb{E} \max_{(x,y)\in\mathcal{Z}} G(x^K, y^K, x, y) = \mathbb{E}\, \text{Gap}(x^K, y^K).$$

Complexity Analysis

Find upper bounds on the number of flops needed to reduce (expected) gap measures below a given threshold $\varepsilon > 0$. Particularly interested in dependence on ε as well as

- Dimensions d (for primal x) and n (for dual y);
- size of A: e.g. $\|A\|$, $\max_{i=1,2,\ldots,n} \|A_i\|$, or $\sum_{i=1}^{n} \|A_i\|$;
- $\mathrm{nnz}(A)$ (for sparse A);
- Distance between (x_0, y_0) and the optimum $(x^\star, y^\star)$.

Some algorithms (e.g. stochastic PDHG [Chambolle et al., 2018]) have less impressive bounds yet perform well for some types of problems.

PURE-CD: Dense A [Alacaoglu et al., 2020]

Define coordinate selection probabilities $p(i)$, $i = 1, 2, \ldots, n$, with $P = \text{diag}(p^{(1)}, \ldots, p^{(n)})$

1: Initialize $x_0 \in \text{dom}\, g$, $y_0 \in \text{dom}\, h^*$
2: **for** $k \geq 0$ **do**
3: $\quad \bar{x}_{k+1} = \text{prox}_{\mathrm{T}_k, g}(x_k - \mathrm{T}_k A^\top y_k)$
4: $\quad$ Pick $i_k \in [n]$ with $\Pr(i_k = i) = p^{(i)}$
5: $\quad \left[y_{k+1} = \text{prox}_{\sigma_k, h^*}(y_k + \sigma_k A \bar{x}_{k+1})\right]_{i_k}, \quad [y_{k+1} = y_k]_{\setminus i_k}$
6: $\quad x_{k+1} = \bar{x}_{k+1} - \mathrm{T}_k \Theta_k A^\top P^{-1}(y_{k+1} - y_k)$
7: **end for**

Notation:

- $[\cdot]_J$ means that the formula is executed on only the components indexed by the set J.
- $[\cdot]_{\setminus J}$ means that the formula is executed on all components *except* those indexed by the set J.

PURE-CD: Dense A: Notes

Cost per iteration: $O(d)$

- $\text{prox}_{\mathrm{T}_k, g}$,
- calculate $A_{i_k} \bar{x}_{k+1}$,
- update $A^\top y_{k+1}$.

After K iterations, output $x^K =$ average of $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_K$.
(If needed, also output a weighted average of the y_k.)

Tracks PDHG with key differences:

- diagonal scalings: T_k, P, Θ_k;
- Update just one (random) component of y in lines 4 and 5.

PURE-CD Dense: Complexity Result for LinOpt

$$\min g(x) \ \text{ s.t. } A_i x \in C_i, \ i = 1, 2, \ldots, n.$$

Assume that primal-dual solution $(x_\star, y_\star)$ exists. For scaling matrices:

$$\mathrm{T}_k = \mathrm{T} = \tau I, \quad \tau = \frac{1}{\sum_{i=1}^n \|A_i\|},$$

$$\Sigma_k = \Sigma = \mathrm{diag}(\sigma^{(1)}, \ldots, \sigma^{(n)}), \quad \sigma^{(i)} = \frac{\gamma}{\|A_i\|}, \ \gamma \in (0,1),$$

$$p^{(i)} = \frac{\|A_i\|}{\sum_{i=1}^n \|A_i\|}, \quad \Theta_k = \theta I = I.$$

Define $x^K = \frac{1}{K}\sum_{k=1}^K \bar{x}_k$. Then for D_1, D_2 dep. on (x_0, y_0) and $(x_\star, y_\star)$:

$$\mathbb{E}|g(x^K) - g(x_\star)| \leq \frac{8D_2}{\gamma(1-\gamma)} \frac{\sum_{i=1}^n \|A_i\|}{K},$$

$$\mathbb{E}\left[\mathrm{dist}(Ax^K, C)\right] \leq \frac{8D_1}{\gamma(1-\gamma)} \frac{\sum_{i=1}^n \|A_i\|}{K},$$

with cost $O(d)$ per iteration.

PURE-CD Dense: Complexity Result for Min-Max

Here need more complicated settings of T_k (used in the algorithm) and Σ_k (used only in analysis). Set $\Theta_k \equiv I$ and $\sigma^{(i)} = \gamma / \|A_i\|$, some $\gamma \in (0, 1)$.

Weighted averages of $\bar{x}_k$ and y_k for $k = 1, 2, \ldots, K$ yield outputs (x^K, y^K).

Use a special choice of starting points $\bar{x}_1$, y_1 due to [Song et al., 2021b]

Obtain a complicated bound for $\mathbb{E}\,\mathrm{Gap}(x^K, y^K)$ as a function of K – ultimately, arithmetic convergence at $O(1/K)$ rate, after a burn-in period of slow linear convergence.

Can guarantee $\mathbb{E}\,\mathrm{Gap}(x^K, y^K) \leq \varepsilon$ with expected complexity

$$K = \tilde{O}\left(nd + D_{\mathcal{Z}} \frac{nd \max_i \|A_i\|}{\varepsilon} \right) \quad \text{iterations,}$$

where $D_{\mathcal{Z}}$ depends on set $\mathcal{Z}$ and (x_0, y_0) and $\tilde{O}$ indicates "ignoring log terms."

PURE-CD: Sparse A [Alacaoglu et al., 2020]

Recall notation $J(i) := \{j \in [d] : A_{i,j} \neq 0\}$

1: Initialize $x_0 \in \operatorname{dom} g$, $y_0 \in \operatorname{dom} h^*$;
2: **for** $k \geq 0$ **do**
3: Pick $i_k \in [n]$ with $\Pr(i_k = i) = \frac{1}{n}$
4: $\left[\bar{x}_{k+1} = \operatorname{prox}_{\tau_k, g_j}\left(x_k - \tau_k (A^\top y_k)\right)\right]_{J(i_k)}$
5: $\left[y_{k+1} = \operatorname{prox}_{\sigma_k, h^*}(y_k + \sigma_k A \bar{x}_{k+1})\right]_{i_k}$
6: $[y_{k+1} = y_k]_{\setminus i_k}$
7: $\left[x_{k+1} = \bar{x}_{k+1} - \tau_k \theta_k A_{i_k}^T (y_{k+1}^{(i_k)} - y_k^{(i_k)})\right]_{J(i_k)}$
8: $[x_{k+1} = x_k]_{\setminus J(i_k)}$
9: **end for**

PURE-CD: Sparse A: Comments

- Assumes that g is separable: $g(x) = \sum_{j=1}^{d} g_j(x^{(j)})$.
- Similar but not identical to the dense version of PURE-CD.
  - Calculate only the $J(i_k)$ components of the intermediate vector $\bar{x}_{k+1}$, which are needed to update the i_k component of y.
  - Since x_{k+1} depends componentwise on $\bar{x}_{k+1}$, we calculate only the $J(i_k)$ components of $\bar{x}_{k+1}$.
  - But in the dense version, *all* components of $\bar{x}_{k+1}$ are required to obtain x_{k+1}.
  - Thus PURE-CD: Sparse needs its own convergence theory.
- Cost for iteration k is $O(|J(i_k)|)$ not $O(d)$.
- (Expected cost is $O(\mathrm{nnz}(A)/n)$.)

PURE-CD Sparse: Complexity Results for Min-Max

Focus on results where strong convexity is present in g and/or h^* (both separable functions).

- Each g_j has modulus of convexity $\mu_g \geq 0$;
- Each h_i^* has modulus of convexity $\mu_h \geq 0$,

Results are for last iterates x_K and/or y_K, not averaged iterates.

When $\mu_g > 0$ and $\mu_h > 0$, we have $\mathbb{E}\left[\|x_K - x_\star\|^2 + \|y_K - y_\star\|^2\right] \leq \varepsilon$ with expected complexity

$$\tilde{O}\left(\left(\mathrm{nnz}(A)\left(1 + \frac{\max_i \|A_i\|}{\sqrt{\mu_h \mu_g}}\right)\right)\log \varepsilon^{-1}\right).$$

Choices of Θ_k, $\sigma_k^{(i)}$, T_k do not depend on k, but require knowledge of μ_g and μ_h

PURE-CD Sparse: Complexity Results for Min-Max

When $\mu_g > 0$ but possibly $\mu_h = 0$ (strong convexity in g only) can make a (complicated) choice of parameters to ensure that $\mathbb{E}\left[\|x_K - x_\star\|^2\right] \leq \varepsilon$ with expected complexity

$$O\left(\text{nnz}(A)\left(1 + \sqrt{\frac{D_\star}{\varepsilon}}\max\left(1, \frac{\max_i \|A_i\|}{\mu_g}\right)\right)\right),$$

When $\mu_h > 0$ but possibly $\mu_g = 0$ (strong convexity in h only) a different (still complicated) choice of parameters $\sigma_k^{(j)}$, $\tau_k^{(j)}$, Θ_k ensures that $\mathbb{E}\left[\|y_K - y_\star\|^2\right] \leq \varepsilon$ with expected complexity

$$O\left(\text{nnz}(A)\left(1 + \sqrt{\frac{D_\star}{\varepsilon}}\max\left(1, \frac{\max_i \|A_i\|}{\mu_h}\right)\right)\right),$$

Complexity Comparisons

The PURE-CD complexity bounds are compared with various other algorithms for Min-Max, or special cases of it:

- PDHG [Chambolle and Pock, 2011]
- SPDHG [Chambolle et al., 2018]
- VRPDA [Song et al., 2021b]
- CLVR [Song et al., 2021a]
- SPDAD [Tan et al., 2020]
- VRVI [Carmon et al., 2019, Alacaoglu and Malitsky, 2021]
- Katyusha [Allen-Zhu, 2017]
- SPDC [Zhang and Lin, 2015]

In each case, PURE-CD matches or improves the complexities of these alternatives, in terms of their dependence on n, d, measures of A, ε.

A typical improvement is $\|A\| \to \max_i \|A_i\|$ – a factor of up to $\sqrt{n}$.

Comments on Proofs

The proofs of these complexity results are extremely technical, involving mostly elementary manipulation of inequalities.

Telescoping sums over iterations $k = 1, 2, \ldots, K$ is used often, and convexity is essential.

But considerable expertise is needed to choose the algorithmic parameters T_k, $\sigma_k^{(i)}$, Θ_k to achieve the desired cancellations.

VRPDA^2: Another algorithm for Min-Max

[Song et al., 2021b]

Input: $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}, (u, v) \in \mathcal{X} \times \mathcal{Y}$.

$\phi_0(\cdot) = \frac{1}{2}\|\cdot - x_0\|^2, \psi_0(\cdot) = \frac{1}{2}\|\cdot - y_0\|^2$. Initialize y_1, x_1, $\tilde{\psi}_1$, $\tilde{\phi}_1$.

$a_0 = B_0 = 0, \tilde{a}_1 = [2n \max_i \|A_i\|]^{-1}$

$\psi_1 := n\tilde{\psi}_1, \phi_1 := n\tilde{\phi}_1, a_1 = B_1 = n\tilde{a}_1, z_1 = A^T y_1,$

for $k = 2, 3, \ldots, K$ **do**

$$a_k = \min\left\{\tilde{a}_1\left(1 + \frac{1}{n-1}\right)^{k-1}, \frac{\sqrt{n(n+\sigma B_{k-1})}}{2R'}\right\}, B_k = B_{k-1} + a_k.$$

Pick i_k uniformly at random in $[n]$.

$\bar{x}_{k-1} = x_{k-1} + \frac{a_{k-1}}{a_k}(x_{k-1} - x_{k-2})$.

$y_k = \arg\min_{y \in \mathbb{R}^n} \psi_k(y) := \psi_{k-1}(y) + na_k(\langle -A_{i_k}\bar{x}_{k-1}, y_{i_k} - v_{i_k}\rangle + h^*_{i_k}(y_{i_k}))$.

$x_k = \arg\min_{x \in \mathbb{R}^d} \phi_k(x) :=$
$\phi_{k-1}(x) + na_k\langle x - u, z_{k-1} + (y_{k,i_k} - y_{k-1,i_k})A^T_{i_k}\rangle + g(x))$.

$z_k = z_{k-1} + (y_{k,i_k} - y_{k-1,i_k})A^T_{i_k}$.

end for

return weighted averages y^K and x^K.

VRPDA^2: Notes

Also motivated by PDHG, similarities with PURE-CD.

- Special initialization for x_1 and y_1. Costs one mult. by A and A^T.
- Extrapolation step in x (Line 8) with variable coefficient a_{k-1}/a_k.
- Update a single coordinate i_k of y (Line 9).
- Based on averaged gradients rather than steepest descent (as in PURE-CD
- Can't adapt to sparsity in A.
- Update steps (Lines 9-10) are prox-operations on $h^*_{i_k}$ and g.
- Outputs weighted-average of x_k and y_k, $k = 1, 2, \ldots, K$ as the result.

The algorithm and its analysis make heavy use of estimate sequences, in the style of [Nesterov, 2004].

$$\phi_k(x) := \phi_{k-1}(x) + a_k(n\langle x - u, z_{k-1} + (y_{k,i_k} - y_{k-1,i_k})A^T_{i_k}\rangle + g(x))$$
$$\psi_k(y) := \psi_{k-1}(y) + na_k(\langle -A_{i_k}\bar{x}_{k-1}, y_{i_k} - v_{i_k}\rangle + h^*_{i_k}(y_{i_k}))$$

VRPDA[2]: Convergence

For general case, attain $\mathbb{E}G(x^K, y^K, x^\star, y^\star) \leq \varepsilon$ (for weighted average iterates) in complexity

$$\tilde{O}\left(\frac{nd \max_i \|A_i\|}{\varepsilon}\right).$$

When g is strongly convex with modulus μ_g, get $\mathbb{E}\|x^\star - x_K\|^2 \leq \varepsilon$ (for last iterate x_K) in complexity

$$\tilde{O}\left(\frac{nd \max_i \|A_i\|}{\mu_g \sqrt{\varepsilon}}\right).$$

In PURE-CD we have the same bound [2] but with *nd* replaced by $\mathrm{nnz}(A)$.

[2] when $\max_i \|A_i\|/\mu_g > 1$

CLVR: Specialized to GLP [Song et al., 2021a]

$$\min\ c^T x + r(x)\ \text{ s.t. } Ax = b,\ x \in \mathcal{X}. \qquad \text{(GLP)}$$

Partition A into m row blocks – index partition $\{S^1, S^2, \ldots, S^m\}$.

Input: $x_0 \in \mathcal{X}, y_0 \in \mathbb{R}^n, z_0 = A^T y_0, \gamma > 0, \hat{L} > 0, \sigma \geq 0, K$.
$a_1 = B_1 = \frac{1}{2\hat{L}m}, q_0 = a_1(z_0 + c)$.
for $k = 1, 2, \ldots, K$ **do**
  $x_k = \text{prox}_{\frac{1}{\gamma} B_k r}(x_0 - \frac{1}{\gamma} q_{k-1})$.
  Pick j_k uniformly at random in $\{1, 2, \ldots, m\}$.
  $[y_k = y_{k-1}]_{\setminus S^{j_k}}$; $[y_k = y_{k-1} + \gamma m a_k (Ax_k - b)]_{S^{j_k}}$;
  $a_{k+1} = \frac{\sqrt{1+\sigma B_k/\gamma}}{2\hat{L}m}$, $B_{k+1} = B_k + a_{k+1}$.
  $z_k = z_{k-1} + A_{S^{j_k}}^T (y_k^{S^{j_k}} - y_{k-1}^{S^{j_k}})$.
  $q_k = q_{k-1} + a_{k+1}(z_k + c) + m a_k (z_k - z_{k-1})$.
end for
return weighted averages x^K and y^K.

CLVR: Notes and Complexity

Again related to PDHD but with variations.

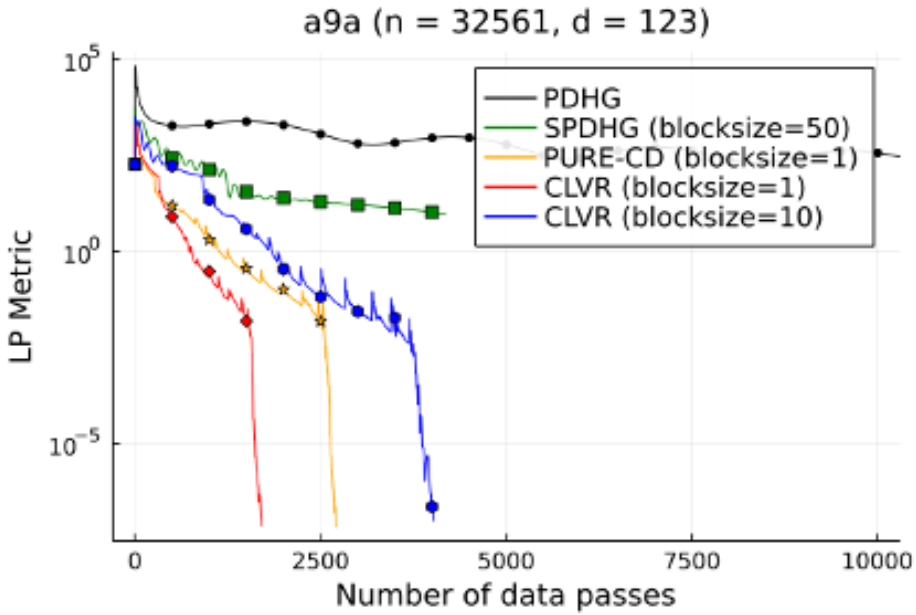
- Averaged gradients in x, block coordinate descent in y.
- Recall that specialized prox-operator involves constraint set $\mathcal{X}$.
- Can be implemented in a way that exploits sparsity in A
  - ....but this involves intermediate vectors and is more complicated than in Sparse PURE-CD.

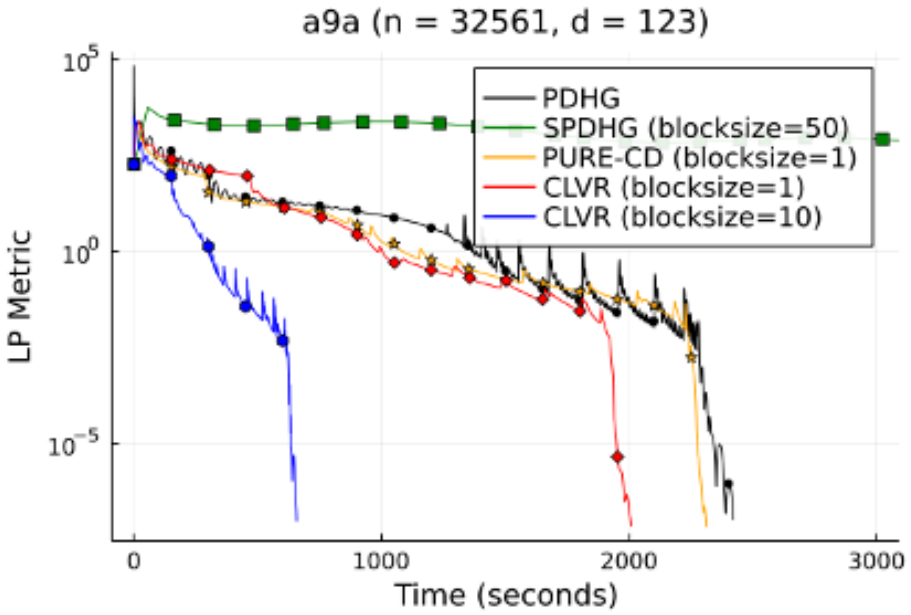
Expected complexity for $\mathbb{E}G(x^K, y^K, x^\star, y^\star) < \varepsilon$ in Sparse CLVR is

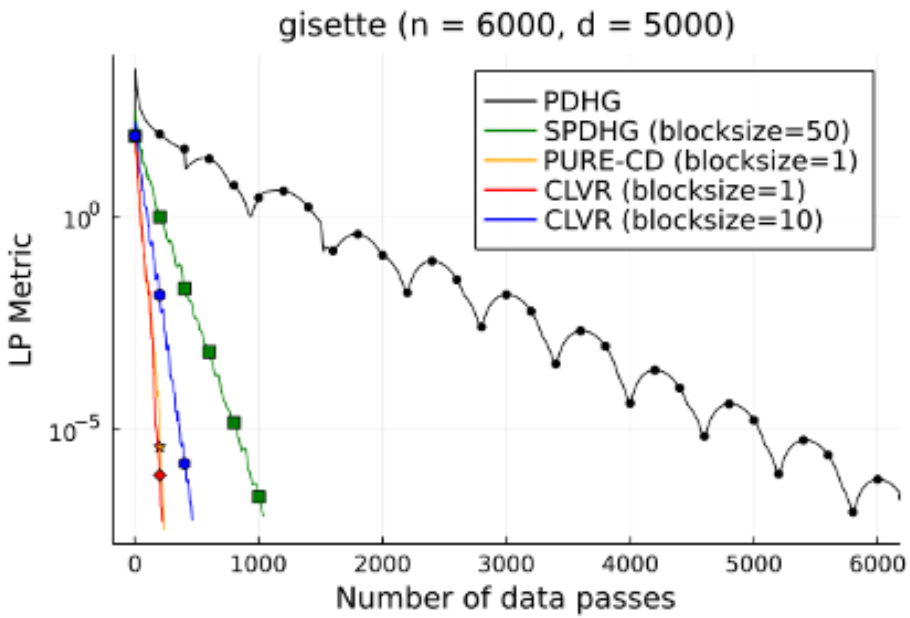
$$O\left(\frac{\mathrm{nnz}(A)\max_{i=1,2,\ldots,m}\|A_{S^i}\|}{\varepsilon}\right).$$

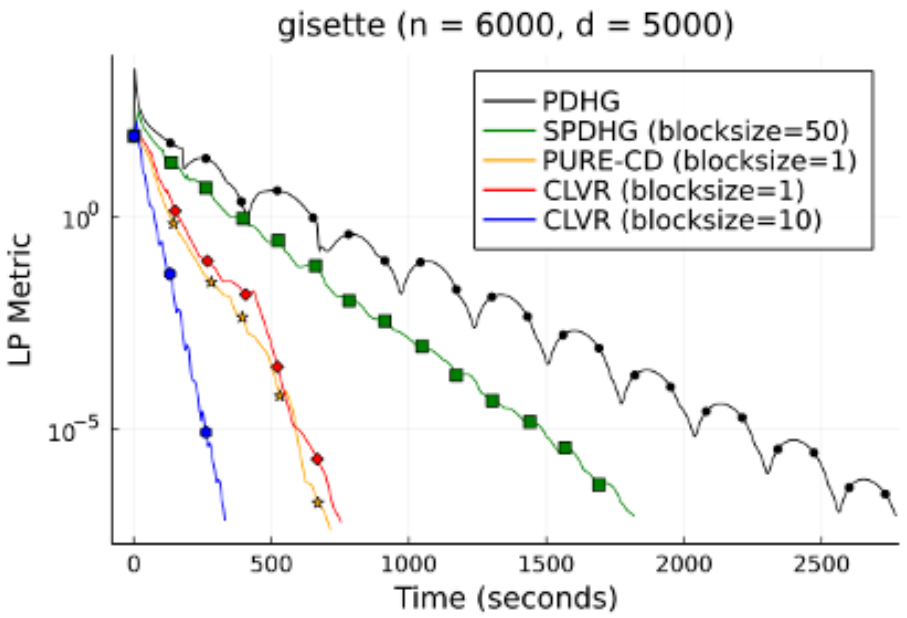
Computational Results: DRO

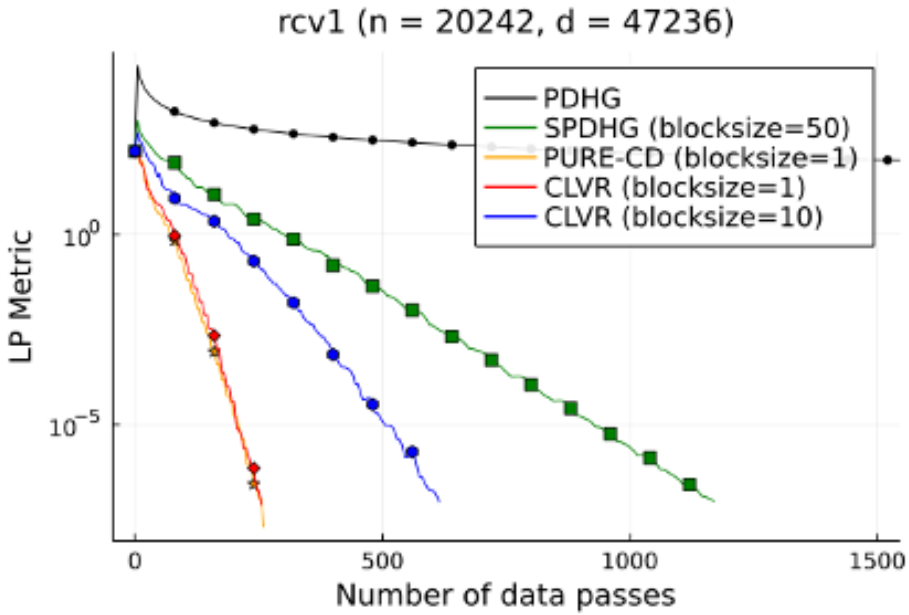
- Wasserstein DRO described above, with ℓ_1 norm and hinge loss.
- Several standard ML datasets (LIBSVM).
- Implemented in **Julia**. Use **SparseArrays** to support sparse vectors and matrices.
- CLVR uses blocks to improve utilization of multiple cores.

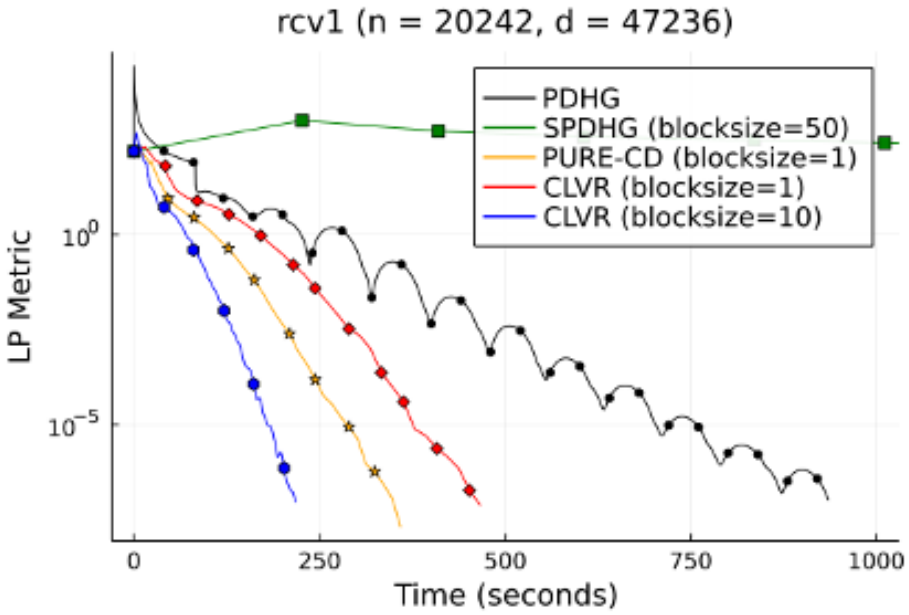

a9a (n = 32561, d = 123)
LP Metric
Number of data passes
PDHG
SPDHG (blocksize=50)
PURE-CD (blocksize=1)
CLVR (blocksize=1)
CLVR (blocksize=10)


a9a (n = 32561, d = 123)
LP Metric
Time (seconds)
PDHG
SPDHG (blocksize=50)
PURE-CD (blocksize=1)
CLVR (blocksize=1)
CLVR (blocksize=10)


gisette (n = 6000, d = 5000)
PDHG
SPDHG (blocksize=50)
PURE-CD (blocksize=1)
CLVR (blocksize=1)
CLVR (blocksize=10)
LP Metric
10^0
10^{-5}
0
1000
2000
3000
4000
5000
6000
Number of data passes


gisette (n = 6000, d = 5000)
PDHG
SPDHG (blocksize=50)
PURE-CD (blocksize=1)
CLVR (blocksize=1)
CLVR (blocksize=10)
LP Metric
10^0
10^{-5}
0
500
1000
1500
2000
2500
Time (seconds)


rcv1 (n = 20242, d = 47236)
PDHG
SPDHG (blocksize=50)
PURE-CD (blocksize=1)
CLVR (blocksize=1)
CLVR (blocksize=10)
LP Metric
10^0
10^{-5}
0
500
1000
1500
Number of data passes


rcv1 (n = 20242, d = 47236)
PDHG
SPDHG (blocksize=50)
PURE-CD (blocksize=1)
CLVR (blocksize=1)
CLVR (blocksize=10)
LP Metric
10^0
10^{-5}
0
250
500
750
1000
Time (seconds)

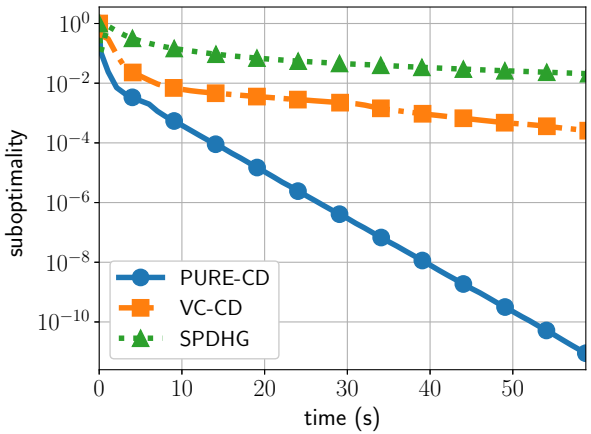
Comparing with General LP solvers (times)

Time (seconds)	Reformulated a9a $d = 130738, n = 97929$	Reformulated gisette $d = 44002, n = 28000$	Reformulated rcv1 $d = 269914, n = 155198$
PDHG	2422	2772	935
SPDHG	$> 4 \times 10^4$	1820	3.7×10^4
JuMP+GLPK	899	$> 4 \times 10^4$	$> 4 \times 10^4$
JuMP+Gurobi(simplex)	893	2482	7008
JuMP+Gurobi(barrier)	**26**	1039.7	1039.5
CLVR	962	**697**	**582**

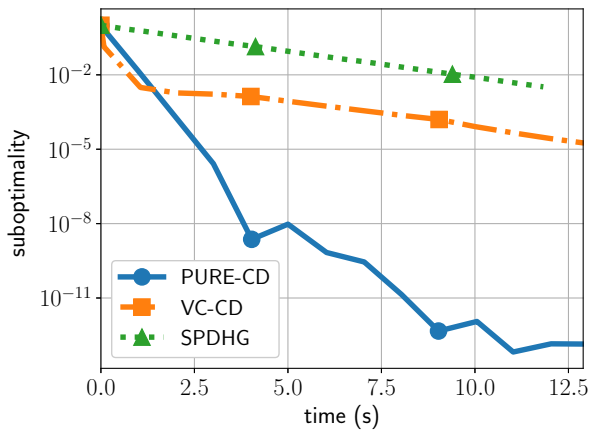
Adapting to sparsity with PURE-CD

Investigating adaptation to sparsity in Lasso problem, compared to other stochastic coordinate methods

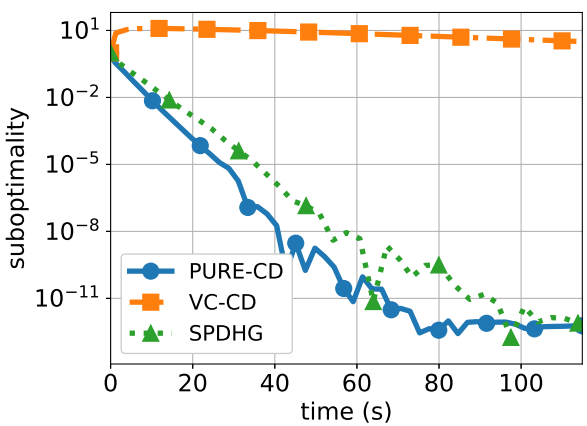
- SPDHG [Chambolle et al., 2018][3]: good for dense, but $O(d)$ per iteration cost
- VC-CD [Fercoq and Bianchi, 2019][4]: good for sparse, but n times worse step size for dense
- PURE-CD: good for dense & sparse



rcv1: 0.16% density



w8a: 3.9% density



covtype: 22.1% density

[3] Stochastic PDHG: Chambolle et al., SIOPT, 2018

[4] Vu-Condat with coordinate descent: Fercoq, Bianchi, SIOPT, 2019

Concluding Thoughts and Questions

> *Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.*

- Edsger Wybe Dijkstra

Algorithms that are simple yet with optimal complexity properties have taken some time to arrive. The analysis is still highly technical.

Can we simplify the analysis? Or define slightly different (but still useful) measures of algorithm performance that admit simpler analysis?

References I


Alacaoglu, A., Fercoq, O., and Cevher, V. (2020).
Random extrapolation for primal-dual coordinate descent.
In *International Conference on Machine Learning*, pages 191–201. PMLR.


Alacaoglu, A. and Malitsky, Y. (2021).
Stochastic variance reduction for variational inequality methods.
arXiv preprint arXiv:2102.08352.


Allen-Zhu, Z. (2017).
Katyusha: The first direct acceleration of stochastic gradient methods.
The Journal of Machine Learning Research, 18(1):8194–8244.


Aragón-Artacho, F. J., Campoy, R., and Tam, M. K. (2020).
The Douglas–Rachford algorithm for convex and nonconvex feasibility problems.
Mathematical Methods of Operations Research, 91:201–240.


Carmon, Y., Jin, Y., Sidford, A., and Tian, K. (2019).
Variance reduction for matrix games.
Advances in Neural Information Processing Systems.


Chambolle, A., Ehrhardt, M. J., Richtárik, P., and Schonlieb, C.-B. (2018).
Stochastic primal-dual hybrid gradient algorithm with arbitrary sampling and imaging applications.
SIAM Journal on Optimization, 28(4):2783–2808.

References II

Chambolle, A. and Pock, T. (2011).
A first-order primal-dual algorithm for convex problems with applications to imaging.
Journal of Mathematical Imaging and Vision, 40(1):120–145.

De Farias, D. P. and Van Roy, B. (2003).
The linear programming approach to approximate dynamic programming.
Operations research, 51(6):850–865.

Fercoq, O. and Bianchi, P. (2019).
A coordinate-descent primal-dual algorithm with large step size and possibly nonseparable functions.
SIAM Journal on Optimization, 29(1):100–134.

Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M. J., et al. (2020).
Algorithms for verifying deep neural networks.
Foundations and Trends® in Optimization, 4.

Nesterov, Y. (2004).
Introductory Lectures on Convex Optimization: A Basic Course.
Springer Science and Business Media, New York.

Song, C., Lin, C. Y., Wright, S. J., and Diakonikolas, J. (2021a).
Coordinate linear variance reduction for generalized linear programming.
arXiv preprint arXiv:2111.01842.

References III

Song, C., Wright, S. J., and Diakonikolas, J. (2021b).
Variance reduction via primal-dual accelerated dual averaging for nonsmooth convex finite-sums.
In *International Conference on Machine Learning*, pages 9824–9834. PMLR.

Tan, C., Qian, Y., Ma, S., and Zhang, T. (2020).
Accelerated dual-averaging primal–dual method for composite convex minimization.
Optimization Methods and Software, 35(4):741–766.

Villani, C. (2009).
Optimal transport: old and new, volume 338.
Springer.

Zhang, Y. and Lin, X. (2015).
Stochastic primal-dual coordinate method for regularized empirical risk minimization.
In *International Conference on Machine Learning*, pages 353–361. PMLR.