Qi Lei*, Jiacheng Zhuo*, Constantine Caramanis*, Inderjit S. Dhillon*,† and Alexandros G. Dimakis*.

* University of Texas at Austin
† Amazon
Problem Setup

Convex-concave saddle point Problem (with constraints):

$$\min_{x \in C \subset \mathbb{R}^d} \max_{y \in \mathbb{R}^n} \left\{ L(x, y) = f(x) + y^\top A x - g(y) \right\}$$

Why is this formulation important?

1. Many machine learning applications
Machine Learning Applications with Convex-Concave Formulations

Empirical Risk Minimization

Logistic Regression Model

\[
\begin{align*}
X_1 & \rightarrow \theta_1 \\
X_2 & \rightarrow \theta_2 \\
X_3 & \rightarrow \theta_3
\end{align*}
\]

Inputs: \(X_1, X_2, X_3\) || Weights: \(\theta_1, \theta_2, \theta_3\) || Outputs: Happy or Sad

@dataaspirant.com

Reinforcement Learning

Robust Optimization

Environment

Action

Reward

Interpreter

State

Agent

NeurIPS19

Primal-Dual Block Generalized Frank-Wolfe

3 / 11
Problem Setup

Convex-Concave Saddle Point Problem:

\[
\min_{x \in \mathbb{R}^d} \max_{y \in \mathbb{R}^n} \{ L(x, y) = f(x) + y^\top Ax - g(y) \}
\]

Why is this formulation important?

1. Many machine learning applications
2. To exploit special structure induced by the constraints
Lessons from simple constrained minimization problems:

Observations.

Frank-Wolfe conducts *partial updates*:
1. For $\ell_1$ ball constraint, FW conducts **1-sparse** update
2. For nuclear norm ball constraint, FW conducts **rank-1** update

Challenges to get full benefits from FW and the partial updates.

1. FW yield **sublinear convergence** even for strongly convex problems
2. Even with partial updates, FW requires to compute the full gradient. (For big data setting, *per iteration complexity is the same with projected gradient descent.* )
Tackle challenge 1: To achieve linear convergence

Continue to look at simple minimization problems:

\[
\min_{x \in \mathbb{R}^d, \|x\|_1 \leq \tau} \{ f(x) \}
\]

<table>
<thead>
<tr>
<th>Method</th>
<th>iteration complexity</th>
<th># update per iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projected GD</td>
<td>( \kappa \log \frac{1}{\epsilon} )</td>
<td>( d ) (feature dimension)</td>
</tr>
<tr>
<td>Frank-Wolfe</td>
<td>( \frac{1}{\epsilon} )</td>
<td>1</td>
</tr>
<tr>
<td>Ours</td>
<td>( \kappa \log \frac{1}{\epsilon} )</td>
<td>( s ) (optimal sparsity)</td>
</tr>
</tbody>
</table>
Tackle challenge 1: block Frank-Wolfe

1: **Input:** Data matrix $A \in \mathbb{R}^{n \times d}$, label matrix $b$, iteration $T$.
2: **Initialize:** $x_1 \leftarrow 0$.
3: **for** $t = 1, 2, \cdots, T - 1$ **do**
   4: \[
   \text{ProjectedGD: } \Delta x_t \leftarrow \text{argmin}_{\|\Delta x\|_1 \leq \tau} \left\{ \langle \nabla f(x_t), \Delta x \rangle + \frac{\beta}{2} \eta \|\Delta x - x_t\|_2^2 \right\}
   \]
5: \[
   x_{t+1} \leftarrow (1 - \eta)x_t + \eta \Delta x_t
   \]
6: **end for**
7: **Output:** $x_T$
Tackle challenge 1: block Frank-Wolfe

1: **Input:** Data matrix $A \in \mathbb{R}^{n \times d}$, label matrix $b$, iteration $T$.
2: **Initialize:** $x_1 \leftarrow 0$.
3: **for** $t = 1, 2, \cdots, T - 1$ **do**
4: 
   ProjectedGD: $\Delta x_t \leftarrow \text{argmin} \{ \langle \nabla f(x_t), \Delta x \rangle + \frac{\beta}{2} \eta \| \Delta x - x_t \|^2 \} \quad \| \Delta x \|_1 \leq \tau$
   
   FW: $\Delta x_t \leftarrow \text{argmin} \{ \langle \nabla f(x_t), \Delta x \rangle \} \quad \| \Delta x \|_1 \leq \tau$

5: 
   $x_{t+1} \leftarrow (1 - \eta)x_t + \eta \Delta x_t$

6: **end for**
7: **Output:** $x_T$
Tackle challenge 1: block Frank-Wolfe

1: **Input:** Data matrix \( A \in \mathbb{R}^{n \times d} \), label matrix \( b \), iteration \( T \).

2: **Initialize:** \( x_1 \leftarrow 0 \).

3: **for** \( t = 1, 2, \cdots, T - 1 \) **do**

4:

   - **ProjectedGD:** \( \Delta x_t \leftarrow \arg\min_{\|\Delta x\|_1 \leq \tau} \{ \langle \nabla f(x_t), \Delta x \rangle + \frac{\beta}{2} \|\Delta x - x_t\|^2 \} \)

   - **FW:** \( \Delta x_t \leftarrow \arg\min_{\|\Delta x\|_1 \leq \tau} \{ \langle \nabla f(x_t), \Delta x \rangle \} \)

   - **Ours:** \( \Delta x_t \leftarrow \arg\min_{\|\Delta x\|_1 \leq \tau, \|\Delta x\|_0 \leq s} \{ \langle \nabla f(x_t), \Delta x \rangle + \frac{\beta}{2} \|\Delta x - x_t\|^2 \} \)

5:

   \( x_{t+1} \leftarrow (1 - \eta)x_t + \eta \Delta x_t \)

6: **end for**

7: **Output:** \( x_T \)
Tackle challenge 2: reduce iteration complexity from partial updates

\[
\begin{align*}
\min_{x \in C \subset \mathbb{R}^d} \max_{y \in \mathbb{R}^n} \left\{ L(x, y) = f(x) + y^\top A x - g(y) \right\}
\end{align*}
\]

Write \( w = Ax \) and \( z = A^\top y \).  

For each iteration, 

<table>
<thead>
<tr>
<th>Operation</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute full gradient ( \partial_x L = A^\top y + f'(x) )</td>
<td>( O(nd) )</td>
</tr>
<tr>
<td>Conduct BlockFW on ( x ) to find ( s )-sparse update ( \Delta x )</td>
<td>( O(d) )</td>
</tr>
<tr>
<td>( x^+ \leftarrow (1 - \eta)x + \eta \Delta x )</td>
<td>( O(d) )</td>
</tr>
<tr>
<td>Greedy block-( k ) coordinate ascent for ( y )</td>
<td>( O(nd) )</td>
</tr>
</tbody>
</table>

Remark 1: take \( k = ns/d \) the iteration complexity is \( O(sn) \).  
Remark 2: the advantage comes from the fact that gradient could be maintained with the bilinear form.
Tackle challenge 2: reduce iteration complexity from partial updates

\[
\min_{x \in C \subseteq \mathbb{R}^d} \max_{y \in \mathbb{R}^n} \left\{ L(x, y) = f(x) + y^\top A x - g(y) \right\}
\]

Maintain \( w = Ax \) and \( z = A^\top y \).

For each iteration,

<table>
<thead>
<tr>
<th>Operation</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute full gradient ( \partial_x L = z + f'(x) )</td>
<td>( \mathcal{O}(d) )</td>
</tr>
<tr>
<td>Conduct BlockFW on ( x ) to find ( s )-sparse update ( \Delta x )</td>
<td>( \mathcal{O}(d) )</td>
</tr>
<tr>
<td>( x^+ \leftarrow (1 - \eta)x + \eta \Delta x )</td>
<td>( \mathcal{O}(d) )</td>
</tr>
<tr>
<td>( w^+ \leftarrow (1 - \eta)w + \eta A \Delta x )</td>
<td>( \mathcal{O}(sn) )</td>
</tr>
<tr>
<td>Greedy block-( k ) coordinate ascent for ( y ) and ( z )</td>
<td>( \mathcal{O}(kd) )</td>
</tr>
</tbody>
</table>

Remark 1: take \( k = ns/d \) the iteration complexity is \( \mathcal{O}(sn) \).

Remark 2: the advantage comes from the fact that gradient could be maintained with the bilinear form
## Time complexity comparisons

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Per Iteration Cost</th>
<th>Iteration Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank Wolfe</td>
<td>$O(nd)$</td>
<td>$O\left(\frac{1}{\epsilon}\right)$</td>
</tr>
<tr>
<td>Accelerated PGD (Nesterov et al. 2013)</td>
<td>$O(nd)$</td>
<td>$O\left(\sqrt{\kappa} \log \frac{1}{\epsilon}\right)$</td>
</tr>
<tr>
<td>SVRG (Rie et al. 2013)</td>
<td>$O(nd)$</td>
<td>$O\left((1 + \kappa/n) \log \frac{1}{\epsilon}\right)$</td>
</tr>
<tr>
<td>SCGS (Lan et al. 2016)</td>
<td>$O\left(\frac{\kappa^2 \text{#iter}^3 d}{\epsilon^2}\right)$</td>
<td>$O\left(\frac{1}{\epsilon}\right)$</td>
</tr>
<tr>
<td>STORC (Hazan et al. 2016)</td>
<td>$O(\kappa^2 d + nd)$</td>
<td>$O\left(\log \frac{1}{\epsilon}\right)$</td>
</tr>
<tr>
<td>Primal Dual FW (ours)</td>
<td>$O(ns)$</td>
<td>$O\left((1 + \kappa/n) \log \frac{1}{\epsilon}\right)$</td>
</tr>
</tbody>
</table>

Remark 1: $s$ is the sparsity of primal optimal induced by $\ell_1$ constraint.
Remark 2: for algorithm and complexity for nuclear norm constraints, refer to our paper to details.
Experiments

Compared methods: (1) Accelerated Projected Gradient Descent (Acc PG) (2) Frank-Wolfe algorithm (FW) (3) Stochastic Variance Reduced Gradient (SVRG) (4) Stochastic Conditional Gradient Sliding (SCGS) and (5) Stochastic Variance-Reduced Conditional Gradient Sliding (STORC)