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

Aurélien Bellet\textsuperscript{1}
Joint work with Yingyu Liang\textsuperscript{2}, Alireza Bagheri Garakani\textsuperscript{1}, Maria-Florina Balcan\textsuperscript{3} and Fei Sha\textsuperscript{1}

\textsuperscript{1}University of Southern California
\textsuperscript{2}Georgia Institute of Technology
\textsuperscript{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
Introduction

Problem of interest

Learn sparse combinations of $n$ distributed “atoms”:

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

- 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 $\Delta_n$ instead of $\ell_1$ ball

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

Applications

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

Example: Kernel SVM

- Training set \( \{ z_i = (x_i, y_i) \}_{i=1}^n \)
- Kernel \( k(x, x') = \langle \varphi(x), \varphi(x') \rangle \)
- Dual problem of L2-SVM:
  \[
  \min_{\alpha \in \Delta_n} \alpha^T \tilde{K} \alpha
  \]
  \[
  \tilde{K} = [\tilde{k}(z_i, z_j)]_{i,j=1}^n \text{ with } \tilde{k}(z_i, z_j) = y_i y_j k(x_i, x_j) + y_i y_j + \frac{\delta_{ij}}{C}
  \]
  \[
  \varphi(z_i) = [y_i \varphi(x_i), y_i, \frac{1}{\sqrt{C}} e_i]
  \]
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 $\mathcal{D}$

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

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

Let $\alpha^{(0)} \in \mathcal{D}$

for $k = 0, 1, \ldots$ do

\[ s^{(k)} = \arg \min_{s \in \mathcal{D}} \langle s, \nabla f(\alpha^{(k)}) \rangle \]
\[ \alpha^{(k+1)} = (1 - \gamma)\alpha^{(k)} + \gamma s^{(k)} \]

end for

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

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

(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 $\ell_1$-norm ball, vertices are signed unit basis vectors $\{\pm e_i\}_{i=1}^n$:
  - FW is greedy: $\alpha^{(0)} = 0 \implies \|\alpha^{(k)}\|_0 \leq k$
  - FW is efficient: simply find max absolute entry of gradient
- FW finds an $\epsilon$-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

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

Algorithm steps

1. Each node computes its local gradient
Distributed Frank-Wolfe (dFW)

Sketch of the algorithm

Recall our problem

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

Algorithm steps

2. Each node broadcast its largest absolute value
Distributed Frank-Wolfe (dFW)

Sketch of the algorithm

Recall our problem

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

Algorithm steps

3. Node with global best broadcasts corresponding atom $a_j \in \mathbb{R}^d$
Recall our problem

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

Algorithm steps

4. All nodes perform a FW update and start over
Let $B$ be the cost of broadcasting a real number.

**Theorem 1 (Convergence of exact dFW)**

After $O(1/\epsilon)$ rounds and $O((Bd + NB)/\epsilon)$ total communication, each node holds an $\epsilon$-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 → 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
  - \( r^{\text{opt}}(A, m) \) to be the optimal \( \ell_1 \)-radius of partitioning atoms in \( A \) into \( m \) clusters, and
  - \( r^{\text{opt}}(m) := \max_i r^{\text{opt}}(A_i, m_i) \)
  - \( G := \max_\alpha \| \nabla g(A\alpha) \|_\infty \)

Theorem 2 (Convergence of approximate dFW)

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

- 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 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 $\epsilon$ and $d$

- Proof outline:
  1. Identify a problem instance for which any $\epsilon$-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 $\alpha$ partitioned as $\alpha = [\alpha_1, \ldots, \alpha_N]$
  - Communicates partial/global predictions: $A_i\alpha_i$ and $\sum_{i=1}^{N} A_i\alpha_i$

- 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

\[^{1}\text{http://www.open-mpi.org}\]
Experiments

Realistic distributed environment

- When distribution of atoms is \textit{roughly balanced}, exact dFW achieves \textit{near-linear speedup}

- When distribution is \textit{unbalanced} (e.g., 1 node has 50\% of the data), \textit{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

![Graph showing dFW under communication errors and asynchrony](image)
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


References II

