Minimizing Finite Sums

Mohamed Osama Ahmed
13/10/2015

Big-N Problems

- Recall the regularized empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i) \quad + \quad \lambda r(x)$$

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

Big-N Problems

- Recall the regularized empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i) \quad + \quad \lambda r(x)$$

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Stochastic methods:
  - $O(1/t)$ convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.

Big-N Problems

- Recall the regularized empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i) \quad + \quad \lambda r(x)$$

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Stochastic methods:
  - $O(1/t)$ convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.
- Deterministic methods:
  - $O(\rho^t)$ convergence but requires N gradients per iteration.
  - The faster rate is possible because N is finite.

Big-N Problems

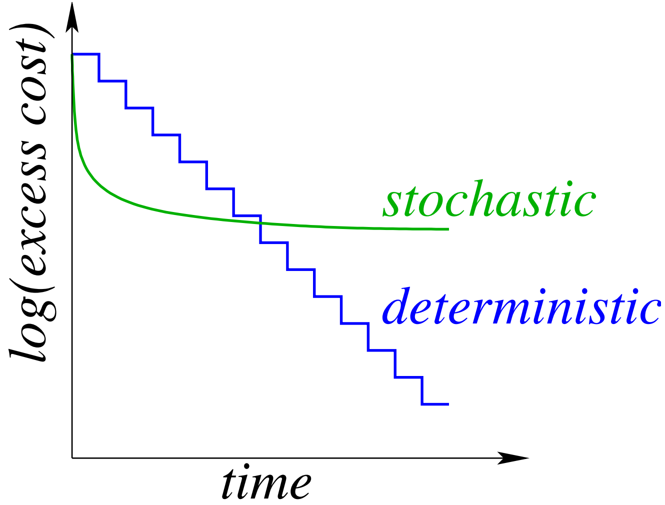
- Recall the regularized empirical risk minimization problem:

$$\min_{x \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} L(x, a_i, b_i) \quad + \quad \lambda r(x)$$

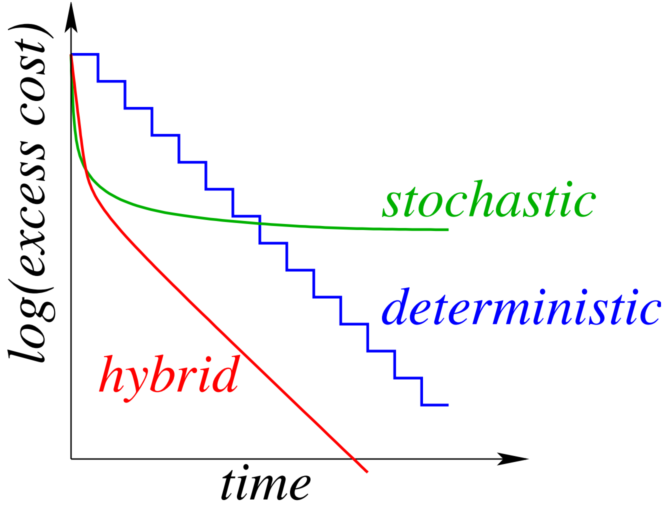
$$\text{data fitting term} \quad + \quad \text{regularizer}$$

- Stochastic methods:
  - $O(1/t)$ convergence but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.
- Deterministic methods:
  - $O(\rho^t)$ convergence but requires N gradients per iteration.
  - The faster rate is possible because N is finite.
- For minimizing finite sums, can we design a better method?

Motivation for Hybrid Methods



Motivation for Hybrid Methods



Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.

Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.
- The FG method uses all N gradients,

$$\nabla f(x^t) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^t).$$

- The SG method approximates it with 1 sample,

$$\nabla f_{i_t}(x^t) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^t).$$

Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.
- The FG method uses all N gradients,

$$\nabla f(x^t) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^t).$$

- The SG method approximates it with 1 sample,

$$\nabla f_{i_t}(x^t) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^t).$$

- A common variant is to use larger sample $\mathcal{B}^t$,

$$\frac{1}{|\mathcal{B}^t|}\sum_{i\in\mathcal{B}^t} \nabla f_i(x^t) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^t).$$

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^t$ uses iterations

$$x^{t+1} = x^t - \frac{\alpha^t}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} f_i(x^t).$$

- For a fixed sample size $|\mathcal{B}^t|$, the rate is sublinear.

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^t$ uses iterations

$$x^{t+1} = x^t - \frac{\alpha^t}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} f_i(x^t).$$

- For a fixed sample size $|\mathcal{B}^t|$, the rate is sublinear.
- **Gradient error decreases as sample size $|\mathcal{B}^t|$ increases.**

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^t$ uses iterations

$$x^{t+1} = x^t - \frac{\alpha^t}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} f_i(x^t).$$

- For a fixed sample size $|\mathcal{B}^t|$, the rate is sublinear.
- **Gradient error decreases as sample size $|\mathcal{B}^t|$ increases.**
- Common to gradually increase the sample size $|\mathcal{B}^t|$.
  [Bertsekas & Tsitsiklis, 1996]
- We can choose $|\mathcal{B}^t|$ to achieve a linear convergence rate:
  - Early iterations are cheap like SG iterations.
  - Later iterations can use a Newton-like method.

Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires O(N) iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**

Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires O(N) iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES!

Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires O(N) iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select i_t from $\{1, 2, \ldots, N\}$ and compute $f'_{i_t}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} \nabla f_i(x^t)$$

Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires O(N) iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select i_t from $\{1, 2, \ldots, N\}$ and compute $f'_{i_t}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} \nabla f_i(x^t)$$

Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires O(N) iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select i_t from $\{1, 2, \ldots, N\}$ and compute $f'_{i_t}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = \nabla f_i(x^t)$ from the last t where i was selected. [Le Roux et al., 2012]

Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires O(N) iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select i_t from $\{1, 2, \ldots, N\}$ and compute $f'_{i_t}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = \nabla f_i(x^t)$ from the last t where i was selected. [Le Roux et al., 2012]
  - Stochastic variant of increment average gradient (IAG). [Blatt et al., 2007]

Stochastic Average Gradient

- Growing $|\mathcal{B}^t|$ eventually requires O(N) iteration cost.
- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select i_t from $\{1, 2, \ldots, N\}$ and compute $f'_{i_t}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = \nabla f_i(x^t)$ from the last t where i was selected. [Le Roux et al., 2012]
  - Stochastic variant of increment average gradient (IAG). [Blatt et al., 2007]
  - Assumes gradients of non-selected examples don't change.
  - Assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.

Convergence Rate of SAG

- If each f_i' is $L-$continuous and f is strongly-convex, with $\alpha_t = 1/16L$ SAG has

$$\mathbb{E}[f(x^t) - f(x^*)] \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t C,$$

  where

$$C = [f(x^0) - f(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$

Convergence Rate of SAG

- If each f_i' is $L-$continuous and f is strongly-convex, with $\alpha_t = 1/16L$ SAG has

$$\mathbb{E}[f(x^t) - f(x^*)] \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t C,$$

  where

$$C = [f(x^0) - f(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$

- Linear convergence rate but only 1 gradient per iteration.
  - For well-conditioned problems, constant reduction per pass:

$$\left(1 - \frac{1}{8N}\right)^N \leq \exp\left(-\frac{1}{8}\right) = 0.8825.$$

  - For ill-conditioned problems, almost same as deterministic method (but N times faster).

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

Rate of Convergence Comparison

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

- Number of f_i' evaluations to reach ϵ:

Rate of Convergence Comparison

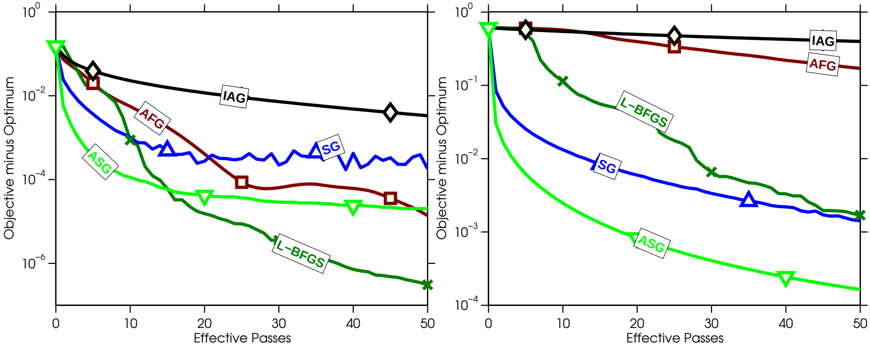
- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.

- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).

- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$.
  - Gradient: $O(N\frac{L}{\mu}\log(1/\epsilon))$.
  - Accelerated: $O(N\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$.
  - SAG: $O(\max\{N, \frac{L}{\mu}\}\log(1/\epsilon))$.

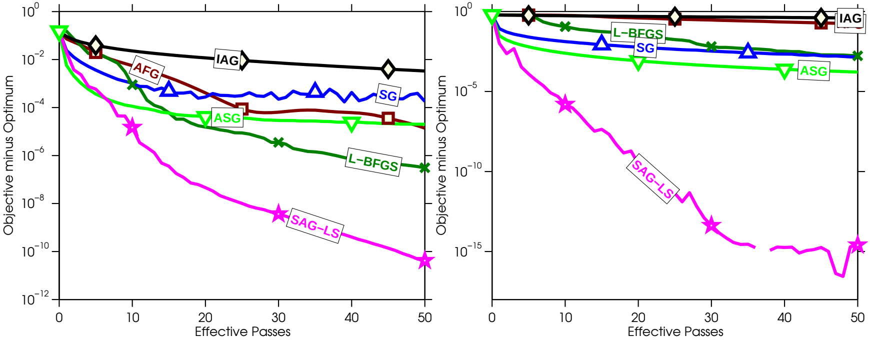
Comparing Deterministic and Stochatic Methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



SAG Compared to FG and SG Methods

- quantum ($n = 50000$, $p = 78$) and rcv1 ($n = 697641$, $p = 47236$)



Other Linearly-Convergent Stochastic Methods

- Subsequent stochastic algorithms with linear rates:
  - Stochastic dual coordinate ascent [Shalev-Schwartz & Zhang, 2013]
  - Incremental surrogate optimization [Mairal, 2013].
  - Stochastic variance-reduced gradient (SVRG) [Johnson & Zhang, 2013, Konecny & Richtarik, 2013, Mahdavi et al., 2013, Zhang et al., 2013]
  - SAGA [Defazio et al., 2014]

Other Linearly-Convergent Stochastic Methods

- Subsequent stochastic algorithms with linear rates:
  - Stochastic dual coordinate ascent [Shalev-Schwartz & Zhang, 2013]
  - Incremental surrogate optimization [Mairal, 2013].
  - Stochastic variance-reduced gradient (SVRG) [Johnson & Zhang, 2013, Konecny & Richtarik, 2013, Mahdavi et al., 2013, Zhang et al., 2013]
  - SAGA [Defazio et al., 2014]
- SVRG has a much lower memory requirement.
- There are also non-smooth extensions.

SAG Implementation Issues

- Basic SAG algorithm:
  - while(1)
  - Sample i from $\{1, 2, \dots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N} d$.

SAG Implementation Issues

- Basic SAG algorithm:
  - while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N} d$.
- Practical variants of the basic algorithm allow:
  - Regularization.
  - Sparse gradients.
  - Automatic step-size selection.
  - Termination criterion.
  - Acceleration [Lin et al., 2015].

SAG Implementation Issues

- Basic SAG algorithm:
  - while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N} d$.
- Practical variants of the basic algorithm allow:
  - Regularization.
  - Sparse gradients.
  - Automatic step-size selection.
  - Termination criterion.
  - Acceleration [Lin et al., 2015].
  - Adaptive non-uniform sampling [Schmidt et al., 2013].

Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - For classic SG: Maybe?
    - Noncommutative arithmetic-geometric mean inequality conjecture.

[Recht & Ré, 2012]

Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - For classic SG: Maybe?
    - Noncommutative arithmetic-geometric mean inequality conjecture.

[Recht & Ré, 2012]

  - For SAG: NO.
  - Performance is intermediate between IAG and SAG.

Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - For classic SG: Maybe?
    - Noncommutative arithmetic-geometric mean inequality conjecture.

      [Recht & Ré, 2012]
  - For SAG: NO.
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?
  - For classic SG methods, can only improve constants.

Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - For classic SG: Maybe?
    - Noncommutative arithmetic-geometric mean inequality conjecture.

[Recht & Ré, 2012]

  - For SAG: NO.
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?
  - For classic SG methods, can only improve constants.
  - For SAG, bias sampling towards Lipschitz constants L_i,

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\|.$$

improves rate to depend on L_{mean} instead of $L_{\max}$.

(with bigger step size)

Reshuffling and Non-Uniform Sampling

- Does re-shuffling and doing full passes work better?
  - For classic SG: Maybe?
    - Noncommutative arithmetic-geometric mean inequality conjecture.

[Recht & Ré, 2012]

  - For SAG: NO.
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?
  - For classic SG methods, can only improve constants.
  - For SAG, bias sampling towards Lipschitz constants L_i,

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\|.$$

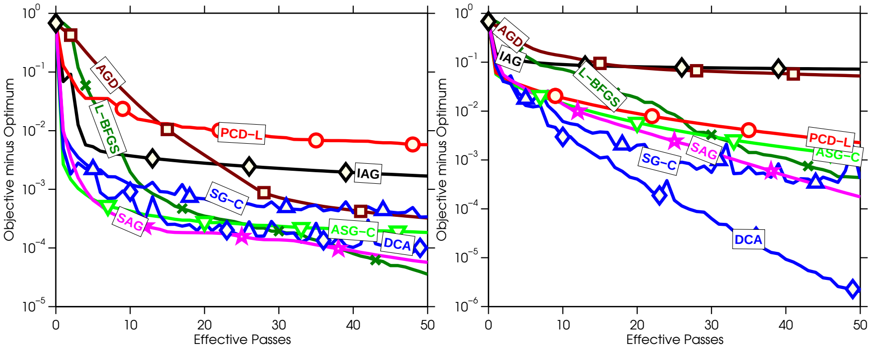
    improves rate to depend on L_{mean} instead of $L_{\max}$.

(with bigger step size)

  - Adaptively estimate L_i as you go.
  - Slowly learns to ignore well-classified examples.

SAG with Adaptive Non-Uniform Sampling

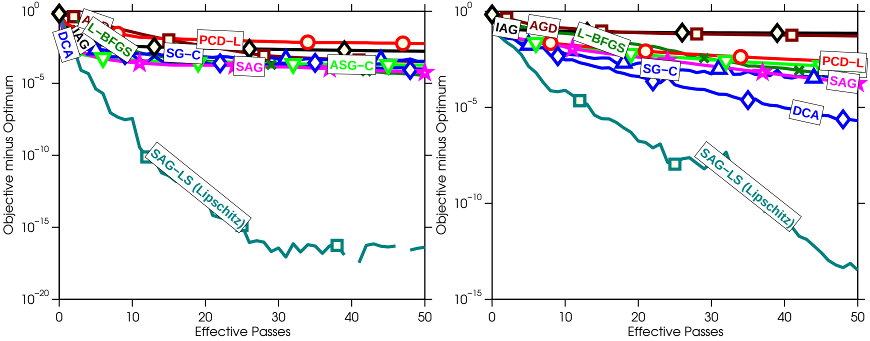
- protein ($n = 145751$, $p = 74$) and sido ($n = 12678$, $p = 4932$)



- Datasets where SAG had the worst relative performance.

SAG with Non-Uniform Sampling

- protein ($n = 145751$, $p = 74$) and sido ($n = 12678$, $p = 4932$)



- Adaptive non-uniform sampling helps a lot.

SAG with Mini-Batches

- Reasons to use mini-batches with SAG:
  1. Parallelize gradient calculation.
  2. Decrease memory (only store gradient of the mini-batch).

SAG with Mini-Batches

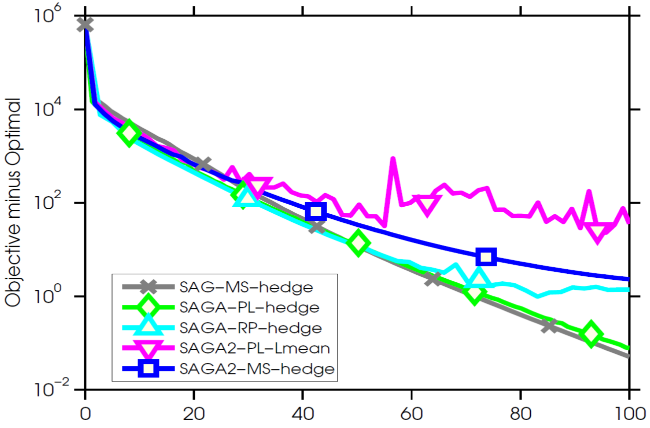
- Reasons to use mini-batches with SAG:
  1. Parallelize gradient calculation.
  2. Decrease memory (only store gradient of the mini-batch).
  3. Increase convergence rate.
     (classic SG methods: only changes constant)

SAG with Mini-Batches

- Reasons to use mini-batches with SAG:
  1. Parallelize gradient calculation.
  2. Decrease memory (only store gradient of the mini-batch).
  3. Increase convergence rate.
     (classic SG methods: only changes constant)
- Convergence rate depends on L for mini-batches:
  - $L(\mathcal{B}) \leq L(i)$, possibly by up to $|\mathcal{B}|$.
  - Allows bigger step-size, $\alpha = 1/L(\mathcal{B})$.
  - Place examples in batches to make $L(\mathcal{B})$ small.

Comparing SAG and SAGA

- named-entity recognition tasks (CoNLL-2000)



Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.

Minimizing Finite Sums: Dealing with the Memory

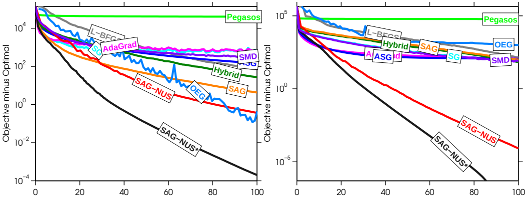
- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
  - Use mini-batches.

Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
  - Use mini-batches.
  - Use structure in the objective:
    - For $f_i(x) = L(a_i^T x)$, only need to store N values of $a_i^T x$.

Minimizing Finite Sums: Dealing with the Memory

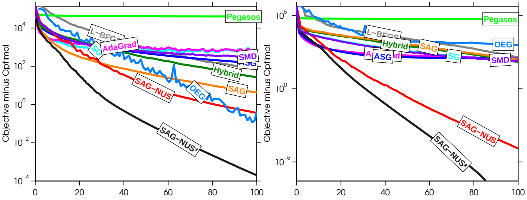
- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
  - Use mini-batches.
  - Use structure in the objective:
    - For $f_i(x) = L(a_i^T x)$, only need to store N values of $a_i^T x$.
    - For CRFs, only need to store marginals of parts.



(optical character and named-entity recognition tasks)

Minimizing Finite Sums: Dealing with the Memory

- A major disadvantage of SAG is the memory requirement.
- There are several ways to avoid this:
  - Use mini-batches.
  - Use structure in the objective:
    - For $f_i(x) = L(a_i^T x)$, only need to store N values of $a_i^T x$.
    - For CRFs, only need to store marginals of parts.



(optical character and named-entity recognition tasks)

- If the above don't work, use SVRG...

Stochastic Variance-Reduced Gradient

SVRG algorithm:

- Start with x_0
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N}\sum_{i=1}^{N} f_i'(x_s)$
  - $x^0 = x_s$

Stochastic Variance-Reduced Gradient

SVRG algorithm:

- Start with x_0
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f_i'(x_s)$
  - $x^0 = x_s$
  - for $t = 1, 2, \ldots m$
    - Randomly pick $i_t \in \{1, 2, \ldots, N\}$
    - $x^t = x^{t-1} - \alpha_t (f_{i_t}'(x^{t-1}) - f_{i_t}'(x_s) + d_s)$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$.

Stochastic Variance-Reduced Gradient

SVRG algorithm:

- Start with x_0
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N}\sum_{i=1}^{N} f_i'(x_s)$
  - $x^0 = x_s$
  - for $t = 1, 2, \ldots m$
    - Randomly pick $i_t \in \{1, 2, \ldots, N\}$
    - $x^t = x^{t-1} - \alpha_t(f_{i_t}'(x^{t-1}) - f_{i_t}'(x_s) + d_s)$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$.

Requires 2 gradients per iteration and occasional full passes, but only requires storing d_s and x_s.

Stochastic Variance-Reduced Gradient

SVRG algorithm:

- Start with x_0
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N}\sum_{i=1}^{N} f_i'(x_s)$
  - $x^0 = x_s$
  - for $t = 1, 2, \ldots m$
    - Randomly pick $i_t \in \{1, 2, \ldots, N\}$
    - $x^t = x^{t-1} - \alpha_t(f_{i_t}'(x^{t-1}) - f_{i_t}'(x_s) + d_s)$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$.

Requires 2 gradients per iteration and occasional full passes, but only requires storing d_s and x_s.

Practical issues similar to SAG (acceleration versions, automatic step-size/termination, handles sparsity/regularization, non-uniform sampling, mini-batches).

Conclusions

- Stochastic methods require 1 gradient per iteration but slow convergence.

- Deterministic methods are fast but requires N gradients per iteration.

- SAG, SVRG, and similar methods achieve faster convergence rate with few gradient evaluations