Trade-Offs in Distributed Learning and Optimization

Ohad Shamir

Weizmann Institute of Science

Includes joint works with Yossi Arjevani, Nathan Srebro and Tong Zhang

IHES Workshop
March 2016

Distributed Learning and Optimization

Why?

- Big data
  - Too large to fit into a single machine
  - Distributed learning as a **constraint**
- Computational Speed-ups
  - Ideally, k machines $= 1/k$ training time
  - Distributed learning as an **opportunity**

Challenge 1: Communication

Very slow compared to local processing[1]:

L1 Cache Reference	$\sim \mathbf{0.5}$ ns
Main memory Reference	$\sim \mathbf{100}$ ns
Round-trip within same datacenter	$\sim \mathbf{500,000}$ ns
Packet California$\Rightarrow$Netherlands & back	$\sim \mathbf{150,000,000}$ ns

[1] `http://norvig.com/21-days.html#answers`

Challenges

Challenge 2: Parallelization

How to parallelize algorithms of the following form:

Challenges

Challenge 3: Accuracy

Given the same data, output quality should resemble that of a non-distributed algorithm

Setting

Convex distributed empirical risk minimization:

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w})$$

$$F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

$f_{i,j}(\cdot)$: Loss on j-th example in i-th machine.

- Strongly-convex/Smooth problems: Each F_i is strongly convex/smooth

Machines can broadcast simultaneously in communication rounds

- $\tilde{\mathcal{O}}(d)$ bits per machine/round

Main Question

How to optimally trade-off

1. Accuracy: $F(\mathbf{w}) - \inf_{\mathbf{w} \in \mathcal{W}} F(\mathbf{w}) \leq \epsilon$
2. Communication
3. Runtime

Notes: In this talk, accuracy in terms of optimization error, not risk

3 Data Partition Scenarios

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

3 Data Partition Scenarios

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

Arbitrary partition

No relationship between F_i's

3 Data Partition Scenarios

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

Arbitrary partition

No relationship between F_i's

Random partition

- mn individual losses $f_{i,j}$ assigned to machines at random
- Strong concentration of measure effects (e.g. $F_i(\cdot)$'s have similar values/gradients/Hessians)

3 Data Partition Scenarios

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

Arbitrary partition

No relationship between F_i's

Random partition

- mn individual losses $f_{i,j}$ assigned to machines at random
- Strong concentration of measure effects (e.g. $F_i(\cdot)$'s have similar values/gradients/Hessians)

δ-related setting

- Generalization of previous scenarios
- Informally: Values/gradients/Hessians at any $\mathbf{w}$ are $\leq \delta$-distant across machines

Talk Outline

Algorithms + worst-case upper and lower bounds* for

1. Arbitrary partition
2. δ-related
3. Random partition
   - Relies on new results for without-replacement sampling in stochastic gradient methods

* In terms of # communication rounds

Arbitrary Partition

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

Baseline: Reduce to non-distributed first-order optimization

Example (Gradient Descent)

- Initialize $\mathbf{w}_0$; For $t = 1, 2, \ldots$
  - Machine i computes $\nabla F_i(\mathbf{w}_t)$
  - Communication round: $\nabla F(\mathbf{w}_t) = \frac{1}{m}\sum_{i=1}^{m} \nabla F_i(\mathbf{w}_t)$
  - Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla F(\mathbf{w}_t)$

Arbitrary Partition

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

Baseline: Reduce to non-distributed first-order optimization

Example (Gradient Descent)

- Initialize $\mathbf{w}_0$; For $t = 1, 2, \ldots$
  - Machine i computes $\nabla F_i(\mathbf{w}_t)$
  - Communication round: $\nabla F(\mathbf{w}_t) = \frac{1}{m}\sum_{i=1}^{m} \nabla F_i(\mathbf{w}_t)$
  - Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\nabla F(\mathbf{w}_t)$

- Can also accelerate; use proximal smoothing etc.
- $\#$ communication rounds $=$ iteration complexity
  - λ-Strongly convex $+$ 1-smooth: $\mathcal{O}\left(\sqrt{1/\lambda}\log\left(\frac{1}{\epsilon}\right)\right)$
  - λ-Strongly convex: $\mathcal{O}(\sqrt{1/\lambda\epsilon})$; Convex: $\mathcal{O}(1/\epsilon)$ (with smoothing)
- Almost full parallelization; but many comm. rounds

Arbitrary Partition

Can we improve on this simple baseline?

Arbitrary Partition

Can we improve on this simple baseline?
For very large family of algorithms, baseline is (worse-case) optimal!

Arbitrary Partition

Can we improve on this simple baseline?
For very large family of algorithms, baseline is (worse-case) optimal!

Assumed Algorithmic Template

- For each machine j, let $W_j = \{\mathbf{0}\}$
- Between communication rounds: Each machine j sequentially computes and adds to W_j an arbitrary number of vectors $\mathbf{w}$, s.t. $\gamma\mathbf{w} + \nu\nabla F_j(\mathbf{w})$ (for some $\gamma, \nu \geq 0$, $\gamma + \nu > 0$) is in

$$\text{span}\Big\{\mathbf{w}'\ ,\ \nabla F_j(\mathbf{w}')\ ,\ (\nabla^2 F_j(\mathbf{w}') + D)\mathbf{w}''\ ,\ (\nabla^2 F_j(\mathbf{w}') + D)^{-1}\mathbf{w}''\ \Big|\ \mathbf{w}', \mathbf{w}'' \in W_j\ \ ,\ \ D \text{ diagonal}\Big\}$$

- During communication round: Broadcast some vectors in W_j

Proof Sketch (λ-strongly convex; 1-smooth; 2 machines)

$$
\begin{aligned}
F_1(\mathbf{w}) &= \mathbf{w}^\top \left(\frac{1-\lambda}{4} A_1 + \frac{\lambda}{2} I \right) \mathbf{w} - \frac{1-\lambda}{2} \mathbf{e}_1^\top \mathbf{w} \\
F_2(\mathbf{w}) &= \mathbf{w}^\top \left(\frac{1-\lambda}{4} A_2 + \frac{\lambda}{2} I \right) \mathbf{w}
\end{aligned}
$$

Proof Sketch (λ-strongly convex; 1-smooth; 2 machines)

$$
\begin{aligned}
F_1(\mathbf{w}) &= \mathbf{w}^\top \left(\frac{1-\lambda}{4} A_1 + \frac{\lambda}{2} I \right) \mathbf{w} - \frac{1-\lambda}{2} \mathbf{e}_1^\top \mathbf{w} \\
F_2(\mathbf{w}) &= \mathbf{w}^\top \left(\frac{1-\lambda}{4} A_2 + \frac{\lambda}{2} I \right) \mathbf{w}
\end{aligned}
$$

$$
A_1 = \begin{bmatrix}
+1 & 0 & 0 & 0 & 0 & 0 & \dots \\
0 & +1 & -1 & 0 & 0 & 0 & \dots \\
0 & -1 & +1 & 0 & 0 & 0 & \dots \\
0 & 0 & 0 & +1 & -1 & 0 & \dots \\
0 & 0 & 0 & -1 & +1 & 0 & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\qquad
A_2 = \begin{bmatrix}
+1 & -1 & 0 & 0 & 0 & 0 & \dots \\
-1 & +1 & 0 & 0 & 0 & 0 & \dots \\
0 & 0 & +1 & -1 & 0 & 0 & \dots \\
0 & 0 & -1 & +1 & 0 & 0 & \dots \\
0 & 0 & 0 & 0 & +1 & -1 & \dots \\
0 & 0 & 0 & 0 & -1 & +1 & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}
$$

- $\exp(-\Omega(T/\sqrt{1/\lambda}))$ error after T iterations
- $\frac{1}{2}F_1(\cdot) + \frac{1}{2}F_2(\cdot)$ is essentially "hard function" for *non-distributed* optimization [Nemirovski and Yudin 1983]

Proof Sketch (λ-strongly convex); 2 machines

$\Omega(1/(\lambda T^2))$ error after T iterations
(matched by AGD with proximal smoothing)

$$
\begin{aligned}
F_1(\mathbf{w}) = \frac{1}{\sqrt{2}}|b - w_1| \\
&+ \frac{1}{\sqrt{2(T+2)}}\left(|w_2 - w_3| + |w_4 - w_5| + \cdots + |w_T - w_{T+1}|\right) \\
&+ \frac{\lambda}{2}\|\mathbf{w}\|^2
\end{aligned}
$$

$$
\begin{aligned}
F_2(\mathbf{w}) = \frac{1}{\sqrt{2(T+2)}}\left(|w_1 - w_2| + |w_3 - w_4| + \cdots + |w_{T+1} - w_{T+2}|\right) \\
&+ \frac{\lambda}{2}\|\mathbf{w}\|^2
\end{aligned}
$$

δ-related Setting

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

- Assuming F_i are smooth, values/gradients/Hessians at any $\mathbf{w}\in\mathcal{W}$ are δ-close

δ-related Setting

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

- Assuming F_i are smooth, values/gradients/Hessians at any $\mathbf{w} \in \mathcal{W}$ are δ-close
- Can show similar lower bounds, but weakened by δ factors
  - e.g. $\sqrt{\frac{\delta}{\lambda}} \log\left(\frac{1}{\epsilon}\right)$ for λ-strongly convex and smooth

Can we build algorithms which utilize δ-relatedness?

More data – less communication??

DANE Algorithm

Distributed **A**pproximate **NE**wton-type method

- Parameters: Learning rate $\eta > 0$; regularizer $\mu > 0$
- Initialize $\mathbf{w}_1 = 0$
- For $t = 1, 2, \ldots$
  - Compute $\nabla F(\mathbf{w}_t) = \frac{1}{m} \sum_i \nabla F_i(\mathbf{w}_t)$
  - For each machine i, solve local optimization problem:

$$\mathbf{w}_{t+1}^i = \arg\min_{\mathbf{w}} \quad F_i(\mathbf{w}) - (\nabla F_i(\mathbf{w}_t) - \eta \nabla F(\mathbf{w}_t))^\top \mathbf{w} + \frac{\mu}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2$$

  - Compute average $\mathbf{w}_{t+1} = \frac{1}{m} \sum_i \mathbf{w}_{t+1}^i$

Structure similar to ADMM

Crucial Property

Equivalent to **approximate Newton step**

Newton step

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \left(\nabla^2 F(\mathbf{w}_t)\right)^{-1} \nabla F(\mathbf{w}_t)$$

Extremely fast convergence, but expensive

Crucial Property

Equivalent to **approximate Newton step**

Newton step

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \left(\frac{1}{m} \sum_i \nabla^2 F_i(\mathbf{w}_t) \right)^{-1} \nabla F(\mathbf{w}_t)$$

Extremely fast convergence, but expensive

Crucial Property

Equivalent to **approximate Newton step**

Newton step

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \left(\frac{1}{m}\sum_i \nabla^2 F_i(\mathbf{w}_t)\right)^{-1} \nabla F(\mathbf{w}_t)$$

Extremely fast convergence, but expensive

DANE (Quadratic Functions)

Equivalent to

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \left(\frac{1}{m}\sum_i \left(\nabla^2 F_i(\mathbf{w}_t)\right)^{-1} + \mu I\right) \nabla F(\mathbf{w}_t)$$

even though Hessians $\nabla^2 F_i$ never computed explicitly!

Theoretical Guarantees: Quadratic Functions

Theorem

$$\left\|\mathbf{w}_{t+1} - \mathbf{w}_{opt}\right\| \leq \left\|I - \eta \tilde{H}^{-1} H\right\|^t \left\|\mathbf{w}_1 - \mathbf{w}_{opt}\right\|$$

where

$$H = \frac{1}{m}\sum_i \nabla^2 F_i \quad , \quad \tilde{H}^{-1} = \frac{1}{m}\sum_i (\nabla^2 F_i + \mu I)^{-1}$$

- In a δ-related setting, $\left\|\nabla^2 F_i - H\right\| \leq \delta$ for all i
- Setting η, μ appropriately,

$$\left\|I - \eta \tilde{H}^{-1} H\right\| \;\leq\; \max\left\{\frac{1}{2}, 1 - \frac{1}{16(\delta/\lambda)^2}\right\}$$

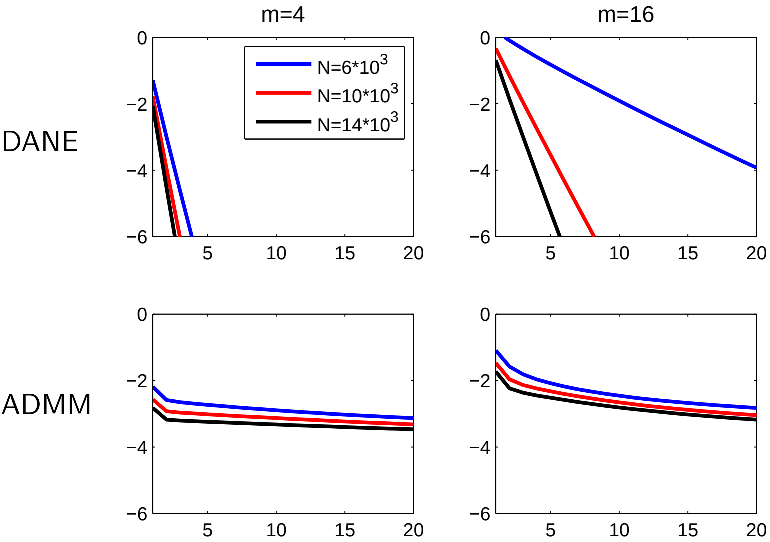
Conclusion

Need $\mathcal{O}((\delta/\lambda)^2)\log(1/\epsilon)$ communication rounds

Some Experiments

Regularized least squares in $\mathbb{R}^{500}$

$\log_{10}$(optimization error) vs. # iterations



Extensions

- Can provide some guarantees for non-quadratic losses as well
- Using a more sophisticated algorithm, can improve guarantees to $\mathcal{O}(\sqrt{\delta/\lambda}\log(1/\epsilon))$ rounds, for quadratics / self-concordant losses [Zhang and Xiao 2015]

Disadvantage: Each iteration requires solving a local optimization problem $\Rightarrow$ Not necessarily cheap in terms of runtime

Random Partition

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

- Special case of δ-related setting, $\delta = \mathcal{O}(1/\sqrt{n})$ in general

Random Partition

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

- Special case of δ-related setting, $\delta = \mathcal{O}(1/\sqrt{n})$ in general
- Previous algorithms: $\mathcal{O}(\log(1/\epsilon))$ communication rounds as long as $\lambda \geq \Omega(1/\sqrt{n})$

Random Partition

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

- Special case of δ-related setting, $\delta = \mathcal{O}(1/\sqrt{n})$ in general
- Previous algorithms: $\mathcal{O}(\log(1/\epsilon))$ communication rounds as long as $\lambda \geq \Omega(1/\sqrt{n})$
- Next:
  - Much simpler approach for randomly partitioned data
  - Can get $\mathcal{O}(\log(1/\epsilon))$ communication rounds as long as $\lambda \geq \tilde{\Omega}(1/n)$
- Detour : without-replacement sampling for stochastic gradient methods...

Stochastic Gradient Methods

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{w})$$

Stochastic Gradient Descent / Subgradient Method

- $\mathbf{w}_1 = \mathbf{0}$
- For $t = 1, 2, \ldots$
  - Sample i_t
  - Let $\mathbf{g}_t \in \partial f_{i_t}(\mathbf{w}_t)$
  - $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}\left(\mathbf{w}_t - \eta_t \mathbf{g}_t\right)$

Stochastic Gradient Methods

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{w})$$

Stochastic Gradient Descent / Subgradient Method

- $\mathbf{w}_1 = \mathbf{0}$
- For $t = 1, 2, \ldots$
  - Sample i_t
  - Let $\mathbf{g}_t \in \partial f_{i_t}(\mathbf{w}_t)$
  - $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}\left(\mathbf{w}_t - \eta_t \mathbf{g}_t\right)$

- Standard analysis: each i_t sampled uniformly from $\{1, \ldots, N\}$
- Works because $\mathbb{E}[\mathbf{g}_t] \in \partial F(\mathbf{w}_t)$
- But: one often samples i_t without replacement
- Equivalently, go over a random shuffle of the data

Why Without Replacement Sampling?

- Often works better (all data equally processed)
- Requires sequential rather than random data access

However, much harder to analyze

Why Without Replacement Sampling?

- Often works better (all data equally processed)
- Requires sequential rather than random data access

However, much harder to analyze

- Incremental gradient bounds (e.g. Nedić and Bertsekas 2001): Hold for any order, but can be exponentially slower

Why Without Replacement Sampling?

- Often works better (all data equally processed)
- Requires sequential rather than random data access

However, much harder to analyze

- Incremental gradient bounds (e.g. Nedić and Bertsekas 2001): Hold for any order, but can be exponentially slower
- (Gürbüzbalaban, Ozdaglar, Parillo 2015): For strongly convex and smooth problems, k passes over N data points:
  - Without-replacement: $\mathcal{O}((N/k)^2)$ error
  - With-replacement: $\mathcal{O}(1/Nk)$

New Results

Without-replacement sampling is not worse than with-replacement sampling (in worst-case sense) under various scenarios, e.g.

- $\mathcal{O}(1/\sqrt{T})$ for convex linear prediction
- $\mathcal{O}(1/\lambda T)$ for λ-strongly-convex and smooth linear prediction
- $\exp\left(-\Omega\left(\frac{T}{N+1/\lambda}\right)\right)$ for least squares on N data points, using no-replacement SVRG algorithm

Analysis uses ideas from adversarial online learning and transductive learning theory

Derivation: Convex Linear Prediction

$$\min_{\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^d} F(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} f_i(\mathbf{w})$$

Algorithm:

- Sequentially processes $f_{\sigma(1)}, f_{\sigma(2)}, \ldots, f_{\sigma(T)}$ where σ random permutation on $\{1, \ldots, N\}$
- Produces iterates $\mathbf{w}_1, \ldots, \mathbf{w}_T$

Goal: Prove

$$\mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} F(\mathbf{w}_t) - F(\mathbf{w}^*)\right] \;\leq\; \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

where $\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathcal{W}} F(\mathbf{w})$

Derivation: Convex Linear Prediction

Assumption: Algorithm has bounded regret in adversarial online setting: For *any* sequence of convex Lipschitz $f_1, f_2, \ldots$ and $\mathbf{w} \in \mathcal{W}$

$$\frac{1}{T}\sum_{i=1}^{T} f_i(\mathbf{w}_t) - \frac{1}{T}\sum_{i=1}^{T} f_i(\mathbf{w}) \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

Satisfied for e.g. stochastic gradient descent

Derivation: Convex Linear Prediction (1/2)

$$\mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T} F(\mathbf{w}_t) - F(\mathbf{w}^*)\right]$$

Derivation: Convex Linear Prediction (1/2)

$$\mathbb{E}_\sigma \left[\frac{1}{T} \sum_{t=1}^{T} F(\mathbf{w}_t) - F(\mathbf{w}^*) \right]$$

$$= \mathbb{E}_\sigma \left[\frac{1}{T} \sum_{t=1}^{T} F(\mathbf{w}_t) - f_{\sigma(t)}(\mathbf{w}^*) \right]$$

Derivation: Convex Linear Prediction (1/2)

$$\mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T} F(\mathbf{w}_t) - F(\mathbf{w}^*)\right]$$

$$= \mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T} F(\mathbf{w}_t) - f_{\sigma(t)}(\mathbf{w}^*)\right]$$

$$= \mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T}\left(f_{\sigma(t)}(\mathbf{w}_t) - f_{\sigma(t)}(\mathbf{w}^*)\right)\right] + \mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T}\left(F(\mathbf{w}_t) - f_{\sigma(t)}(\mathbf{w}_t)\right)\right]$$

Derivation: Convex Linear Prediction (1/2)

$$
\begin{aligned}
&\mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T} F(\mathbf{w}_t) - F(\mathbf{w}^*)\right] \\
&= \mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T} F(\mathbf{w}_t) - f_{\sigma(t)}(\mathbf{w}^*)\right] \\
&= \mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T} \left(f_{\sigma(t)}(\mathbf{w}_t) - f_{\sigma(t)}(\mathbf{w}^*)\right)\right] + \mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T} \left(F(\mathbf{w}_t) - f_{\sigma(t)}(\mathbf{w}_t)\right)\right] \\
&= \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \mathbb{E}_\sigma\left[\frac{1}{T}\sum_{t=1}^{T}\left(\frac{\sum_{i=1}^{N} f_{\sigma(i)}(\mathbf{w}_t)}{N} - \frac{\sum_{i=t}^{N} f_{\sigma(i)}(\mathbf{w}_t)}{N-t+1}\right)\right]
\end{aligned}
$$

Derivation: Convex Linear Prediction (2/2)

Following algebraic manipulations,

$$= \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=1}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[F_{1:t-1}(\mathbf{w}_t) - F_{t:N}(\mathbf{w}_t)\right]$$

where $F_{a:b}$ is average over $f_{\sigma(a)},\ldots,f_{\sigma(b)}$

Derivation: Convex Linear Prediction (2/2)

Following algebraic manipulations,

$$= \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=1}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[F_{1:t-1}(\mathbf{w}_t) - F_{t:N}(\mathbf{w}_t)\right]$$

where $F_{a:b}$ is average over $f_{\sigma(a)}, \ldots, f_{\sigma(b)}$

$$\leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=2}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[\sup_{\mathbf{w}\in\mathcal{W}}\left(F_{1:t-1}(\mathbf{w}) - F_{t:N}(\mathbf{w})\right)\right]$$

Following algebraic manipulations,

$$= \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=1}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[F_{1:t-1}(\mathbf{w}_t) - F_{t:N}(\mathbf{w}_t)\right]$$

where $F_{a:b}$ is average over $f_{\sigma(a)}, \ldots, f_{\sigma(b)}$

$$\leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=2}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[\sup_{\mathbf{w}\in\mathcal{W}}\left(F_{1:t-1}(\mathbf{w}) - F_{t:N}(\mathbf{w})\right)\right]$$

$$\leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=2}^{T}\frac{t-1}{N}\cdot\mathcal{R}_{t-1:N-t+1}(\mathcal{W}) + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right),$$

where $\mathcal{R}_{t-1:N-t+1}(\mathcal{W})$ is transductive Rademacher complexity of $\mathcal{W}$ w.r.t. $f_1, f_2, \ldots, f_N$

Derivation: Convex Linear Prediction (2/2)

Following algebraic manipulations,

$$= \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=1}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[F_{1:t-1}(\mathbf{w}_t) - F_{t:N}(\mathbf{w}_t)\right]$$

where $F_{a:b}$ is average over $f_{\sigma(a)}, \ldots, f_{\sigma(b)}$

$$\leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=2}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[\sup_{\mathbf{w}\in\mathcal{W}}\left(F_{1:t-1}(\mathbf{w}) - F_{t:N}(\mathbf{w})\right)\right]$$

$$\leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=2}^{T}\frac{t-1}{N}\cdot\mathcal{R}_{t-1:N-t+1}(\mathcal{W}) + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right),$$

where $\mathcal{R}_{t-1:N-t+1}(\mathcal{W})$ is transductive Rademacher complexity of $\mathcal{W}$ w.r.t. $f_1, f_2, \ldots, f_N$

Specializing to linear predictors and convex Lipschitz losses, get familiar $\mathcal{O}(1/\sqrt{T})$ rate

Derivation: Convex Linear Prediction (2/2)

Following algebraic manipulations,

$$= \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=1}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[F_{1:t-1}(\mathbf{w}_t) - F_{t:N}(\mathbf{w}_t)\right]$$

where $F_{a:b}$ is average over $f_{\sigma(a)}, \ldots, f_{\sigma(b)}$

$$\leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=2}^{T}\frac{t-1}{N}\cdot\mathbb{E}\left[\sup_{\mathbf{w}\in\mathcal{W}}\left(F_{1:t-1}(\mathbf{w}) - F_{t:N}(\mathbf{w})\right)\right]$$

$$\leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \frac{1}{T}\sum_{t=2}^{T}\frac{t-1}{N}\cdot\mathcal{R}_{t-1:N-t+1}(\mathcal{W}) + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right),$$

where $\mathcal{R}_{t-1:N-t+1}(\mathcal{W})$ is transductive Rademacher complexity of $\mathcal{W}$ w.r.t. $f_1, f_2, \ldots, f_N$

Specializing to linear predictors and convex Lipschitz losses, get familiar $\mathcal{O}(1/\sqrt{T})$ rate

With more effort, get $\mathcal{O}(1/\lambda T)$ with λ-strong convexity

SVRG

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{w})$$

- In recent years, fast stochastic algorithms to solve finite-sum problems
  - SAG [Le Roux et al. 2012], SDCA (as analyzed in [Shalev-Shwartz et al. 2013, 2016]), SVRG [Johnson and Zhang 2013], SAGA [Defazio et al. 2014], Finito [Defazio et al. 2014], S2GD [Konecný and Richtárik 2013]...
- Cheap stochastic iterations + linear convergence rate (assuming strong convexity and smoothness)
- All existing analyses based on with-replacement sampling
- We consider without-replacement SVRG

With-Replacement SVRG

Initialize $\tilde{\mathbf{w}}_1 = \mathbf{0}$
for $s = 1, 2, \ldots, S$ **do**
  Compute $\nabla F(\tilde{\mathbf{w}}_s) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\tilde{\mathbf{w}}_s)$
  $\mathbf{w}_1 := \tilde{\mathbf{w}}_s$
  for $t = 1, 2, \ldots, T$ **do**
    Draw i_t uniformly at random from $\{1, \ldots, N\}$
    $\mathbf{w}_{t+1} := \mathbf{w}_t - \eta \left(\nabla f_{i_t}(\mathbf{w}_t) - \nabla f_{i_t}(\tilde{\mathbf{w}}_s) + \nabla F(\tilde{\mathbf{w}}_s) \right)$
  end for
  Pick $\tilde{\mathbf{w}}_{s+1}$ at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$
end for

With-Replacement SVRG

Initialize $\tilde{\mathbf{w}}_1 = \mathbf{0}$
for $s = 1, 2, \ldots, S$ **do**
  Compute $\nabla F(\tilde{\mathbf{w}}_s) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\tilde{\mathbf{w}}_s)$
  $\mathbf{w}_1 := \tilde{\mathbf{w}}_s$
  for $t = 1, 2, \ldots, T$ **do**
    Draw i_t uniformly at random from $\{1, \ldots, N\}$
    $\mathbf{w}_{t+1} := \mathbf{w}_t - \eta\left(\nabla f_{i_t}(\mathbf{w}_t) - \nabla f_{i_t}(\tilde{\mathbf{w}}_s) + \nabla F(\tilde{\mathbf{w}}_s)\right)$
  end for
  Pick $\tilde{\mathbf{w}}_{s+1}$ at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$
end for

Analysis for 1-smooth, λ-strongly convex functions:
$S = \mathcal{O}(\log(1/\epsilon))$, $T \geq 1/\lambda$

With-Replacement SVRG

Initialize $\tilde{\mathbf{w}}_1 = \mathbf{0}$
for $s = 1, 2, \ldots, S$ **do**
  Compute $\nabla F(\tilde{\mathbf{w}}_s) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\tilde{\mathbf{w}}_s)$
  $\mathbf{w}_1 := \tilde{\mathbf{w}}_s$
  for $t = 1, 2, \ldots, T$ **do**
    Draw i_t uniformly at random from $\{1, \ldots, N\}$
    $\mathbf{w}_{t+1} := \mathbf{w}_t - \eta\left(\nabla f_{i_t}(\mathbf{w}_t) - \nabla f_{i_t}(\tilde{\mathbf{w}}_s) + \nabla F(\tilde{\mathbf{w}}_s)\right)$
  end for
  Pick $\tilde{\mathbf{w}}_{s+1}$ at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$
end for

Analysis for 1-smooth, λ-strongly convex functions:
$S = \mathcal{O}(\log(1/\epsilon))$, $T \geq 1/\lambda$

- Observation [Lee, Lin, Ma 2015]: Can simulate it in distributed setting with randomly partitioned data, as long as $T \leq \mathcal{O}(1/\sqrt{N})$

With-Replacement SVRG

Initialize $\tilde{\mathbf{w}}_1 = \mathbf{0}$
for $s = 1, 2, \ldots, S$ **do**
  Compute $\nabla F(\tilde{\mathbf{w}}_s) \ = \ \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\tilde{\mathbf{w}}_s)$
  $\mathbf{w}_1 \ := \ \tilde{\mathbf{w}}_s$
  for $t = 1, 2, \ldots, T$ **do**
    Draw i_t uniformly at random from $\{1, \ldots, N\}$
    $\mathbf{w}_{t+1} \ := \ \mathbf{w}_t - \eta \left(\nabla f_{i_t}(\mathbf{w}_t) - \nabla f_{i_t}(\tilde{\mathbf{w}}_s) + \nabla F(\tilde{\mathbf{w}}_s) \right)$
  end for
  Pick $\tilde{\mathbf{w}}_{s+1}$ at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$
end for

Analysis for 1-smooth, λ-strongly convex functions:
$S = \mathcal{O}(\log(1/\epsilon))$, $T \geq 1/\lambda$

- Observation [Lee, Lin, Ma 2015]: Can simulate it in distributed setting with randomly partitioned data, as long as $T \leq \mathcal{O}(1/\sqrt{N})$
- By analysis, only applicable when $\lambda \geq \Omega(1/\sqrt{N})$

Without-Replacement SVRG

Given: Permutation σ on $\{1,\ldots,N\}$
Initialize $\tilde{\mathbf{w}}_1 = \mathbf{0}$
for $s = 1,2,\ldots,S$ **do**
  Compute $\nabla F(\tilde{\mathbf{w}}_s) \;=\; \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\tilde{\mathbf{w}}_s)$
  $\mathbf{w}_1 \;:=\; \tilde{\mathbf{w}}_s$
  for $t = 1,2,\ldots,T$ **do**
    $\mathbf{w}_{t+1} \;:=\; \mathbf{w}_t - \eta\left(\nabla f_{\sigma(t)}(\mathbf{w}_t) - \nabla f_{\sigma(t)}(\tilde{\mathbf{w}}_s) + \nabla F(\tilde{\mathbf{w}}_s)\right)$
  end for
  Pick $\tilde{\mathbf{w}}_{s+1}$ at random from $\mathbf{w}_1,\ldots,\mathbf{w}_T$
end for

Observation: Can simulate it in distributed setting with randomly partitioned data, all the way up to $T = \Omega(N)$

Bound for Regularized Least Squares

$f_i(\mathbf{w}) = \frac{1}{2}(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2 + \frac{\hat{\lambda}}{2}\|\mathbf{w}\|^2$; $F(\cdot)$ λ-strongly convex

Theorem

If perform $S = \mathcal{O}(\log(1/\epsilon))$ epochs with $T = \Theta(1/\lambda)$ without-replacement stochastic iterations,

$$\mathbb{E}[F(\tilde{\mathbf{w}}_{S+1}) - F(\mathbf{w}^*)] \leq \epsilon$$

Bound for Regularized Least Squares

$f_i(\mathbf{w}) = \frac{1}{2}(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2 + \frac{\hat{\lambda}}{2} \|\mathbf{w}\|^2$; $F(\cdot)$ λ-strongly convex

Theorem

If perform $S = \mathcal{O}(\log(1/\epsilon))$ *epochs with* $T = \Theta(1/\lambda)$ *without-replacement stochastic iterations,*

$$\mathbb{E}[F(\tilde{\mathbf{w}}_{S+1}) - F(\mathbf{w}^*)] \leq \epsilon$$

Implications:

Bound for Regularized Least Squares

$f_i(\mathbf{w}) = \frac{1}{2}(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2 + \frac{\hat{\lambda}}{2} \|\mathbf{w}\|^2$; $F(\cdot)$ λ-strongly convex

Theorem

If perform $S = \mathcal{O}(\log(1/\epsilon))$ *epochs with* $T = \Theta(1/\lambda)$ *without-replacement stochastic iterations,*

$$\mathbb{E}[F(\tilde{\mathbf{w}}_{S+1}) - F(\mathbf{w}^*)] \leq \epsilon$$

Implications:

- Non-distributed optimization: $\lambda \geq \tilde{\Omega}(1/N) \Rightarrow \epsilon$-optimal solution without data reshuffling

Bound for Regularized Least Squares

$f_i(\mathbf{w}) = \frac{1}{2}(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2 + \frac{\hat{\lambda}}{2}\|\mathbf{w}\|^2$; $F(\cdot)$ λ-strongly convex

Theorem

If perform $S = \mathcal{O}(\log(1/\epsilon))$ epochs with $T = \Theta(1/\lambda)$ without-replacement stochastic iterations,

$$\mathbb{E}[F(\tilde{\mathbf{w}}_{S+1}) - F(\mathbf{w}^*)] \leq \epsilon$$

Implications:

- Non-distributed optimization: $\lambda \geq \tilde{\Omega}(1/N) \Rightarrow \epsilon$-optimal solution without data reshuffling
- Distributed optimization: $\lambda \geq \tilde{\Omega}(1/n) \Rightarrow \epsilon$-optimal solution with randomly-partitioned data
  - $\mathcal{O}(\log(1/\epsilon))$ communication rounds
  - Runtime dominated by fully-parallelizable gradient computation

Summary and Open Questions

$$\min_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} F_i(\mathbf{w}) \qquad F_i(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} f_{i,j}(\mathbf{w})$$

- Arbitrary partition ✓
  - Baseline is also (worst-case) optimal...
- δ-related
  - For quadratics ✓
  - More generally X✓
    - Self-concordant losses
    - Algorithms communication-efficient, not necessarily runtime-efficient
- Random partition
  - Least squares ✓
  - More generally X✓
    - λ-strongly-convex and smooth, but only when $\lambda \geq \Omega(1/\sqrt{\text{data size}})$