A Distributed Frank-Wolfe Algorithm for Communication-Efficient Sparse Learning

Alireza Bagheri Garakani[1]
Joint work with Aurélien Bellet[2], Yingyu Liang[3],
Maria-Florina Balcan[4] and Fei Sha[1]

[1]University of Southern California
[2]Télécom ParisTech
[3]Princeton University
[4]Carnegie Mellon University

SIAM International Conference on Data Mining

May 1, 2015

Introduction

Distributed learning

- General setting
  - Data arbitrarily distributed across different nodes
  - Examples: sensor networks, mobile devices, storage purposes
- Research questions
  - Practice: derive scalable algorithms, with small communication and synchronization overhead
  - Theory: study tradeoff between communication complexity and learning/optimization error

Introduction

Problem of interest

Problem of interest

Learn sparse combinations of n distributed "atoms":

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad f(\boldsymbol{\alpha}) = g(\boldsymbol{A}\boldsymbol{\alpha}) \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq \beta \qquad (\boldsymbol{A} \in \mathbb{R}^{d \times n})$$

Note: domain can be unit simplex Δ_n instead of ℓ_1 ball

- Atoms are distributed across a set of N nodes $V = \{v_i\}_{i=1}^N$
- Nodes communicate across a network (connected graph)
- Many applications, including
  - LASSO with distributed features
  - Kernel SVM with distributed training instances
  - Boosting with distributed learners

Introduction

Contributions

- Main ideas
  - Adapt the Frank-Wolfe (FW) algorithm to distributed setting
  - Turn FW sparsity guarantees into communication guarantees
- Summary of results
  - Worst-case optimal communication complexity
  - Balance local computation through approximation
  - Good practical performance on synthetic and real data

Outline

1. Frank-Wolfe in the centralized setting
2. Proposed distributed FW algorithm
3. Approximate variant
4. Communication complexity analysis
5. Experiments

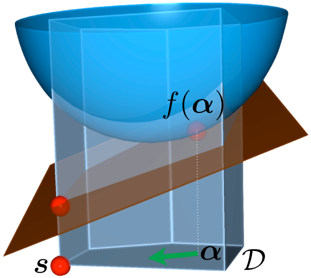
Frank-Wolfe in the centralized setting

Algorithm and convergence

Convex minimization over a compact domain $\mathcal{D}$

$$\min_{\boldsymbol{\alpha}\in\mathcal{D}} \quad f(\boldsymbol{\alpha})$$

- $\mathcal{D}$ convex, f convex and continuously differentiable



```
Let $\boldsymbol{\alpha}^{(0)} \in \mathcal{D}$
for $k = 0, 1, \ldots$ do
    $\boldsymbol{s}^{(k)} = \arg\min_{\boldsymbol{s}\in\mathcal{D}} \left\langle \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}^{(k)}) \right\rangle$
    $\boldsymbol{\alpha}^{(k+1)} = (1-\gamma)\boldsymbol{\alpha}^{(k)} + \gamma \boldsymbol{s}^{(k)}$
end for
```

Convergence [Frank and Wolfe, 1956, Clarkson, 2010, Jaggi, 2013]

After $O(1/\epsilon)$ iterations, FW returns $\boldsymbol{\alpha}$ s.t. $f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*) \leq \epsilon$.

(figure adapted from [Jaggi, 2013])

Frank-Wolfe in the centralized setting

Use-case: sparsity constraint

- Solution to linear minimization step lies at a vertex of $\mathcal{D}$
- When $\mathcal{D}$ is the ℓ_1-norm ball, vertices are signed unit basis vectors $\{\pm \boldsymbol{e}_i\}_{i=1}^n$:
  - FW is greedy: $\boldsymbol{\alpha}^{(0)} = \mathbf{0} \Longrightarrow \|\boldsymbol{\alpha}^{(k)}\|_0 \leq k$
  - FW is efficient: simply find max absolute entry of gradient
- FW finds an ϵ-approximation with $O(1/\epsilon)$ nonzero entries, which is worst-case optimal [Jaggi, 2013]
- Similar derivation for simplex constraint [Clarkson, 2010]

Distributed Frank-Wolfe (dFW)

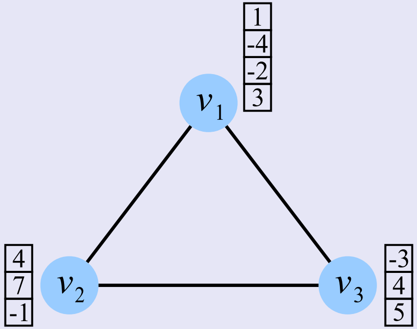
Sketch of the algorithm

Recall our problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad f(\boldsymbol{\alpha}) = g(\boldsymbol{A}\boldsymbol{\alpha}) \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq \beta \qquad (\boldsymbol{A}\in\mathbb{R}^{d\times n})$$

Algorithm steps per iteration

1. Each node computes its local gradient



Distributed Frank-Wolfe (dFW)

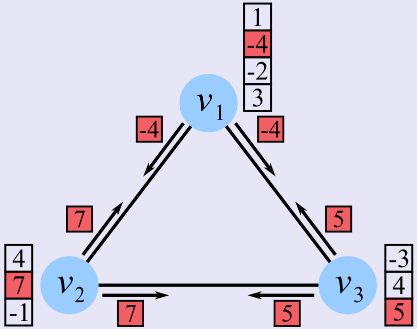
Sketch of the algorithm

Recall our problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad f(\boldsymbol{\alpha}) = g(\boldsymbol{A}\boldsymbol{\alpha}) \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq \beta \qquad (\boldsymbol{A} \in \mathbb{R}^{d\times n})$$

Algorithm steps per iteration

2. Each node broadcast its largest absolute value



Distributed Frank-Wolfe (dFW)

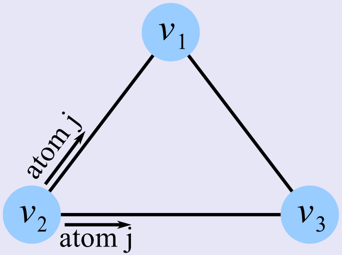
Sketch of the algorithm

Recall our problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad f(\boldsymbol{\alpha}) = g(\boldsymbol{A}\boldsymbol{\alpha}) \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq \beta \qquad (\boldsymbol{A} \in \mathbb{R}^{d\times n})$$

Algorithm steps per iteration

3. Node with global max broadcasts corresponding atom $\boldsymbol{a}_j \in \mathbb{R}^d$



Distributed Frank-Wolfe (dFW)

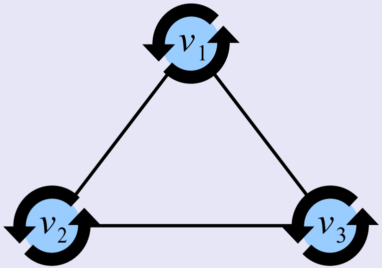
Sketch of the algorithm

Recall our problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad f(\boldsymbol{\alpha}) = g(\boldsymbol{A}\boldsymbol{\alpha}) \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \leq \beta \qquad (\boldsymbol{A} \in \mathbb{R}^{d\times n})$$

Algorithm steps per iteration

4. All nodes update current solution α, and loop



Distributed Frank-Wolfe (dFW)

Convergence

- Tradeoff between communication and optimization error
- Let B be the cost of broadcasting a real number

Theorem 1 (Convergence of exact dFW)

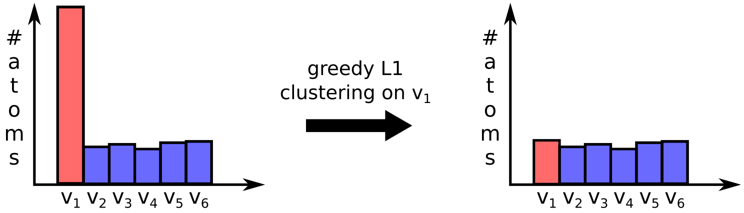
After $O(1/\epsilon)$ rounds and $O\left((Bd + NB)/\epsilon\right)$ total communication, each node holds an ϵ-approximate solution.

- No dependence on total number of combining elements

Distributed Frank-Wolfe (dFW)

Approximate variant

- Exact dFW is scalable but requires synchronization
  - Unbalanced local computation $\rightarrow$ significant wait time
- Strategy to balance local costs:
  - Node v_i clusters its n_i atoms into m_i groups
  - We use the greedy m-center algorithm [Gonzalez, 1985]
  - Run dFW on resulting centers
- Use-case examples:
  - Balance number of atoms across nodes
  - Set m_i proportional to computational resources of v_i



Distributed Frank-Wolfe (dFW)

Approximate variant

- Define
  - $r^{opt}(\mathcal{A}, m)$ to be the optimal ℓ_1-radius of partitioning atoms in $\mathcal{A}$ into m clusters, and $r^{opt}(\boldsymbol{m}) := \max_i r^{opt}(\mathcal{A}_i, m_i)$
  - $G := \max_{\boldsymbol{\alpha}} \|\nabla g(\boldsymbol{A}\boldsymbol{\alpha})\|_\infty$

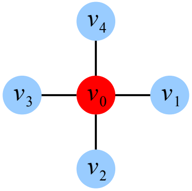
Theorem 2 (Convergence of approximate dFW)

After $O(1/\epsilon)$ iterations, the algorithm returns a solution with optimality gap at most $\epsilon + O(Gr^{opt}(\boldsymbol{m}^0))$. Furthermore, if $r^{opt}(\boldsymbol{m}^{(k)}) = O(1/Gk)$, then the gap is at most ϵ.

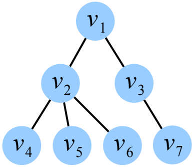
- Additive error depends on cluster tightness
- Can gradually add more centers to make error vanish

Communication complexity analysis

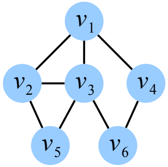
Cost of dFW under various network topologies



Star graph



Rooted tree



General connected graph

- Star graph and rooted tree: $O(Nd/\epsilon)$ communication (use network structure to reduce cost)
- General connected graph: $O(M(N+d)/\epsilon)$, where M is the number of edges (use a message-passing strategy)

Communication complexity analysis

Matching lower bound

Theorem 3 (Communication lower bound)

Under mild assumptions, the worst-case communication cost of any deterministic algorithm is $\Omega(d/\epsilon)$.

- Shows that dFW is worst-case optimal in ϵ and d

Experiments

- Objective value achieved for given communication budget
  - Compared to distributed ADMM method [Boyd et al., 2011], dFW is advantageous when data and/or solution is sparse
  - Compared to Local FW method [Lodi et al., 2010], dFW consistently outperforms due to better selection strategy
- Runtime of dFW in large-scale distributed setting
  - Benefits of approximate variant
  - Asynchronous updates

Experiments

Large-scale distributed setting

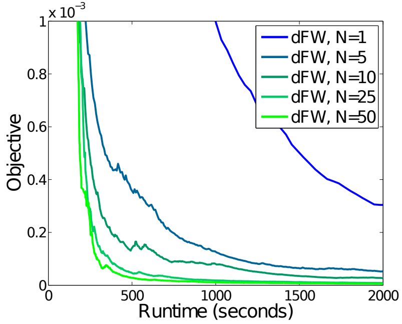
- Infrastructure
  - Fully connected with $N \in \{1, 5, 10, 25, 50\}$ nodes
  - A node is a single 2.4GHz CPU core of a separate host
  - Communication over 56.6-gigabit network
- Task
  - SVM with Gaussian RBF kernel
  - Speech data with 8.7M training examples, 41 classes
  - Implementation of dFW in `C++` with openMPI[1]

[1] `http://www.open-mpi.org`

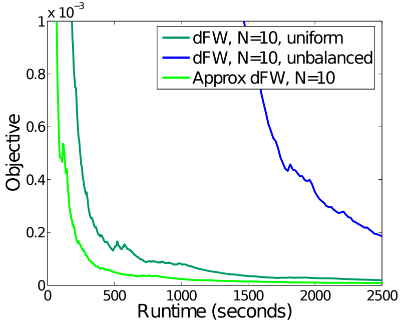
Experiments

Large-scale distributed setting

- When distribution of atoms is roughly balanced, dFW achieves near-linear speedup
- When distribution is unbalanced (e.g., 1 node has 50% of the data), great benefits from approximate variant



(a) dFW on uniform distribution

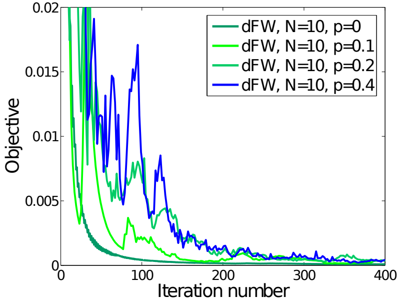


(b) Approximate dFW to balance costs

Experiments

Large-scale distributed setting

- Another way to reduce synchronization costs is to perform asynchronous updates
- To simulate this, we randomly drop communication messages with probability p
- dFW is fairly robust, even with 40% random drops



dFW under communication errors and asynchrony

Summary and perspectives

- The proposed distributed algorithm
    - is applicable to a family of sparse learning problems
    - has theoretical guarantees and good practical performance
    - appears robust to asynchronous updates and communication errors
- See paper for details, proofs and additional experiments
- Future directions
    - Propose an asynchronous version of dFW
    - A theoretical study in this challenging setting

References I

[Boyd et al., 2011] Boyd, S. P., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011).
Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.
Foundations and Trends in Machine Learning, 3(1):1–122.

[Clarkson, 2010] Clarkson, K. L. (2010).
Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm.
ACM Transactions on Algorithms, 6(4):1–30.

[Frank and Wolfe, 1956] Frank, M. and Wolfe, P. (1956).
An algorithm for quadratic programming.
Naval Research Logistics Quarterly, 3(1-2):95–110.

[Gonzalez, 1985] Gonzalez, T. F. (1985).
Clustering to minimize the maximum intercluster distance.
Theoretical Computer Science, 38:293–306.

[Jaggi, 2013] Jaggi, M. (2013).
Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization.
In *ICML*.

[Lodi et al., 2010] Lodi, S., Ñanculef, R., and Sartori, C. (2010).
Single-Pass Distributed Learning of Multi-class SVMs Using Core-Sets.
In *SDM*, pages 257–268.