Distributed Frank-Wolfe Algorithm

A Unified Framework for Communication-Efficient Sparse Learning

Aurélien Bellet[1]

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

[1]University of Southern California

[2]Georgia Institute of Technology

[3]Carnegie Mellon University

ICML 2014 Workshop on New Learning Frameworks and Models for Big Data

June 25, 2014

Introduction

Distributed learning

- General setting
  - Data arbitrarily distributed across different sites (*nodes*)
  - Examples: large-scale data, sensor networks, mobile devices
  - Communication between nodes can be a serious bottleneck
- Research questions
  - Theory: study tradeoff between communication complexity and learning/optimization error
  - Practice: derive scalable algorithms, with small communication and synchronization overhead

Problem of interest

Learn sparse combinations of n distributed "atoms":

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

- Atoms are distributed across a set of N nodes $V = \{v_i\}_{i=1}^N$
- Nodes communicate across a network (connected graph)
- Note: domain can be unit simplex Δ_n instead of ℓ_1 ball

$$\Delta_n = \{\alpha \in \mathbb{R}^n : \boldsymbol{\alpha} \geq 0, \sum_i \alpha_i = 1\}$$

Introduction

Applications

- Many applications
  - LASSO with distributed features
  - Kernel SVM with distributed training points
  - Boosting with distributed learners
  - ...

Example: Kernel SVM

- Training set $\{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^n$
- Kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}') \rangle$
- Dual problem of L2-SVM:

$$\min_{\boldsymbol{\alpha} \in \Delta_n} \quad \boldsymbol{\alpha}^{\mathrm{T}} \tilde{\boldsymbol{K}} \boldsymbol{\alpha}$$

- $\tilde{\boldsymbol{K}} = [\tilde{k}(\boldsymbol{z}_i, \boldsymbol{z}_j)]_{i,j=1}^n$ with $\tilde{k}(\boldsymbol{z}_i, \boldsymbol{z}_j) = y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$
- Atoms are $\tilde{\varphi}(\boldsymbol{z}_i) = [y_i \varphi(\boldsymbol{x}_i), y_i, \frac{1}{\sqrt{C}} \boldsymbol{e}_i]$

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. Communication complexity analysis
4. Experiments

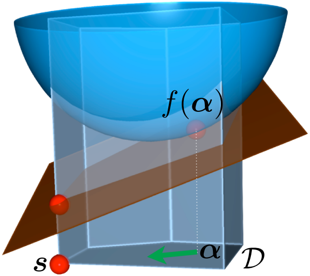
Frank-Wolfe in the centralized setting

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**

$\quad \boldsymbol{s}^{(k)} = \arg\min_{\boldsymbol{s} \in \mathcal{D}} \left\langle \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}^{(k)}) \right\rangle$

$\quad \boldsymbol{\alpha}^{(k+1)} = (1-\gamma)\boldsymbol{\alpha}^{(k)} + \gamma \boldsymbol{s}^{(k)}$

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

- A solution to linear subproblem 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)

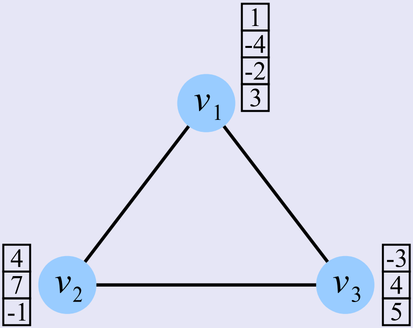
Sketch of the algorithm

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

1. Each node computes its local gradient



Distributed Frank-Wolfe (dFW)

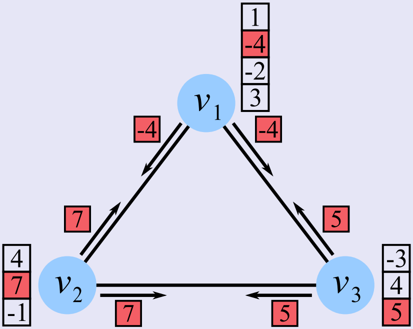
Sketch of the algorithm

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

2. Each node broadcast its largest absolute value



Distributed Frank-Wolfe (dFW)

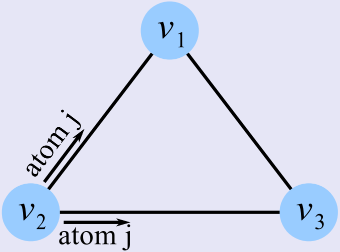
Sketch of the algorithm

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

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



Distributed Frank-Wolfe (dFW)

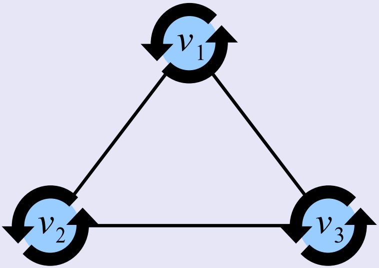
Sketch of the algorithm

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

4. All nodes perform a FW update and start over



Distributed Frank-Wolfe (dFW)

Convergence

- Let B be the cost of broadcasting a real number

Theorem 1 (Convergence of exact dFW)

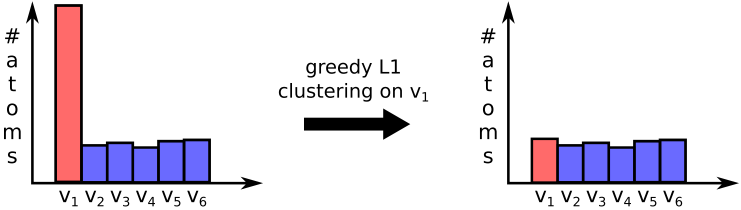
After $O(1/\epsilon)$ rounds and $O\left((Bd + NB)/\epsilon\right)$ total communication, each node holds an ϵ-approximate solution.

- Tradeoff between communication and optimization error
- 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 power 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 $r^{opt}(\boldsymbol{m}) := \max_i r^{opt}(\mathcal{A}_i, m_i)$
  - $G := \max_{\boldsymbol{\alpha}} \|\nabla g(\boldsymbol{A\alpha})\|_\infty$

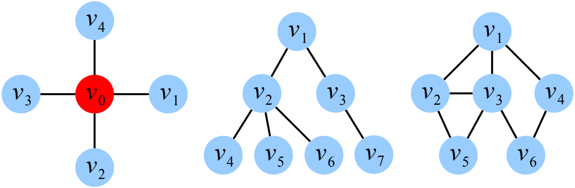
Theorem 2 (Convergence of approximate dFW)

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
- Proof outline:
  1. Identify a problem instance for which any ϵ-approximate solution has $O(1/\epsilon)$ atoms
  2. Distribute data across 2 nodes s.t. these atoms are almost evenly split across nodes
  3. Show that for any fixed dataset on one node, there are T different instances on the other node s.t. in any 2 such instances, the sets of selected atoms are different
  4. Any node then needs $O(\log T)$ bits to figure out the selected atoms, and we show that $\log T = \Omega(d/\epsilon)$

Experiments

- Objective value achieved for given communication budget
  - Comparison to baselines
  - Comparison to distributed ADMM
- Runtime of dFW in realistic distributed setting
  - Exact dFW
  - Benefits of approximate variant
  - Asynchronous updates

Experiments

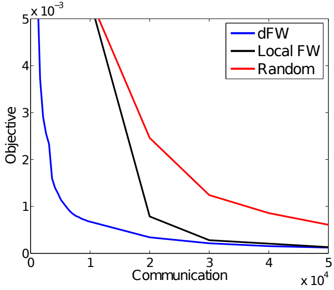
Comparison to baselines

- dFW can be seen as a method to select "good" atoms
- We investigate 2 baselines:
  - Random: each node picks a fixed set of atoms at random
  - Local FW [Lodi et al., 2010]: each node runs FW locally to select a fixed set of atoms
- Selected atoms are sent to a coordinator node which solves the problem using only these atoms

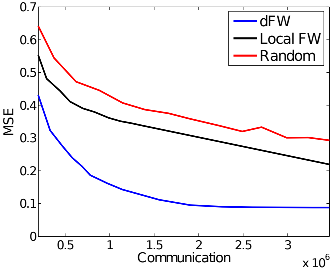
Experiments

Comparison to baselines

- Experimental setup
  - SVM with RBF kernel on Adult dataset ($n = 32K$, $d = 123$)
  - LASSO on Dorothea dataset ($n = 100K$, $d = 1.15K$)
  - Atoms distributed across 100 nodes uniformly at random
- dFW outperforms both baselines



(a) Kernel SVM results



(b) LASSO results

Experiments

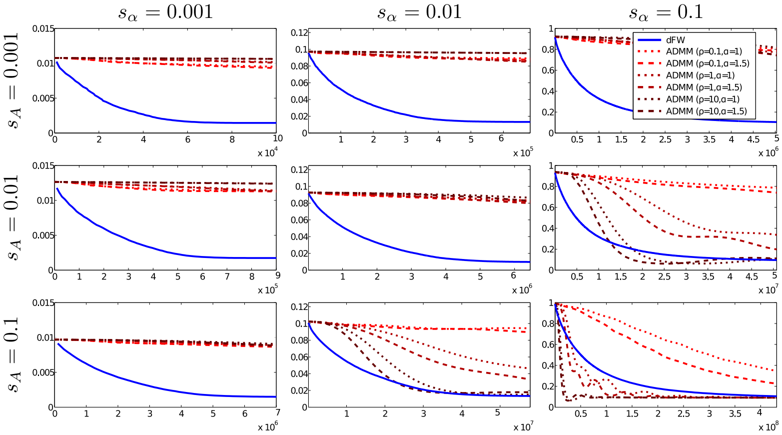
Comparison to distributed ADMM

- ADMM [Boyd et al., 2011] is popular to tackle many distributed optimization problems
  - Like dFW, can deal with LASSO with distributed features
  - Parameter vector $\boldsymbol{\alpha}$ partitioned as $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_N]$
  - Communicates partial/global predictions: $\boldsymbol{A}_i\boldsymbol{\alpha}_i$ and $\sum_{i=1}^{N} \boldsymbol{A}_i\boldsymbol{\alpha}_i$
- Experimental setup
  - Synthetic data ($n = 100K$, $d = 10K$) with varying sparsity
  - Atoms distributed across 100 nodes uniformly at random

Experiments

Comparison to distributed ADMM

- dFW advantageous for sparse data and/or solution, while ADMM is preferable in the dense setting
- Note: no parameter to tune for dFW



LASSO results (MSE vs communication)

Experiments

Realistic distributed environment

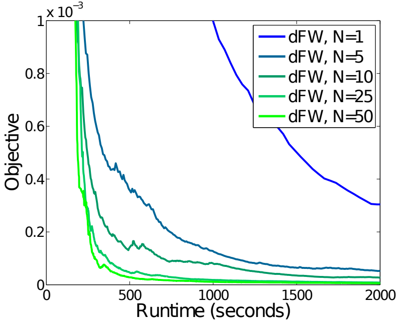
- Network specs
  - Fully connected with $N \in \{1, 5, 10, 25, 50\}$ nodes
  - A node is a single 2.4GHz CPU core of a separate host
  - Communication over 56.6-gigabit infrastructure
- The task
  - SVM with Gaussian RBF kernel
  - Speech data with 8.7M training examples, 41 classes
  - Implementation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

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

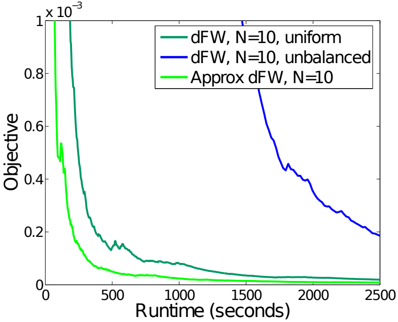
Experiments

Realistic distributed environment

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



(a) Exact dFW on uniform distribution

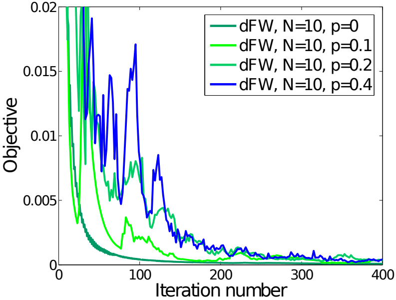


(b) Approximate dFW to balance costs

Experiments

Real-world distributed environment

- 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 asynchrony and communication errors
- See arXiv paper for details, proofs and additional experiments
- Future directions
  - Propose an asynchronous version of dFW
  - A theoretical study in this challenging setting
    - Could potentially build on recent work in distributed optimization that assumes or enforces a bound on the age of the updates [Ho et al., 2013, Liu et al., 2014]

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.

[Ho et al., 2013] Ho, Q., Cipar, J., Cui, H., Lee, S., Kim, J. K., Gibbons, P. B., Gibson, G. A., Ganger, G. R., and Xing, E. P. (2013).
More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server.
In *NIPS*, pages 1223–1231.

References II

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

[Liu et al., 2014] Liu, J., Wright, S. J., Ré, C., Sridhar, S., and Bittorf, V. (2014).
An Asynchronous Parallel Stochastic Coordinate Descent Algorithm.
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.