Zeroth-order Frank-Wolfe Variants for Matrix Completion

Project for "Optimization for Data Science" 2021-22

Matteo Bergamaschi, Simone De Renzis, Andrea Marchini, Luca Veronese

Università degli Studi di Padova
Dipartimento di Matematica "Tullio Levi-Civita"

16th September, 2022

Università degli Studi di Padova

1. Introduction
2. Algorithms
3. Low Rank Matrix Completion
4. Implementation
5. Results

1. Introduction
2. Algorithms
3. Low Rank Matrix Completion
4. Implementation
5. Results

The project

- Low rank matrix completion for recommender systems
- Zeroth-order variants of the Frank-Wolfe method

Frank-Wolfe

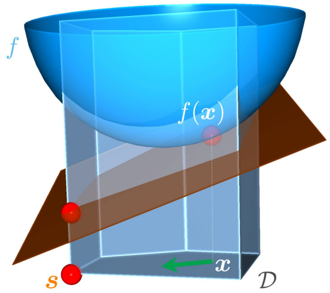
- Marguerite Frank and Philip Wolfe (1956)
- Iterative first-order optimization algorithm for problems with linear constraints:
$$\min_{x \in C} f(x)$$
- $f : \mathbb{R}^n \to \mathbb{R}$ convex and LCG with constant $L > 0$
- $C \subseteq \mathbb{R}^n$ convex and compact with diameter D

Idea

- Find a descent direction on the linear approximation of f in x_k:

$$\min_{x \in C} f(x_k) + \nabla f(x_k)^\top (x - x_k)$$

- Update the iterate
- No need to use projection



Pseudo-code

Algorithm 1 `Frank-Wolfe method`

```
Choose a point $x_1 \in C$
For $k = 1, \dots$
    Set $\hat{x}_k = \operatorname*{Argmin}_{x \in C} \nabla f(x_k)^\top (x - x_k)$
    If $\hat{x}_k$ satisfies some specific condition, then STOP
    Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k \in (0, 1]$
    suitably chosen stepsize
End for
```

Zeroth order

- Real-world application:
  - No access to the gradient
  - Slow to compute
- Zeroth order algorithms:
  - Use only function queries to approximate the gradient
  - Maintain good convergence rate, sometimes even better

Our work

- Convex case:
    - Deterministic Zeroth-order Frank-Wolfe (DZFW)
    - Stochastic Gradient Free Frank-Wolfe (3 variants)
    - First Order Frank-Wolfe (FOFW)
- Non-Convex case:
    - Stochastic Gradient Free Frank-Wolfe (SGFFW - I-RDSA)
    - Faster Zeroth-order Frank Wolfe (FZFW)
    - Faster Zeroth-order Conditional Gradient Sliding (FZCGS)

1. Introduction
2. Algorithms
3. Low Rank Matrix Completion
4. Implementation
5. Results

SGFFW - Stochastic Gradient Free Frank-Wolfe

- Described by Kar, Sahu and Zaheer in *Towards Gradient Free and Projection Free Stochastic Optimization*
- Every iteration involves m oracle calls
- The estimation of the gradient keeps trace of information coming from previous estimations

SGFFW algorithm

Algorithm 1 Stochastic Gradient Free Frank-Wolfe (SGFFW – I-RDSA)

Require: Input, Loss Function $F(x)$, Convex Set C, number of directions m, sequences $\gamma = \frac{2}{T^{3/4}}$,

$$(p_t, c_t) = \left(\frac{4}{1 + \frac{d}{m}^{1/3}(t+8)^{2/3}}, \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}} \right)$$

Output: $\mathbf{x}_T$

1: Initialize $\mathbf{x}_0 \in C$

2: **for do**

3: Compute
Sample $\{\mathbf{z}_{i,t}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I}_d)$,
$\mathbf{g}(\mathbf{x}_t, \mathbf{y}) = \sum_{i=1}^{d} \frac{F(\mathbf{x}_t + c_t \mathbf{z}_t; \mathbf{y}) - F(\mathbf{x}_t, \mathbf{y})}{c_t} \mathbf{z}_{i,t}$

4: Compute $\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \mathbf{g}(\mathbf{x}_t, \mathbf{y})$

5: Compute $\mathbf{v}_t = \mathrm{argmin}_{s \in C} \langle \mathbf{s}, \mathbf{d}_t \rangle$,

6: Compute $\mathbf{x}_{t+1} = (1 - \gamma)\mathbf{x}_t + \gamma \mathbf{v}_t$

7: **end for**

FZFW - Faster Zeroth-Order Frank-Wolfe Method

- Described by H. Huang and H. Gao in *Can Stochastic Zeroth-Order Frank-Wolfe Method Converge Faster for Non-Convex Problems?*
- Computes the estimation of the gradient every q iterations over all the components
- The remaining $q-1$ times the estimation is computed over a small number of components

FZFW Algorithm

Algorithm 1 Stochastic Gradient Free Frank-Wolfe (SGFFW – I-RDSA)

Require: Input, Loss Function $F(x)$, Convex Set C, number of directions m, sequences $\gamma = \frac{2}{T^{3/4}}$,

$$(p_t, c_t) = \left(\frac{4}{1 + \frac{d}{m}^{1/3}(t+8)^{2/3}}, \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}} \right)$$

Output: $\mathbf{x}_T$

1: Initialize $\mathbf{x}_0 \in C$
2: **for do**
3: Compute
Sample $\{\mathbf{z}_{i,t}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I}_d)$,
$\mathbf{g}(\mathbf{x}_t, \mathbf{y}) = \sum_{i=1}^{d} \frac{F(\mathbf{x}_t + c_t \mathbf{z}_t; \mathbf{y}) - F(\mathbf{x}_t, \mathbf{y})}{c_t} \mathbf{z}_{i,t}$
4: Compute $\mathbf{d}_t = (1 - \rho_t)\mathbf{d}_{t-1} + \rho_t \mathbf{g}(\mathbf{x}_t, \mathbf{y})$
5: Compute $\mathbf{v}_t = \text{argmin}_{s \in C} \langle \mathbf{s}, \mathbf{d}_t \rangle$,
6: Compute $\mathbf{x}_{t+1} = (1 - \gamma)\mathbf{x}_t + \gamma \mathbf{v}_t$
7: **end for**

FZCGS - Faster Zeroth-Order Conditional Gradient Method

Variant of FZFW with Conditional Gradient Sliding

Algorithm 3 $\mathbf{u}^+ = \text{condg}(\mathbf{g}, \mathbf{u}, \gamma, \eta)$ (Qu et al., 2017)

1: $\mathbf{u}_1 = \mathbf{u}$, $t = 1$

2: $\mathbf{v}_t$ be an optimal solution for

$$V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) = \max_{\mathbf{x}\in\Omega}\langle \mathbf{g} + \frac{1}{\gamma}(\mathbf{u}_t - \mathbf{u}), \mathbf{u}_t - \mathbf{x}\rangle$$

3: If $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) \leq \eta$, return $\mathbf{u}^+ = \mathbf{u}_t$.

4: Set $\mathbf{u}_{t+1} = (1 - \alpha_t)\mathbf{u}_t + \alpha_t \mathbf{v}_t$ where $\alpha_t = \min\{1, \frac{\langle \frac{1}{\gamma}(\mathbf{u}-\mathbf{u}_t)-\mathbf{g}, \mathbf{v}_t-\mathbf{u}_t\rangle}{\frac{1}{\gamma}\|\mathbf{v}_t-\mathbf{u}_t\|^2}\}$.

5: Set $t \leftarrow t + 1$ and goto step 2.

Convergence results

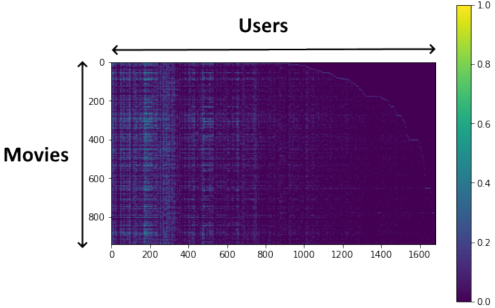
Algorithm	Number of oracle calls
SGFFW	$\mathcal{O}\left(\frac{d^{4/3}}{\epsilon^4}\right)$
FZFW	$\mathcal{O}\left(\frac{n^{1/2}d}{\epsilon^2}\right)$
FZCGS	$\mathcal{O}\left(\frac{n^{1/2}d}{\epsilon}\right)$

Table 1: Order of oracle calls to reach an ϵ-stationary point

1. Introduction
2. Algorithms
3. Low Rank Matrix Completion
4. Implementation
5. Results

MovieLens100k

- Movie ratings from 1 to 5
- Original dataset: list of tuples *[user,movie,rating]*
- Reshaped into a 2d matrix:



Low Rank Matrix Completion

- Most of movies are not rated
- Objective: infer the ratings in order to provide recommendation

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} \sum_{(i,j) \in \mathcal{O}} \left(X_{ij} - Y_{ij} \right)^2 \tag{1}$$

- Keep only meaningful recommendation
  - Constraint on the rank

$$\text{subject to} \quad rank(X) \leq R$$

- Rephrase the constraint with the nuclear norm

$$\|X\|_* \leq \delta$$

Low Rank Matrix Completion - 2

- (1) is the convex formulation of the problem
- Very simple gradient

$$\nabla f(X_{\mathcal{O}}) = (X - Y)_{\mathcal{O}}$$

- Main focus: robust objective function (non convex)

$$\min_{X \in \mathbb{R}^{m \times n}} \sum_{(i,j) \in \mathcal{O}} \left(1 - \exp\left\{ - \frac{(x_{ij} - y_{ij})^2}{\sigma^2} \right\}\right)$$
$$\text{subject to} \quad \|X\|_* \leq \delta$$

- Gradient not easy to calculate: Zeroth-Order methods

Linear Subproblem

$$\arg\min\left\{\mathrm{Tr}(\nabla f(X_{\mathcal{O}}) \cdot X) : \|X\|_* \leq \delta\right\}$$

- Solve the FW subproblem
  - Perform SVD decomposition on $-\nabla f(X_{\mathcal{O}})$
  - Obtain u_1 and v_1
  - Return $\delta u_1 v_1^T$
- At every iteration of the main FW routine
- At the beginning after random initialization

1 Introduction

2 Algorithms

3 Low Rank Matrix Completion

4 Implementation

5 Results

Code example

```python
def DZFW(Y, O, constr):
  """
  Deterministic Zeroth-order FW

  - Y: starting matrix
  - O: binary mask of observed values
  - constr: numerical value for the constraint on the nuclear norm
  """

  d0, d1 = Y.shape
  X0 = np.random.normal(0,1, (d0,d1)) # random initialization
  X0 = subproblem(X0,constr) # bring the initial iterate into the feasible set
  xs = [X0] # list that accumulates iterates

  for t in range(1, 100):
    stepsize = 1/(t+5)
    oracle = DZFW_oracle_call(Y, O, xs[-1], constr, stepsize) # oracle call
    x = (1 - stepsize) * xs[-1] + stepsize * (oracle - xs[-1])
    xs.append(x)
  return xs
```

- Python implementation for all the algorithms
- *NumPy* library for fast mathematical calculations

Execution time

Algorithm	CPU time (seconds)
FOFW	7.79
DZFW	9.10
SGFFW – KWSA	8.93
SGFFW – RDSA	23.51
SGFFW – I-RDSA	268.15

Table 2: CPU time for convex setting

Algorithm	CPU time (seconds)
SGFFW – I-RDSA	719.37
FZFW	26.48
FZCGS	15.49

Table 3: CPU time for non convex setting

Vectorization

```
for i in range(m):
  z_i = np.random.normal(0,1,(d0,d1))
  grad += (F(X + c_t * z_i) - F(X)) * z_i / c_t
return 1/m * grad
```

- *SGFFW – I-RDSA* algorithm requires `m` evaluations of gradient
- Unlike others, impossible to optimize with code vectorization
- Very slow

1. Introduction
2. Algorithms
3. Low Rank Matrix Completion
4. Implementation
5. Results

Convergence - Convex case

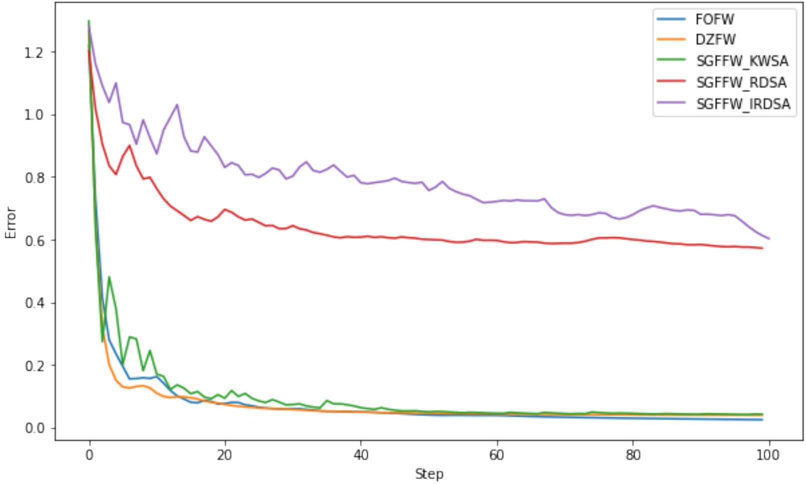


Figure 2: Convergence of methods for the convex case

Convergence - Non-convex case

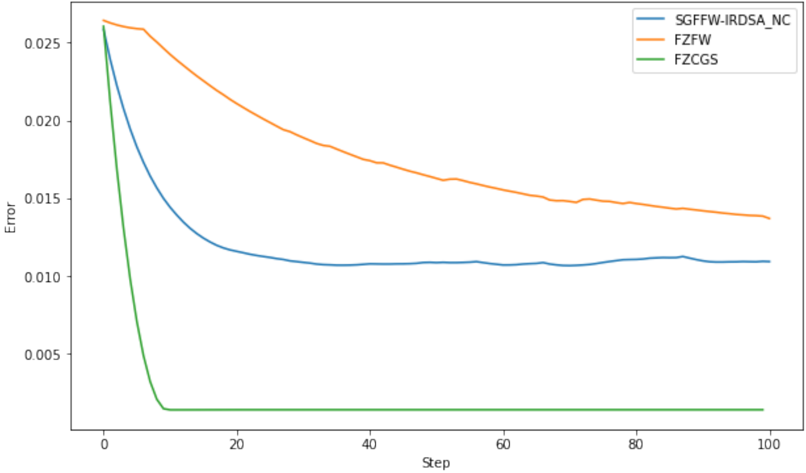
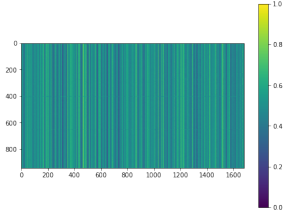
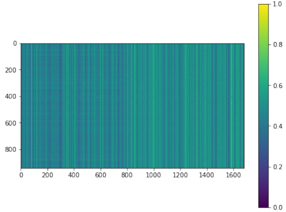


Figure 3: Convergence of methods for the non-convex case

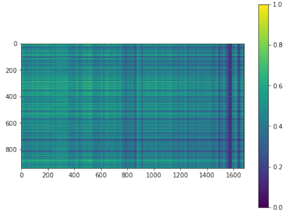
Generated matrices - Convex case



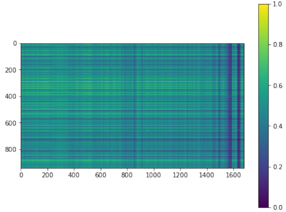
(a) SGFFW - RDSA



(b) SGFFW - I-RDSA

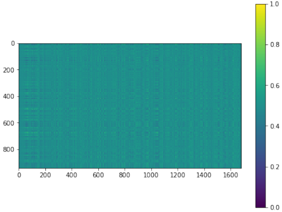


(c) SGFFW - KWSA

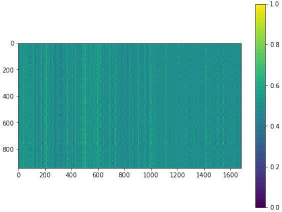


(d) Deterministic 0^{th} order

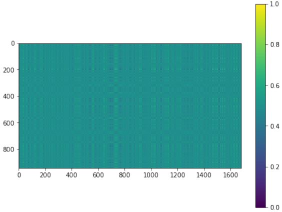
Generated matrices - Non-convex case



(a) SGFFW - I-RDSA



(b) FZFW



(c) FZCGS

Conclusions

- The plots of algorithms' convergence largely agree with those of (Sahu, Zaheer, and Kar 2019) and (Gao and Huang 2020)
- Experimental results have confirmed the effectiveness of the implemented methods for this low-rank matrix completion task

Gao, Hongchang and Heng Huang (2020). "Can Stochastic Zeroth-Order Frank-Wolfe Method Converge Faster for Non-Convex Problems?" In: *ICML'20*.

Sahu, Anit Kumar, Manzil Zaheer, and Soummya Kar (2019). "Towards Gradient Free and Projection Free Stochastic Optimization". In: *ArXiv* abs/1810.03233.