Using Taylor-Approximated Gradients to Improve the Frank-Wolfe Method for Empirical Risk Minimization

Zikai Xiong

zikai@mit.edu
https://zikaixiong.github.io/
MIT Operations Research Center

Joint work with Robert M. Freund

July 25, 2022

(Paper available soon)

Empirical Risk Minimization

Empirical risk minimization with "linear prediction"

$$\mathrm{ERM}_\ell : \quad \min_{x \in \mathcal{C} \subset \mathbb{R}^p} \quad F(x) := \frac{1}{n} \sum_{i=1}^{n} \left[f_i(x) = l_i(w_i^\top x) \right] \tag{1}$$

- The "linear prediction" means the model's losses are a function of $\{w_i^\top x\}_i$. This structural model was introduced by [1].
- $l_i(\cdot)$ is the univariate loss function of observation/sample i for $i \in [n] := \{1, \ldots, n\}$
- n is the number of observations/samples
- $\mathcal{C} \subset \mathbb{R}^p$ is a compact convex set
- p is the order (dimension) of the model variable x (the number of features)
- We are particularly interested in studying this problem when $n \gg 0$ is huge-scale (For instance, $n > p^2$)

Applications in Machine and Statistical Learning

Applications: Support vector machines (SVMs), LASSO, logistic regression, matrix completion, and others.

- LASSO:

$$\min_{x\in\mathbb{R}^p} \frac{1}{2n}\sum_{j=1}^{n}\left(y_j - w_j^T x\right)^2 \quad \text{s.t. } \|x\|_1 \le \lambda \ ,$$

  here $l_i(\cdot) := \frac{1}{2}(y_j - \cdot)^2$, and $\mathcal{C} := \{x : \|x\|_1 \le \lambda\}$.

- Matrix completion:

$$\min_{X\in\mathbb{R}^{n\times p}} \frac{1}{2|\Omega|}\sum_{(i,j)\in\Omega}\left(M_{i,j} - X_{i,j}\right)^2 \quad \text{s.t. } \|X\|_* \le \lambda \ ,$$

  here $l_{(i,j)}(\cdot) := \frac{1}{2}(\cdot - M_{i,j})^2$, and $\mathcal{C} := \{X : \|X\|_* \le \lambda\}$.

- Sparse logistic regression:

$$\min_{x\in\mathbb{R}} \ \frac{1}{n}\sum_{i=1}^{n}\ln(1+\exp(-y_i w_i^\top x)) \quad \text{s.t. } \|x\|_1 \le \lambda \ ,$$

  here $l_i(\cdot) := \ln(1+\exp(-y_i\cdot))$, and $\mathcal{C} := \{x : \|x\|_1 \le \lambda\}$.

Classic gradient decent method and Frank-Wolfe method

Classic gradient descent methods for ERM_ℓ

At iteration k :

1. Compute gradient $g^k := \nabla F(x^k) = \frac{1}{n}\sum_{i=1}^n l_i'(w_i^\top x^k)w_i$.
2. $x^{k+1} \leftarrow \arg\min_{x\in\mathcal{C}} \left\| x - (x^k - \gamma_k g^k) \right\|^2$, where $\gamma_t \in (0, +\infty)$.

Classic Frank-Wolfe methods for ERM_ℓ

At iteration k:

1. Compute gradient $g^k := \nabla F(x^k) = \frac{1}{n}\sum_{i=1}^n l_i'(w_i^\top x^k)w_i$.
2. Compute $s^k \leftarrow \text{lmo}(g^k)$, which is $\arg\min_{s\in\mathcal{C}} \langle g^k, s\rangle$.
3. Set $x^{k+1} \leftarrow x^k + \gamma_k(s^k - x^k)$, where $\gamma_t \in [0, 1]$.

Computing gradients requires at least $\mathcal{O}(np)$ flops, expensive when $n \gg p$.

What Should We Do When n is Huge?

Naïve SGD methods for ERM_ℓ

At iteration k :

1. Compute gradient estimator g^k
   - Sample $i \sim \mathcal{U}([n])$
   - $g^k \leftarrow l_i'(w_i^\top x^k) w_i$
2. $x^{k+1} \leftarrow \arg\min_{x \in \mathcal{C}} \left\| x - (x^k - \gamma_k g^k) \right\|$, where $\gamma_t \in (0, +\infty)$.

Convergence under proper assumptions.

Naïve stochastic Frank-Wolfe methods for ERM_ℓ

At iteration k:

1. Compute gradient estimator g^k
   - Sample $i \sim \mathcal{U}([n])$
   - $g^k \leftarrow l_i'(w_i^\top x^k) w_i$
2. Compute $s^k \leftarrow \text{lmo}(g^k)$.
3. Set $x^{k+1} \leftarrow x^k + \gamma_k (s^k - x^k)$, where $\gamma_t \in [0, 1]$.

Cannot work without increasing batch-size, which is expensive!

Other Stochastic Frank-Wolfe Methods

When loss functions are convex:

- Baseline (the classic deterministic Frank-Wolfe method [2]): $\mathcal{O}(n/\varepsilon)$
- Lan and Zhou (2016) [3]: $\mathcal{O}(1/\varepsilon^2)$
- Hazan and Luo (2016) [4]: $\mathcal{O}(n \log(1/\varepsilon) + 1/\varepsilon^{3/2})$
- Yurtsever, Sra and Cevher (2019) [5]: $\mathcal{O}(1/\varepsilon^2)$
- Mokhtari, Hassani and Karbasi (2020) [6]: $\mathcal{O}(1/\varepsilon^3)$
- Lu and Freund (2021) [1]: $\mathcal{O}(n/\varepsilon + n/\sqrt{\varepsilon})$
- Négiar, Dresdner, Tsai, Ghaoui, Locatello, Freund and Pedregosa (2020) [7]: $\mathcal{O}(n/\varepsilon + n^{3/2}/\sqrt{\varepsilon})$

When loss functions are nonconvex:

- Baseline (the deterministic Frank-Wolfe method [8]): $\mathcal{O}(n/\varepsilon^2)$
- Reddi, Sra, Póczos and Smola (2016) [9]: $\mathcal{O}(n + n^{2/3}/\varepsilon^2)$
- Yurtsever, Sra and Cevher (2019) [5]: $\mathcal{O}(n^{1/2}/\varepsilon^2)$
- Shen, Fang, Zhao, Huang and Qian (2019) [10]: $\mathcal{O}(n^{3/4}p/\varepsilon^{3/2} + p/\varepsilon^2)$
- Zhang, Shen, Mokhtari, Hassani and Karbasi (2020) [11]: $\mathcal{O}(p/\varepsilon^3)$
- Hassani, Karbasi, Mokhtari and Shen (2020) [12]: $\mathcal{O}(p/\varepsilon^3)$

We omit a factor of p for each method's complexity.

	SGD	**Stochastic FW**
1)	Many results on reducing the influence of n while maintaining good dependence on $1/\varepsilon$;	Linear dependence on n while maintaining good dependence on $1/\varepsilon$;
2)	Requires projections, which can be expensive;	Replaces projections by linear minimization, which is cheaper for many applications;
3)	Does not promote structured solutions (sparsity, low-rank).	Promotes structured solutions.

Our paper addresses the following question:

Question: Are there efficient stochastic (or deterministic) Frank-Wolfe methods that eliminate or reduce the dependence on the number of observations n, in theory and in practice?

Theoretical Results

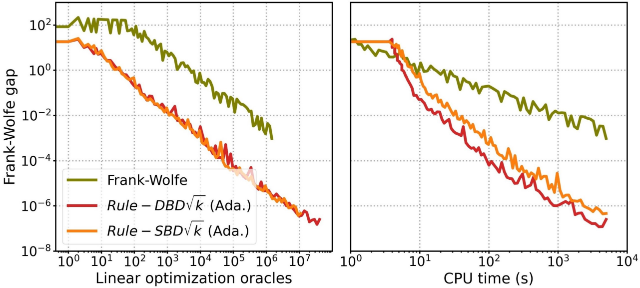
We develop stochastic and deterministic Frank-Wolfe methods. Both of them can reduce the dependence on n.

- For convex objectives: $\mathcal{O}(n/\varepsilon) \to \mathcal{O}(p/\varepsilon + np/\sqrt{\varepsilon})$;
- for nonconvex objectives: $\mathcal{O}(n/\varepsilon^2) \to \mathcal{O}(p/\varepsilon^2 + np/\varepsilon^{3/2})$,

here $n \gg p$.

Practical Performance

Taking the convex case as an example, all of our 6 variants largely outperform the classic Frank-Wolfe method. Results of sparse logistic regression on dataset a9a ($n = 32561,\ p = 123$)



Empirical risk minimization with linear prediction

$$\mathrm{ERM}_\ell : \quad \min_{x \in \mathcal{C} \subset \mathbb{R}^p} \quad F(x) := \frac{1}{n} \sum_{i=1}^{n} \left[f_i(x) = l_i(w_i{}^\top x) \right]$$

Assumption 1

1. $\mathcal{C}$ is compact and convex,
2. *lmo* on $\mathcal{C}$ can be easily solved,
3. for any i. $l_i(\cdot)$ is twice-differentiable and $l_i'(\cdot)$ is L-Lipschitz continuous on the range of $w_i{}^\top x$ over $x \in \mathcal{C}$, namely
$$|l_i'(\bar{\theta}) - l_i'(\hat{\theta})| \leq L|\bar{\theta} - \hat{\theta}| \quad \text{for any } \bar{\theta}, \hat{\theta} \in w_i(\mathcal{C}), \text{ and}$$
4. for any i, $l_i''(\cdot)$ is $\hat{L}$-Lipschitz continuous on the range of $w_i{}^\top x$ over $x \in \mathcal{C}$, namely
$$|l_i''(\bar{\theta}) - l_i''(\hat{\theta})| \leq \hat{L}|\bar{\theta} - \hat{\theta}| \quad \text{for any } \bar{\theta}, \hat{\theta} \in w_i(\mathcal{C}).$$

Examples of Lipschitz Constants

- $\hat{L} = 0$:
  - LASSO;
  - Matrix completion with quadratic losses;
  - Structured sparse matrix estimation with CUR factorization;
  - See [13, 1].
- $L \leq 1/4,\ \hat{L} \leq 1/6\sqrt{3}$:
  - Logistic regression for binary classification

$$\min_{x \in \mathcal{C}} \ F(x) := \frac{1}{n} \sum_{i=1}^{n} \ln(1 + \exp(-y_i w_i^\top x)) \ .$$

- $L \leq 2L_\sigma^2 + 2L_\sigma,\ \hat{L} \leq 6L_\sigma^2 + 2L_\sigma$:
  - Binary linear classification with non-convex losses.

$$\min_{x \in \mathcal{C}} \ \frac{1}{n} \sum_{i=1}^{n} (y_i - \sigma(w_i^\top x))^2 \ , \text{ where } \sigma(\cdot) = \frac{1}{1 + \exp(\cdot)}$$

and L_σ is the upper bound of $|\sigma|$, $|\sigma'|$, and $|\sigma''|$. See [14].

Taylor-approximated Gradients

Classic approach of computing $\nabla F(x^k)$:

1. Calculate $l_i'(w_i^\top x^k)$ for each $i \in [n]$
2. and then $\nabla F(x^k) := \frac{1}{n}\sum_{i=1}^n l_i'(w_i^\top x^k) w_i$.

Taylor-approximated gradient estimator g^k for $\nabla F(x^k)$:

1. Since
$$l_i'(w_i^\top x^k) \approx l_i'(w_i^\top b_i) + l_i''(w_i^\top b_i)(w_i^\top x^k - w_i^\top b_i) \ ,$$
if $b_i \in \mathcal{C}$ is the "Taylor-point" for the Taylor approximation,
2. then for each $i \in [n]$,
$$g_i^k := \left(l_i'(w_i^\top b_i) + l_i''(w_i^\top b_i)(w_i^\top x^k - w_i^\top b_i) \right) w_i \ ,$$
and
$$g^k := \frac{1}{n}\sum_{i=1}^n g_i^k \ .$$

Taylor-point Updating Frank-Wolfe (TUFW)

Classic Frank-Wolfe methods for ERM_ℓ

At iteration k:

1. Compute gradient $g^k := \nabla F(x^k) = \frac{1}{n}\sum_{i=1}^n l_i'(w_i^\top x^k) w_i$.
2. Compute $s^k \leftarrow \text{lmo}(g^k)$, which is $\arg\min_{s\in\mathcal{C}} \langle g^k, s\rangle$.
3. Set $x^{k+1} \leftarrow x^k + \gamma_k(s^k - x^k)$, where $\gamma_t \in [0,1]$.

Taylor-point Updating Frank-Wolfe (TUFW) for ERM_ℓ

Initialize anchor points: $b_i \leftarrow x^0$ for $i \in \{1,\ldots,n\}$. And then at iteration k:

1. Compute Taylor-approximated gradient estimator g^k
   - Update anchor points:
     use some **Rule** to create a set $\mathcal{B}_k \subset \{1,\ldots,n\}$, and then $b_i \leftarrow x^k$ for $i \in \mathcal{B}_k$
   - Compute estimates of individual gradients:

$$g^k := \frac{1}{n}\sum_{i=1}^n \left[g_i^k := \left(l_i'(w_i^\top b_i) + l_i''(w_i^\top b_i)(w_i^\top x^k - w_i^\top b_i) \right) w_i \right]$$

2. Compute $s^k \leftarrow \text{lmo}(g^k)$, which is $\arg\min_{s\in\mathcal{C}} \langle g^k, s\rangle$.
3. Set $x^{k+1} \leftarrow x^k + \gamma_k(s^k - x^k)$, where $\gamma_t \in [0,1]$.

Incrementally updating

The gradient estimator in TUFW is $g^k = q_k + H_k x^k$, where

$$q_k := \frac{1}{n}\sum_{i=1}^{n} l_i'(w_i^\top b_i)w_i - l_i''(w_i^\top b_i)w_i w_i^\top b_i \ ,$$
$$H_k := \frac{1}{n}\sum_{j=1}^{n} l_i''(w_i^\top b_i)w_i w_i^\top .$$

By incrementally updating q_k and H_k instead of summing all the n components up every time, calculating g^k only needs

$$\mathcal{O}(p^2(|\mathcal{B}_k| + 1)) \text{ flops}$$

and the memory needed is

$$\mathcal{O}(n + p^2) \text{ for ERM}_\ell \ .$$

Taylor-point Updating Rules

Stochastic Batch-size Decreasing ("SBD")

Rule–SBD$\sqrt{k}$ (Convex):

$\mathcal{B}_k$ is comprised of β_k samples from $\mathcal{U}(\{1, \ldots, n\})$, where $\beta_k := n/\sqrt{k}$.

Rule–SBD$\sqrt[4]{K}$ (Nonconvex):

Similar to above with $\beta_k := n/\sqrt[4]{K}$, with fixed number of overall iterations K.

Deterministic Batch-frequency Decreasing ("DBD")

Rule–DBD$\sqrt{k}$ (Convex): $\mathcal{B}_k = \begin{cases} [n] & \text{if } \sqrt{k} \in \mathbb{N} \\ \emptyset & \text{if } \sqrt{k} \notin \mathbb{N} \end{cases}.$

Rule–DBD$\sqrt[4]{K}$ (Nonconvex): $\mathcal{B}_k = \begin{cases} [n] & \text{if } k/\lfloor \sqrt[4]{K} \rfloor \in \mathbb{N} \\ \emptyset & \text{if } k/\lfloor \sqrt[4]{K} \rfloor \notin \mathbb{N} \end{cases}.$

No updating

Rule–$\emptyset$ ($\hat{L} = 0$): $\mathcal{B}_k = \begin{cases} [n] & \text{if } k = 0 \\ \emptyset & \text{if } k \neq 0 \end{cases}.$

...

Empirical risk minimization with linear prediction

$$\text{ERM}_\ell : \quad \min_{x \in \mathcal{C} \subset \mathbb{R}^p} \quad F(x) := \frac{1}{n} \sum_{i=1}^{n} \left[f_i(x) = l_i(w_i{}^\top x) \right]$$

- Diameter of $\mathcal{C}$:

$$D := \max_{x,y \in \mathcal{C}} \|x - y\| < \infty$$

- "q-Diameter" of $\mathcal{C}$:

$$D_q := \max_{x,y \in \mathcal{C}} \left\| W^\top (x - y) \right\|_q < \infty \text{ for } q \in [1, \infty],$$

  where $W := (w_1, \ldots, w_n) \in \mathbb{R}^{p \times n}$. The q-diameter is a common metric used for ERM_ℓ. See [1, 7].

- Under the boundedness assumption of feature vectors (there exists $M < \infty$, such that $\|w_i\|_* \leq M$ for any i), then

$$D_q \leq n^{1/q} M D.$$

  In the worst case, there is an opaque $n^{1/q}$ in q-diameter D_q.

Results for Convex Objective

Classic Frank-Wolfe (for $F(x^k) - F(x^\star) \leq \varepsilon$):

$$\mathcal{O}\left((\mathrm{fLMO} + np) \left[\frac{LM^2 D^2}{\varepsilon} \right] \right) \ . \tag{2}$$

Here fLMO denotes the complexity of a *lmo* call.

Classic Frank-Wolfe (for $F(x^k) - F(x^\star) \le \varepsilon$):

$$\mathcal{O}\left((\mathrm{fLMO} + np) \left[\frac{LM^2 D^2}{\varepsilon} \right] \right) \quad . \tag{2}$$

Here fLMO denotes the complexity of a *lmo* call.

Theorem ($Rule{-}SBD\sqrt{k}$)

Suppose that F is convex and Assumption 1 holds, and TUFW with $Rule{-}SBD\sqrt{k}$ is applied to ERM_ℓ with step-sizes $\gamma_k := 2/(k+2)$. Then:

$$\mathbb{E}[F(x^k) - F(x^\star)] \le \frac{2LD_2^2 + 134\hat{L}D_1 D_\infty^2}{n(k+1)} \quad . \tag{3}$$

Complexity of obtaining $\mathbb{E}[F(x^k) - F(x^\star)] \le \varepsilon$ is

$$O\left((\mathrm{fLMO} + p^2) \left[\frac{LM^2 D^2 + \hat{L}M^3 D^3}{\varepsilon} \right] + np^2 \left[\frac{\sqrt{LM^2 D^2 + \hat{L}M^3 D^3}}{\sqrt{\varepsilon}} \right] \right).$$

Results for Convex Objective

Method	Overall Complexity
Frank-Wolfe	$\mathcal{O}\left((\mathrm{fLMO} + np) \cdot \frac{c_1}{\varepsilon}\right)$
Négiar et al. [7]	$\mathcal{O}\left((n \cdot \mathrm{fLMO} + np)\left[\frac{c_1}{\varepsilon} + \frac{\sqrt{F(x^0) - F(x^\star)}}{n\sqrt{\varepsilon}} + \frac{\sqrt{nc_1}}{\sqrt{\varepsilon}}\right]\right)$
$Rule{-}SBD\sqrt{k}$	$\mathcal{O}\left((\mathrm{fLMO} + p^2) \cdot \frac{c_1 + c_2}{\varepsilon} + np^2 \cdot \frac{\sqrt{c_1 + c_2}}{\sqrt{\varepsilon}}\right)$
$Rule{-}DBD\sqrt{k}$	$\mathcal{O}\left((\mathrm{fLMO} + p^2) \cdot \frac{c_1 + c_2}{\varepsilon} + np^2 \cdot \frac{\sqrt{c_1 + c_2}}{\sqrt{\varepsilon}}\right)$

- Here fLMO denotes the cost of a *lmo*.
- $c_1 := LM^2D^2$ and $c_2 := \hat{L}M^3D^3$.

Let $x^{k+1}(\gamma) := x^k + \gamma(s^k - x^k)$, then

$$F(x^{k+1}(\gamma)) = F(x^k) + \gamma(g^k)^\top(s^k - x^k) + \frac{\gamma^2}{2}(s^k - x^k)^\top H_k(s^k - x^k) \\ + \frac{1}{n}\sum_{i=1}^n \int_{t=0}^{\gamma} \Big(\nabla f_i(x^k + t(s^k - x^k)) - \nabla f_i(b_i) \\ - \nabla^2 f_i(b_i)\big(x^k + t(s^k - x^k) - b_i\big)\Big)^\top (s^k - x^k)dt.$$

The $\arg\min_{\gamma\in[0,\gamma_k]} F(x^{k+1}(\gamma))$ is approximated by the following $\tilde{\gamma}_k$:

Adaptive step-sizes

Let γ_k be the normal step-size. The adaptive step-size is defined by

$$\arg\min_{\gamma\in[0,\gamma_k]} F(x^k) + \gamma(g^k)^\top(s^k - x^k) + \frac{\gamma^2}{2}(s^k - x^k)^\top H_k(s^k - x^k),$$

which has closed-form solution

$$\tilde{\gamma}_k := \begin{cases} \min\left\{\gamma_k, \frac{(g^k)^\top(x^k - s^k)}{(s^k - x^k)^\top H_k(s^k - x^k)}\right\} & \text{when } (s^k - x^k)^\top H_k(s^k - x^k) > 0\ , \\ \gamma_k & \text{when } (s^k - x^k)^\top H_k(s^k - x^k) \le 0\ . \end{cases}$$

Measure of Performance in Experiments

Frank-Wolfe gap

For $x \in \mathcal{C}$, the Frank-Wolfe gap of F on x is defined by

$$\mathcal{G}(x) := \max_{s \in \mathcal{C}} \langle x - s, \nabla F(x) \rangle \ .$$

- ▷ It is an upper bound of optimality gap $F(x^k) - F(x^\star)$, for convex F.
- ▷ Since it is easier to compute than the optimality gap $F(x^k) - F(x^\star)$, it is a commonly used performance measure.
- ▷ It is also a measure of stationarity for constrained nonconvex problems.

Sparse Logistic Regression Problems

Experiments are conducted on

$$\min_{x} \; F(x) := \frac{1}{n}\sum_{i=1}^{n} \ln(1+\exp(-y_i w_i^\top x)) \quad \text{s.t. } \|x\|_1 \le \lambda \; .$$

- We chose the large-scale datasets with $n \gg p$ from SVMLIB.
- The λ is chosen by cross-validation, and we also tested $\lambda' := 100\lambda$.
- The performance measure is Frank-Wolfe gap.
- We tested TUFW with standard step-size γ_k and adaptive step-size $\tilde{\gamma}_k$.
- To be fair, all methods did not exploit the benefits of sparse structure.
- Implemented in Python, on MIT Engaging Cluster.

Comparison with Classic Frank-Wolfe

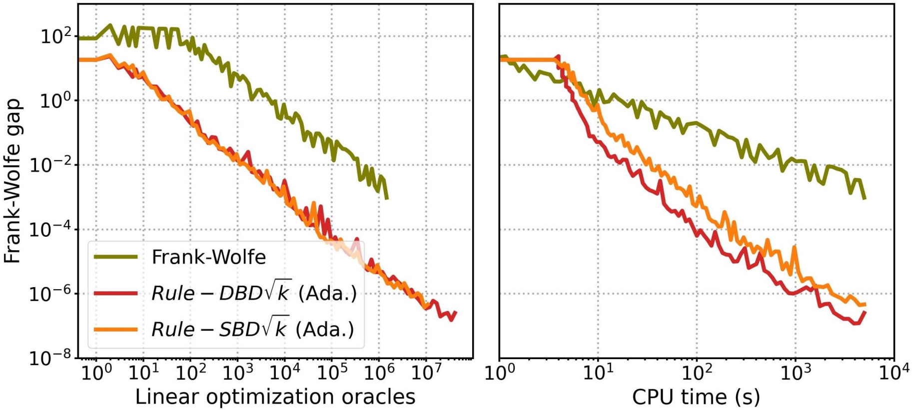


Figure 1: Sparse logistic regression on a9a ($n = 32561,\ p = 123$)

- ▷ Both TUFW methods can largely outperform the classic Frank-Wolfe method.
- ▷ Better than the theory indicates, the TUFW's *lmo* complexity is also better than the classic Frank-Wolfe method's in practice.

Comparison with More Methods

- FW – the classic Frank-Wolfe;
- FW-ada – the classic Frank-Wolfe method with adaptive step, $\tilde{\gamma}_k := \min\{1, \mathcal{G}(x^k)/L\|x^k - s^k\|^2\}$;
- SPIDER-FW – the Frank-Wolfe method with stochastic path-integrated differential estimator technique proposed by Yurtsever, Sra & Cevher [5];
- CSFW – the constant batch-size stochastic Frank-Wolfe proposed by Négiar, Dresdner, Tsai, Ghaoui, Locatello, Freund & Pedregosa [7].

CPU Time Comparison with More Methods (with Cross-validated λ)

Runtime required for $\mathcal{G}(x^k) \leq \varepsilon$ ("-" indicates using more than 5000 seconds)

ε	dataset	n	p	$Rule-SBD\sqrt{k}$	$Rule-DBD\sqrt{k}$	FW	FW-ada	SPIDER-FW	CSFW	Speedup
1e-1	a1a	1605	123	0.73	**0.37**	2.02	1.18	348.58	5.03	**3.20**
1e-3	a1a	1605	123	14.40	**5.14**	138.87	380.10	-	160.02	**27.02**
1e-5	a1a	1605	123	3029.61	**1034.66**	2956.50	-	-	-	**2.86**
1e-1	a2a	2265	123	1.02	**0.52**	3.03	1.55	358.72	5.47	**2.97**
1e-3	a2a	2265	123	21.24	**8.16**	197.80	472.72	-	167.66	**20.54**
1e-5	a2a	2265	123	2039.75	**631.23**	-	-	-	-	-
1e-1	a8a	22696	123	7.75	**4.66**	78.18	12.83	443.85	15.22	**2.75**
1e-3	a8a	22696	123	51.29	**23.71**	-	3178.02	-	1250.12	**52.72**
1e-5	a8a	22696	123	420.64	**174.87**	-	-	-	-	-
1e-1	a9a	32561	123	11.26	**6.80**	94.50	21.00	440.73	19.82	**2.91**
1e-3	a9a	32561	123	65.25	**31.22**	-	3844.13	-	1660.13	**53.17**
1e-5	a9a	32561	123	490.36	**197.15**	-	-	-	-	-
1e-1	w1a	2477	300	10.45	**5.71**	17.51	331.22	3566.83	44.39	**3.07**
1e-3	w1a	2477	300	51.51	**21.93**	327.38	-	-	1219.03	**14.93**
1e-5	w1a	2477	300	1743.08	**541.71**	-	-	-	-	-
1e-1	w2a	3470	300	13.11	**6.80**	37.34	574.42	-	54.06	**5.50**
1e-3	w2a	3470	300	128.14	**50.77**	369.82	-	-	1207.01	**7.28**
1e-5	w2a	3470	300	-	**1422.93**	-	-	-	-	-
1e-1	w7a	24692	300	88.86	**55.43**	541.10	-	-	129.19	**2.33**
1e-3	w7a	24692	300	320.17	**178.87**	-	-	-	4413.77	**24.68**
1e-5	w7a	24692	300	3106.98	**1516.83**	-	-	-	-	-
1e-1	w8a	49749	300	174.89	**114.66**	1198.35	-	-	212.80	**1.86**
1e-3	w8a	49749	300	553.94	**355.23**	-	-	-	-	-
1e-5	w8a	49749	300	-	**2707.92**	-	-	-	-	-
1e-1	svmguide3	1243	22	0.09	**0.02**	1.35	0.31	21.46	2.06	**13.43**
1e-3	svmguide3	1243	22	8.01	**2.64**	53.64	175.49	-	48.85	**18.53**
1e-5	svmguide3	1243	22	1132.50	**410.63**	629.39	-	-	1484.23	**1.53**
1e-7	svmguide3	1243	22	-	-	-	-	-	-	-
1e-1	phishing	11055	68	0.47	0.38	0.01	0.71	0.01	**0.01**	0.02
1e-3	phishing	11055	68	2.81	1.24	0.67	201.65	**0.32**	0.53	0.26
1e-5	phishing	11055	68	45.36	**16.10**	47.01	-	1391.81	43.70	**2.71**
1e-7	phishing	11055	68	2478.92	**812.38**	3370.12	-	-	-	**4.15**
1e-1	ijcnn1	49990	22	0.85	0.24	1.21	9.02	**0.22**	0.92	0.91
1e-3	ijcnn1	49990	22	4.15	**0.86**	54.72	565.14	193.34	71.65	**63.40**
1e-5	ijcnn1	49990	22	7.67	**1.66**	-	2322.76	-	-	**1402.31**
1e-7	ijcnn1	49990	22	10.20	**2.32**	-	3910.71	-	-	**1688.65**
1e-1	covtype	581012	54	54.62	29.76	143.55	123.79	**5.47**	18.91	0.18
1e-3	covtype	581012	54	552.47	**231.70**	2964.49	-	843.60	690.78	**2.98**
1e-5	covtype	581012	54	-	**3777.45**	-	-	-	-	-

CPU Time Comparison with More Methods (On Larger Regions $\lambda' = 100\lambda$)

Runtime required for $\mathcal{G}(x^k) \leq \varepsilon$ ("-" indicates using more than 5000 seconds)

ε	dataset	n	p	$Rule{-}SBD\sqrt{k}$	$Rule{-}DBD\sqrt{k}$	FW	FW-ada	SPIDER-FW	CSFW	Speedup
1e0	a1a	1605	123	1.08	**0.56**	3322.66	21.88	-	-	**39.14**
1e-2	a1a	1605	123	61.47	**22.15**	-	2671.01	-	-	**120.59**
1e-4	a1a	1605	123	4561.28	**1683.57**	-	-	-	-	-
1e0	a2a	2265	123	**2.40**	4.30	3858.43	32.69	-	-	**13.64**
1e-2	a2a	2265	123	6.70	**5.68**	-	1959.43	-	-	**345.13**
1e-4	a2a	2265	123	28.72	**13.29**	-	-	-	-	-
1e0	a8a	22696	123	15.35	**8.50**	-	268.54	-	-	**31.59**
1e-2	a8a	22696	123	34.86	**17.75**	-	-	-	-	-
1e-4	a8a	22696	123	60.72	**30.10**	-	-	-	-	-
1e0	a9a	32561	123	20.80	**10.97**	-	326.83	-	-	**29.79**
1e-2	a9a	32561	123	52.70	**24.35**	-	-	-	-	-
1e-4	a9a	32561	123	97.60	**45.91**	-	-	-	-	-
1e0	w1a	2477	300	16.17	**8.94**	-	4569.30	-	-	**511.10**
1e-2	w1a	2477	300	75.05	**34.01**	-	-	-	-	-
1e-4	w1a	2477	300	1687.82	**560.34**	-	-	-	-	-
1e0	w2a	3470	300	34.13	**18.39**	-	-	-	-	-
1e-2	w2a	3470	300	138.82	**65.27**	-	-	-	-	-
1e-4	w2a	3470	300	2311.84	**778.48**	-	-	-	-	-
1e0	w7a	24692	300	147.81	**123.91**	-	-	-	-	-
1e-2	w7a	24692	300	443.15	**339.27**	-	-	-	-	-
1e-4	w7a	24692	300	3434.91	**1740.40**	-	-	-	-	-
1e0	w8a	49749	300	522.63	**165.85**	-	-	-	-	-
1e-2	w8a	49749	300	1053.55	**515.06**	-	-	-	-	-
1e-4	w8a	49749	300	-	**4608.20**	-	-	-	-	-
1e-1	svmguide3	1243	22	3.58	**1.17**	-	127.60	-	-	**109.24**
1e-3	svmguide3	1243	22	11.28	**3.67**	-	485.90	-	-	**132.22**
1e-5	svmguide3	1243	22	18.83	**6.08**	-	844.46	-	-	**138.80**
1e-7	svmguide3	1243	22	26.37	**8.54**	-	1201.28	-	-	**140.65**
1e-1	phishing	11055	68	2.26	**1.20**	66.23	254.91	3563.61	64.96	**53.93**
1e-3	phishing	11055	68	5.15	**2.45**	3958.88	4057.22	-	-	**1613.39**
1e-5	phishing	11055	68	19.12	**8.35**	-	-	-	-	-
1e-7	phishing	11055	68	592.44	**207.94**	-	-	-	-	-
1e-1	ijcnn1	49990	22	1.52	**0.47**	-	100.76	-	-	**213.41**
1e-3	ijcnn1	49990	22	2.32	**0.64**	-	243.63	-	-	**378.96**
1e-5	ijcnn1	49990	22	2.92	**0.80**	-	425.17	-	-	**531.71**
1e-7	ijcnn1	49990	22	3.45	**0.92**	-	607.30	-	-	**660.82**
1e-1	covtype	581012	54	250.33	**115.22**	-	-	-	-	-
1e-3	covtype	581012	54	1163.77	**484.39**	-	-	-	-	-
1e-5	covtype	581012	54	2230.09	**890.50**	-	-	-	-	-

Comments

- The two TUFW methods have huge advantages over the other methods in practice.
- As $1/\varepsilon$ increases, the dependence of TUFW's complexity on n sharply decreases.
- Better than the theories indicate, the TUFW methods with adaptive step-sizes exhibit much faster convergence while other methods become slower on larger feasible regions.

Theorem for Nonconvex Objective

Classic Frank-Wolfe (for $\mathbb{E}_{\hat{k}\sim U[K]}[\mathcal{G}(x^{\hat{k}})] \leq \varepsilon$):

$$\mathcal{O}\left((\mathrm{fLMO} + np)\left[\frac{(F(x^0) - F(x^*)) + LM^2D^2}{\varepsilon^2}\right]\right) \quad . \tag{4}$$

Theorem for Nonconvex Objective

Classic Frank-Wolfe (for $\mathbb{E}_{\hat{k}\sim U[K]}[\mathcal{G}(x^{\hat{k}})] \le \varepsilon$):

$$\mathcal{O}\left((\text{fLMO} + np)\left[\frac{(F(x^0) - F(x^*)) + LM^2D^2}{\varepsilon^2} \right] \right) . \tag{4}$$

Theorem ($Rule{-}SBD\sqrt[4]{K}$)

Suppose Assumption 1 holds, and TUFW with $Rule{-}SBD\sqrt[4]{K}$ is applied to ERM_ℓ with step-sizes $\gamma_k := 1/\sqrt{K+1}$, where K is the fixed number of overall iterations. Then:

$$\sum_{k=0}^{K} \frac{\mathbb{E}\mathcal{G}(x^k)}{K+1} \le \frac{\varepsilon_0}{\sqrt{K+1}} + \frac{3\hat{L}D_1D_\infty^2 + LD_2^2}{2n\sqrt{K+1}} . \tag{5}$$

Complexity of obtaining $\mathbb{E}_{\hat{k}\sim U[K]}\mathbb{E}[\mathcal{G}(x^{\hat{k}})] \le \varepsilon$:

$$\mathcal{O}\left((\text{fLMO} + p^2)\left[\frac{((F(x^0) - F(x^*)) + LM^2D^2 + \hat{L}M^3D^3)^2}{\varepsilon^2} \right] \right.$$
$$\left. + np^2\left[\frac{((F(x^0) - F(x^*)) + LM^2D^2 + \hat{L}M^3D^3)^{3/2}}{\varepsilon^{3/2}} \right] \right) .$$

Results for Nonconvex Objective

Method	Overall Complexity
Frank-Wolfe [8]	$O\left((\mathrm{fLMO} + np) \cdot \frac{(\varepsilon_0 + c_1)^2}{\varepsilon^2}\right)$
Rule−SBD $\sqrt[4]{K}$	$O\left((\mathrm{fLMO} + p^2) \cdot \frac{(\varepsilon_0 + c_1 + c_2)^2}{\varepsilon^2} + np^2 \cdot \frac{(\varepsilon_0 + c_1 + c_2)^{3/2}}{\varepsilon^{3/2}}\right)$
Rule−DBD $\sqrt[4]{K}$	$O\left((\mathrm{fLMO} + p^2) \cdot \frac{(\varepsilon_0 + c_1 + c_2)^2}{\varepsilon^2} + np^2 \cdot \frac{(\varepsilon_0 + c_1 + c_2)^{3/2}}{\varepsilon^{3/2}}\right)$

- Here fLMO denotes the cost of a *lmo*.
- $c_1 := LM^2D^2$, $c_2 := \hat{L}M^3D^3$, and $\varepsilon_0 := F(x^0) - F(x^\star)$.

Computational Experiments on Binary Classification Problems with Nonconvex Losses

Binary Classification Problems with Nonconvex Losses

$$\min_{x} \ \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{1}{1+\exp(-w_i^\top x)}\right)^2 \quad \text{s.t. } \|x\|_1 \leq \lambda \ ,$$

Binary Classification with Nonconvex Losses

For each $K \in 10 \times \{2^0, 2^1 \ldots, 2^{20}\}$, we conduct 5 independent trials of each method for at most K iterations.

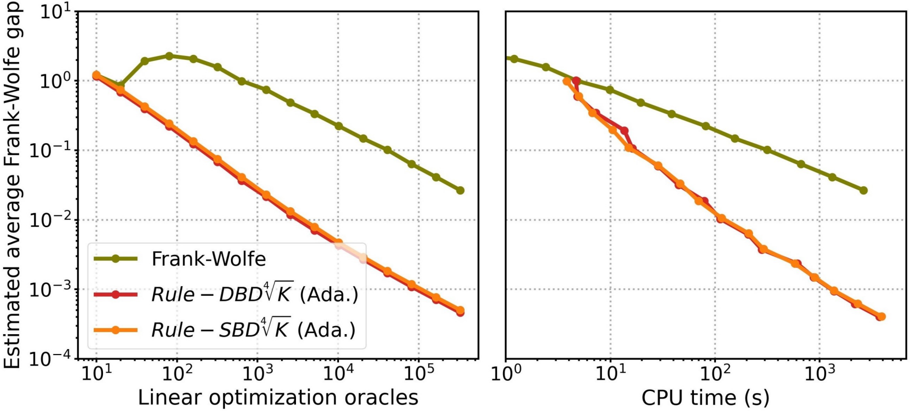


Figure 2: Binary classification with nonconvex losses on dataset a9a ($n = 32561,\ p = 123$)

Binary Classification with Nonconvex Losses

The least runtime (and the corresponding K) for $\sum_{i=1}^{k} \mathcal{G}(x^i)/k \leq \varepsilon$

ε	dataset	n	p	*Rule−SBD*$\sqrt[4]{K}$	*Rule−DBD*$\sqrt[4]{K}$	FW	SPIDER-FW	CASPIDERG	ratio
1e-2	a1a	1605	123	5.58(9.0e4)	**4.34(9.0e4)**	9.58(3.1e6)	16.31(2.1e7)	-	**2.21**
1e-3	a1a	1605	123	78.24(2.0e5)	**56.82(2.0e5)**	892.15(4.2e7)	-	-	**15.70**
1e-4	a1a	1605	123	2027.57(6.3e6)	**1255.90(6.3e6)**	-	-	-	-
1e-2	a2a	2265	123	7.36(7.2e3)	**5.76(1.3e4)**	13.18(3.7e6)	14.32(2.1e7)	-	**2.29**
1e-3	a2a	2265	123	114.72(2.3e5)	**89.56(7.2e5)**	824.00(4.2e7)	-	-	**9.20**
1e-4	a2a	2265	123	2338.66(7.3e6)	**1557.63(7.3e6)**	-	-	-	-
1e-2	a8a	22696	123	72.91(2.1e5)	67.14(1.0e4)	271.15(5.2e6)	**52.29(3.8e7)**	-	0.78
1e-3	a8a	22696	123	898.26(3.4e5)	**884.71(2.2e6)**	-	-	-	-
1e-4	a8a	22696	123	-	-	-	-	-	-
1e-2	a9a	32561	123	89.08(1.6e4)	87.94(2.5e4)	368.65(4.7e6)	**43.29(4.2e7)**	-	0.49
1e-3	a9a	32561	123	1143.13(4.3e5)	**1047.64(6.6e5)**	-	-	-	-
1e-4	a9a	32561	123	-	-	-	-	-	-
5e-2	w1a	2477	300	9.22(3.1e4)	8.97(1.8e4)	**0.42(1.8e4)**	0.94(1.0e6)	89.77(3.1e7)	0.05
1e-2	w1a	2477	300	74.56(2.6e5)	104.45(1.1e6)	**4.98(1.1e5)**	20.03(1.3e7)	-	0.07
2e-3	w1a	2477	300	1133.47(5.2e6)	6053.98(4.2e7)	**101.11(1.6e6)**	-	-	0.09
5e-2	w2a	3470	300	13.07(5.5e4)	12.84(4.3e4)	**0.57(1.6e4)**	0.92(7.9e5)	117.96(3.8e7)	0.04
1e-2	w2a	3470	300	86.79(1.1e5)	113.86(3.3e5)	**7.02(8.2e4)**	21.31(1.3e7)	-	0.08
2e-3	w2a	3470	300	898.55(3.1e6)	2895.85(3.4e7)	**160.91(2.4e6)**	-	-	0.18
5e-2	w7a	24692	300	101.75(3.1e4)	93.43(3.7e4)	8.28(2.3e4)	**2.23(7.2e5)**	547.89(1.9e7)	0.02
1e-2	w7a	24692	300	535.36(2.0e5)	559.76(6.6e4)	80.62(5.7e4)	**28.56(7.3e6)**	-	0.05
2e-3	w7a	24692	300	4130.16(1.7e6)	-	**1770.65(9.2e5)**	2572.64(4.2e7)	-	0.43
5e-2	w8a	49749	300	239.33(6.8e4)	238.28(3.8e4)	19.06(1.0e4)	**3.35(1.2e6)**	1034.63(1.4e7)	0.01
1e-2	w8a	49749	300	1295.48(8.2e4)	1385.60(7.4e4)	161.57(8.2e4)	**39.93(9.4e6)**	-	0.03
2e-3	w8a	49749	300	-	-	**3371.83(9.8e5)**	-	-	-
1e-1	svmguide3	1243	22	0.43(1.8e4)	**0.14(2.9e4)**	2.80(4.2e7)	62.11(2.5e7)	-	**19.81**
1e-2	svmguide3	1243	22	8.44(4.9e4)	**2.52(7.4e4)**	-	-	-	-
1e-3	svmguide3	1243	22	120.56(9.2e5)	**35.47(7.9e5)**	-	-	-	-
1e-4	svmguide3	1243	22	2505.93(1.3e7)	**785.44(2.1e7)**	-	-	-	-
1e-1	phishing	11055	68	1.96(4.5e4)	1.60(6.8e4)	2.61(5.2e5)	**1.16(4.7e6)**	-	0.73
1e-2	phishing	11055	68	20.21(6.6e4)	**15.28(8.4e4)**	170.13(1.5e7)	-	-	**11.14**
1e-3	phishing	11055	68	277.48(5.9e5)	**167.23(1.6e5)**	-	-	-	-
1e-4	phishing	11055	68	-	**4296.52(1.0e7)**	-	-	-	-
1e-1	ijcnn1	49990	22	2.57(3.3e4)	1.31(8.4e3)	11.01(6.6e5)	**1.13(2.6e6)**	428.82(4.2e7)	0.86
1e-2	ijcnn1	49990	22	26.37(3.3e4)	**13.27(2.7e5)**	656.15(2.1e7)	-	-	**49.45**
1e-3	ijcnn1	49990	22	312.15(2.6e5)	**171.37(1.2e6)**	-	-	-	-
1e-4	ijcnn1	49990	22	-	**2789.00(5.2e6)**	-	-	-	-
1e-1	covtype	581012	54	77.61(1.8e6)	53.82(2.2e6)	482.25(2.1e7)	**4.02(1.7e7)**	-	0.07
1e-2	covtype	581012	54	786.03(4.2e6)	**565.53(3.1e6)**	-	-	-	-
1e-3	covtype	581012	54	-	-	-	-	-	-

CASPIDERG is proposed by Shen, Fang, Zhao, Huang & Qian [10]

Comments

- The TUFW methods have the best performance for most problems.
- As $1/\varepsilon$ increases, the dependence of TUFW's complexity on n largely decreases, compared with other methods.

General empirical risk minimization with constraints

$$\text{ERM:} \qquad \text{minimize}_{x \in \mathcal{C}} \quad F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \ , \tag{6}$$

L and $\hat{L}$ are the Lipschitz constants of ∇f_i and $\nabla^2 f_i$.

Theorem (*Rule*$-$*SBD*$\sqrt{k}$ for general ERM)

Suppose that F is convex and Assumption 1 holds, and TUFW with Rule$-$*SBD*$\sqrt{k}$ *is applied to ERM with step-sizes* $\gamma_k := 2/(k+2)$. *Then:*

$$\mathbb{E}[F(x^k) - F(x^\star)] \leq \frac{2LD^2 + 134\hat{L}D^3}{k+1} \tag{7}$$

All other Rules can be extended to general ERM problems.

Summary

- We develop a new family of Frank-Wolfe methods, which we call TUFW that replaces the exact gradient computation by a sum of (second order) Taylor-approximated gradients around certain current and previous iterates.
- All of our rules for TUFW exhibit a *decreasing* number of gradient calls over the course of iterations, while retaining the optimal LMO complexity, together with only a relatively small amount of other flops. The joint dependence on n and ε is $O(n/\sqrt{\varepsilon})$ for convex losses and $O(n/\varepsilon^{3/2})$ for non-convex losses.
- Computational experiments show that TUFW exhibits very significant speed-ups over existing methods on real-world datasets.
- Our TUFW method easily extends to the more general ERM problem, with corresponding versions of our theoretical guarantees.
- We also propose a novel adaptive step-size approach with computational experiments that point to its efficiency in practice.

Thank You

Brief Proof Idea

If we use the same proof techniques of classic FW, we have

Lemma

For ERM_ℓ, suppose that Assumption 1 holds, F is convex, and $\gamma_k = 2/(k+2)$ for all $k \geq 0$. Then

$$F(x^k) - F(x^\star) \leq \frac{2LD_2^2}{n(k+1)} + \frac{2}{k(k+1)} \sum_{t=1}^{k} t\left(\nabla F(x^{t-1}) - g^{t-1}\right)^\top \left(s^{t-1} - x^\star\right) .$$

For $Rule{-}SBD\sqrt{k}$, we can further prove that $\mathbb{E}\text{RED} \leq \mathcal{O}(\hat{L}D_1 D_\infty^2/nt)$.
Similarly, for nonconvex problems,

$$\sum_{k=0}^{K} \frac{\mathcal{G}(x^k)}{K+1} \leq \frac{\varepsilon_0 + LD_2^2/n}{\sqrt{K+1}} + \frac{1}{K+1} \sum_{k=0}^{K} (\nabla F(x^k) - g^k)^\top (s^k - \bar{s}^k).$$

- H. Lu and R. M. Freund, "Generalized stochastic frank–wolfe algorithm with stochastic substitute gradient for structured convex optimization," *Mathematical Programming*, vol. 187, no. 1, pp. 317–349, 2021.
- M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization," in *International Conference on Machine Learning*, pp. 427–435, PMLR, 2013.
- G. Lan and Y. Zhou, "Conditional gradient sliding for convex optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1379–1409, 2016.
- E. Hazan and H. Luo, "Variance-reduced and projection-free stochastic optimization," in *International Conference on Machine Learning*, pp. 1263–1271, PMLR, 2016.
- A. Yurtsever, S. Sra, and V. Cevher, "Conditional gradient methods via stochastic path-integrated differential estimator," in *International Conference on Machine Learning*, pp. 7282–7291, PMLR, 2019.
- A. Mokhtari, H. Hassani, and A. Karbasi, "Stochastic conditional gradient methods: From convex minimization to submodular maximization," *Journal of machine learning research*, 2020.

G. Négiar, G. Dresdner, A. Tsai, L. E. Ghaoui, F. Locatello, R. Freund, and F. Pedregosa, "Stochastic frank-Wolfe for constrained finite-sum minimization," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 7253–7262, PMLR, 13–18 Jul 2020.

S. Lacoste-Julien, "Convergence rate of frank-wolfe for non-convex objectives," *arXiv preprint arXiv:1607.00345*, 2016.

S. J. Reddi, S. Sra, B. Póczos, and A. Smola, "Stochastic frank-wolfe methods for nonconvex optimization," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1244–1251, IEEE, 2016.

Z. Shen, C. Fang, P. Zhao, J. Huang, and H. Qian, "Complexities in projection-free stochastic non-convex minimization," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2868–2876, PMLR, 2019.

M. Zhang, Z. Shen, A. Mokhtari, H. Hassani, and A. Karbasi, "One sample stochastic frank-wolfe," in *International Conference on Artificial Intelligence and Statistics*, pp. 4012–4023, PMLR, 2020.

H. Hassani, A. Karbasi, A. Mokhtari, and Z. Shen, "Stochastic conditional gradient++:(non) convex minimization and continuous submodular maximization," *SIAM Journal on Optimization*, vol. 30, no. 4, pp. 3315–3344, 2020.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach, “Convex and network flow optimization for structured sparsity.,” *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

S. Mei, Y. Bai, and A. Montanari, “The landscape of empirical risk for nonconvex losses,” *The Annals of Statistics*, vol. 46, no. 6A, pp. 2747–2774, 2018.