Optimization for Machine Learning

(Large-scale methods - B)

SUVRIT SRA

LIDS, Massachusetts Institute of Technology

PKU Summer School on Data Science (July 2017)

Large-scale ML

Regularized Empirical Risk Minimization

$$\min_{w} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^T x_i) + \lambda r(w).$$

This is the $f(w) + r(w)$ "composite objective" form we saw.
(e.g., regression, logistic regression, lasso, CRFs, etc.)

Large-scale ML

Regularized Empirical Risk Minimization

$$\min_{w} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^T x_i) + \lambda r(w).$$

This is the $f(w) + r(w)$ “composite objective” form we saw.
(e.g., regression, logistic regression, lasso, CRFs, etc.)

- training data: $(x_i, y_i) \in \mathbb{R}^d \times \mathcal{Y}$ (i.i.d.)
- large-scale ML: Both d and n are large:
  - ▶ d: dimension of each input sample
  - ▶ n: number of training data points / samples
- Assume training data “sparse”; so total datasize $\ll dn$.
- Running time $O(\#\text{nnz})$

Finite-sum problems

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

Finite-sum problems

$$\min_{x\in\mathbb{R}^d} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

Gradient / subgradient methods

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k \nabla f(x_k) \\ x_{k+1} &= x_k - \alpha_k g(x_k), \quad g \in \partial f(x_k) \\ x_{k+1} &= \operatorname{prox}_{\alpha_k r}(x_k - \alpha_k \nabla f(x_k)) \end{aligned}$$

Stochastic gradient

At iteration k, we randomly pick an integer

$$i(k) \in \{1, 2, \ldots, m\}$$

$$x_{k+1} = x_k - \alpha_k \nabla f_{i(k)}(x_k)$$

- The update requires only gradient for $f_{i(k)}$
- Uses unbiased estimate $\mathbb{E}[\nabla f_{i(k)}] = \nabla f$
- One iteration now n times faster using $\nabla f(x)$
- But how many iterations do we need?

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum\nolimits_{i=1}^{m}(a_i x - b_i)^2$$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Solving $f'(x) = 0$ we obtain

$$x^* = \frac{\sum_i a_i b_i}{\sum_i a_i^2}$$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2} \sum_{i=1}^{m} (a_i x - b_i)^2$$

- Solving $f'(x) = 0$ we obtain

$$x^* = \frac{\sum_i a_i b_i}{\sum_i a_i^2}$$

- Minimum of a single $f_i(x) = \frac{1}{2}(a_i x - b_i)^2$ is $x_i^* = b_i / a_i$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Solving $f'(x) = 0$ we obtain

$$x^* = \frac{\sum_i a_i b_i}{\sum_i a_i^2}$$

- Minimum of a single $f_i(x) = \frac{1}{2}(a_i x - b_i)^2$ is $x_i^* = b_i/a_i$
- Notice now that

$$x^* \in [\min_i x_i^*, \max_i x_i^*] =: R$$

(Use: $\sum_i a_i b_i = \sum_i a_i^2 (b_i/a_i)$)

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2} \sum_{i=1}^{m} (a_i x - b_i)^2$$

- Notice: $x^* \in \left[\min_i x_i^*, \max_i x_i^*\right] =: R$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum_{i=1}^{m}(a_i x - b_i)^2$$

- Notice: $x^* \in \left[\min_i x_i^*, \max_i x_i^*\right] =: R$
- If we have a scalar x that lies outside R?
- We see that

$$\nabla f_i(x) = a_i(a_i x - b_i)$$
$$\nabla f(x) = \sum_i a_i(a_i x - b_i)$$

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \frac{1}{2} \sum_{i=1}^{m} (a_i x - b_i)^2$$

- Notice: $x^* \in \left[\min_i x_i^*, \max_i x_i^*\right] =: R$
- If we have a scalar x that lies outside R?
- We see that

$$\nabla f_i(x) = a_i(a_i x - b_i)$$
$$\nabla f(x) = \sum_i a_i(a_i x - b_i)$$

- $\nabla f_i(x)$ has **same sign** as $\nabla f(x)$. So using $\nabla f_i(x)$ **instead** of $\nabla f(x)$ also ensures progress.

Example (Bertsekas)

- Assume all variables involved are **scalars**.

$$\min \quad f(x) = \tfrac{1}{2}\sum\nolimits_{i=1}^{m} (a_i x - b_i)^2$$

- Notice: $x^* \in \left[\min_i x_i^*, \max_i x_i^*\right] =: R$
- If we have a scalar x that lies outside R?
- We see that

$$\begin{aligned} \nabla f_i(x) &= a_i(a_i x - b_i) \\ \nabla f(x) &= \sum\nolimits_i a_i(a_i x - b_i) \end{aligned}$$

- $\nabla f_i(x)$ has **same sign** as $\nabla f(x)$. So using $\nabla f_i(x)$ **instead** of $\nabla f(x)$ also ensures progress.
- But once inside region R, **no guarantee** that incremental method will make progress towards optimum.

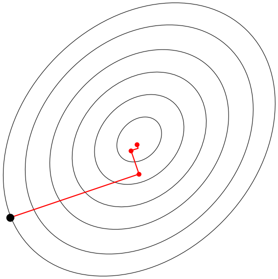
Iteration Complexity

Iteration complexity: smooth problems

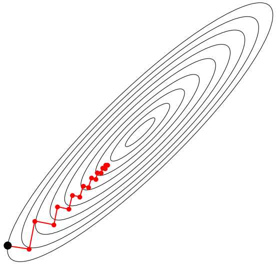
- **Assumption**: f convex and L-smooth on $\mathbb{R}^d$
- **Gradient descent:** $\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1})$

  $g(\theta_t) - g(x^*) \leqslant O(1/t)$

  $g(\theta_t) - g(x^*) \leqslant O(e^{-t(\mu/L)}) = O(e^{-t/\kappa})$ if μ-strongly convex



(small $\kappa = L/\mu$)



(large $\kappa = L/\mu$)

Iteration complexity: smooth problems

- **Assumption**: f convex and L-smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1})$
  $O(1/t)$ convergence rate for convex functions
  $O(e^{-\frac{t}{\kappa}})$ if strongly-convex $\Leftrightarrow$ complexity $= O(nd \cdot \kappa \log \frac{1}{\varepsilon})$

Iteration complexity: smooth problems

- **Assumption**: f convex and L-smooth on $\mathbb{R}^d$
- **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1})$

  $O(1/t)$ convergence rate for convex functions

  $O(e^{-\frac{t}{\kappa}})$ if strongly-convex $\Leftrightarrow$ complexity $= O(nd \cdot \kappa \log \frac{1}{\varepsilon})$

▶ **Key insights for ML (Bottou and Bousquet, 2008)**

1. No need to optimize below statistical error

Iteration complexity: smooth problems

– **Assumption**: f convex and L-smooth on $\mathbb{R}^d$
– **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1})$
  $O(1/t)$ convergence rate for convex functions
  $O(e^{-\frac{t}{\kappa}})$ if strongly-convex $\Leftrightarrow$ complexity $= O(nd \cdot \kappa \log \frac{1}{\varepsilon})$

▶ **Key insights for ML (Bottou and Bousquet, 2008)**

1. No need to optimize below statistical error
2. Cost functions are averages

Iteration complexity: smooth problems

– **Assumption**: f convex and L-smooth on $\mathbb{R}^d$

– **Gradient descent**: $\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1})$
$O(1/t)$ convergence rate for convex functions
$O(e^{-\frac{t}{\kappa}})$ if strongly-convex $\Leftrightarrow$ complexity $= O(nd \cdot \kappa \log \frac{1}{\varepsilon})$

▶ **Key insights for ML (Bottou and Bousquet, 2008)**

1. No need to optimize below statistical error
2. Cost functions are averages
3. Testing error is more important than training error

Stochastic gradient descent for finite sums

$$\min_{\theta \in \mathbb{R}^d} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

- **Iteration**: $\theta_t = \theta_{t-1} - \gamma_k f'_{i(t)}(\theta_{t-1})$
  - Sampling with replacement: $i(t) \sim \text{Unif}(\{1, \ldots, n\})$
  - Polyak-Ruppert averaging: $\bar{\theta}_t = \frac{1}{t+1} \sum_{u=0}^{t} \theta_u$
- **Convergence rate** if each f_i is convex L-smooth and f μ-strongly-convex:

$$\mathbb{E}[g(\bar{\theta}_t) - g(\theta^*)] \leqslant \begin{cases} O(1/\sqrt{k}) & \text{if } \gamma_k = 1/(L\sqrt{k}) \\ O(L/(\mu k)) = O(\kappa/k) & \text{if } \gamma_k = 1/(\mu k) \end{cases}$$

Stochastic vs. deterministic – strongly cvx

- Min $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i, \theta)\big) + \lambda\Omega(\theta)$
- Batch gradient descent:

$$\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma_k}{n}\sum_{i=1}^{n} f_i'(\theta_{t-1})$$

– Linear (e.g., exponential) convergence rate in $O(e^{-t/\kappa})$

– Iteration complexity is linear in n

Stochastic vs. deterministic – strongly cvx

- Min $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i, \theta)\big) + \lambda\Omega(\theta)$
- Batch gradient descent:

$$\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma_k}{n}\sum_{i=1}^{n} f_i'(\theta_{t-1})$$

  – Linear (e.g., exponential) convergence rate in $O(e^{-t/\kappa})$

  – Iteration complexity is linear in n
- Stochastic gradient descent: $\theta_t = \theta_{t-1} - \gamma_k f'_{i(t)}(\theta_{t-1})$

  – Sampling with replacement: $i(t)$ random element of $\{1, \ldots, n\}$

  – Convergence rate in $O(\kappa/t)$

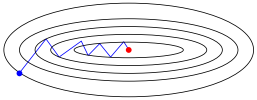
  – Iteration complexity is independent of n

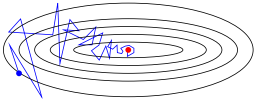
Stochastic vs. deterministic – strongly cvx

- Min $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$ with $f_i(\theta) = \ell\big(y_i, h(x_i, \theta)\big) + \lambda\Omega(\theta)$
- Batch gradient descent:

$$\theta_t = \theta_{t-1} - \gamma_k g'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma_k}{n}\sum_{i=1}^{n} f_i'(\theta_{t-1})$$

  – Linear (e.g., exponential) convergence rate in $O(e^{-t/\kappa})$
  – Iteration complexity is linear in n
- Stochastic gradient descent: $\theta_t = \theta_{t-1} - \gamma_k f'_{i(t)}(\theta_{t-1})$
  – Sampling with replacement: $i(t)$ random element of $\{1, \ldots, n\}$
  – Convergence rate in $O(\kappa/t)$
  – Iteration complexity is independent of n


GD


SGD

Stochastic gradient

Method	Assumptions	Full	Stochastic
Subgradient	convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
Subgradient	strongly cvx	$O(1/k)$	$O(1/k)$

So using stochastic subgradient, solve n times faster.

Stochastic gradient

Method	Assumptions	Full	Stochastic
Subgradient	convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
Subgradient	strongly cvx	$O(1/k)$	$O(1/k)$

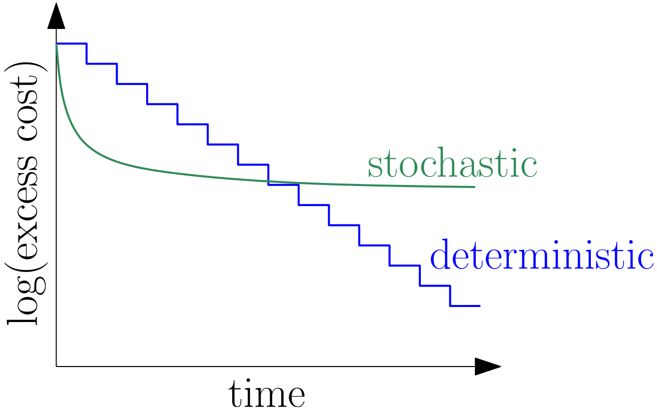
So using stochastic subgradient, solve n times faster.

Method	Assumptions	Full	Stochastic
Gradient	convex	$O(1/k)$	$O(1/\sqrt{k})$
Gradient	strongly cvx	$O((1-\mu/L)^k)$	$O(1/k)$

– For smooth problems, stochastic gradient needs more iterations
– Widely used in ML, rapid initial convergence
– Several speedup techniques studied, but worst case remains same

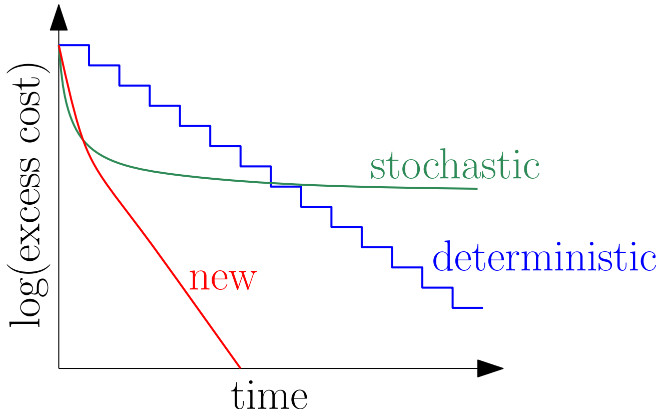
Stochastic vs. deterministic methods

Goal = best of both worlds: Linear rate with $O(d)$ iteration cost
Simple choice of step size



Stochastic vs. deterministic methods

Goal = best of both worlds: Linear rate with $O(d)$ iteration cost
Simple choice of step size



Linearly convergent stochastic gradient algorithms

- **Many related algorithms**
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SDCA (Shalev-Shwartz and Zhang, 2013)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - MISO (Mairal, 2015)
  - Finito (Defazio et al., 2014b)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014a)
  - $\cdots$
- **Similar rates of convergence and iterations**

Linearly convergent stochastic gradient algorithms

- **Many related algorithms**
  - SAG (Le Roux, Schmidt, and Bach, 2012)
  - SDCA (Shalev-Shwartz and Zhang, 2013)
  - SVRG (Johnson and Zhang, 2013; Zhang et al., 2013)
  - MISO (Mairal, 2015)
  - Finito (Defazio et al., 2014b)
  - SAGA (Defazio, Bach, and Lacoste-Julien, 2014a)
  - ...
- **Similar rates of convergence and iterations**

– **Different interpretations and proofs / proof lengths**
  - Lazy gradient evaluations
  - Variance reduction

Generic matlab code

```
function [x,f] = gradientDescent(x0)

    fx  = @(x)  objfn(x);   % handle to f(x)
    gfx = @(x)  grad(x);    % handle to nabla f(x)

    x=x0;                   % input starting point
    maxiter = 100;          % tunable parameter

    for k=1:maxiter         % or other criterion
       g = gfx(x);          % compute gradient at x
       al = stepSize(x);    % compute a stepsize
       x = x - al*g;        % perform update
       fprintf('Iter: %d\t Obj: %d\n', fx(x));
    end
end
```

Hybrid methods

► Hybrid of stochastic gradient with full gradient.

Stochastic Average Gradient (SAG) (Le Roux, Schmidt, Bach 2012)

- store the gradients of ∇f_i for $i = 1, .., n$
- Select uniformly at random $i(k) \in \{1, \dots, n\}$
- Perform the update

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} y_i^k \quad y_i^k = \begin{cases} \nabla f_i(x_k) & \text{if } i = i(k) \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

Hybrid methods

▶ Hybrid of stochastic gradient with full gradient.

Stochastic Average Gradient (SAG) (Le Roux, Schmidt, Bach 2012)

- store the gradients of ∇f_i for $i = 1, .., n$
- Select uniformly at random $i(k) \in \{1, \ldots, n\}$
- Perform the update

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} y_i^k \quad y_i^k = \begin{cases} \nabla f_i(x_k) & \text{if } i = i(k) \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

- Randomized / stochastic version of incremental gradient method of Blatt et al (2008)
- Storage overhead; acceptable in some ML settings:
  - $f_i(x) = \ell(l_i, x^T\Phi(a_i))$, $\nabla f_i(x) = \nabla \ell(l_i, x^T\Phi(a_i))\Phi(a_i)$
  - Store only n scalars (since depends only on $x^T a_i$)

SAG

Method	Assumptions	Rate
Gradient	convex	$O(1/k)$
Gradient	strongly cvx	$O((1-\mu/L)^k)$
Stochastic	strongly cvx	$O(1/k)$
SAG	strongly convex	$O((1-\min\{\frac{\mu}{n}, \frac{1}{8n}\})^k)$

This speedup also observed in practice

Complicated convergence analysis

Similar rates for many other methods

- stochastic dual coordinate (SDCA); [Shalev-Shwartz, Zhang, 2013]
- stochastic variance reduced gradient (SVRG); [Johnson, Zhang, 2013]
- proximal SVRG [Xiao, Zhang, 2014]
- hybrid of SAG and SVRG, SAGA (also proximal); [Defazio et al, 2014]
- accelerated versions [Lin, Mairal, Harchoui; 2015]
- asynchronous hybrid SVRG [Reddi et al. 2015]
- incremental Newton method, S2SGD and MS2GD, . . .

Running-time comparisons (strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth and f is μ-strongly convex

Running-time comparisons (strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth and f is μ-strongly convex

Stochastic gradient descent	$d\times$	$\frac{L}{\mu}$	$\times \quad \frac{1}{\varepsilon}$
Gradient descent	$d\times$	$n\frac{L}{\mu}$	$\times \log\frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times$	$n\sqrt{\frac{L}{\mu}}$	$\times \log\frac{1}{\varepsilon}$
SAG/SVRG	$d\times$	$(n+\frac{L}{\mu})$	$\times \log\frac{1}{\varepsilon}$

Running-time comparisons (strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth and f is μ-strongly convex

Stochastic gradient descent	$d\times$	$\frac{L}{\mu}$	$\times \quad \frac{1}{\varepsilon}$
Gradient descent	$d\times$	$n\frac{L}{\mu}$	$\times \log \frac{1}{\varepsilon}$
Accelerated gradient descent	$d\times$	$n\sqrt{\frac{L}{\mu}}$	$\times \log \frac{1}{\varepsilon}$
SAG/SVRG	$d\times$	$(n + \frac{L}{\mu})$	$\times \log \frac{1}{\varepsilon}$

- **Beating two lower bounds** (Nemirovski and Yudin, 1983; Nesterov, 2004): with additional assumptions
  (1) stochastic gradient: exponential rate for finite sums
  (2) full gradient: better exponential rate using the sum structure

Running-time comparisons (non-strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth
  - Ill conditioned problems: f may not be strongly-convex

Stochastic gradient descent	$d\times$	$1/\varepsilon^2$
Gradient descent	$d\times$	n/ε
Accelerated gradient descent	$d\times$	$n/\sqrt{\varepsilon}$
SAG/SVRG	$d\times$	$\sqrt{n}/\varepsilon$

Running-time comparisons (non-strongly-convex)

- **Assumptions**: $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$
  - Each f_i convex L-smooth
  - Ill conditioned problems: f may not be strongly-convex

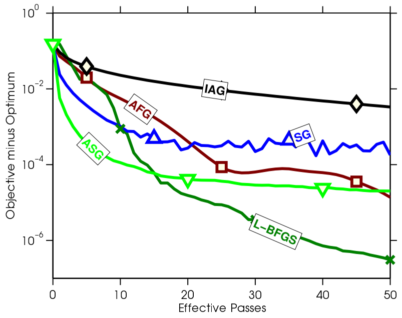
Stochastic gradient descent	$d\times$	$1/\varepsilon^2$
Gradient descent	$d\times$	n/ε
Accelerated gradient descent	$d\times$	$n/\sqrt{\varepsilon}$
SAG/SVRG	$d\times$	$\sqrt{n}/\varepsilon$

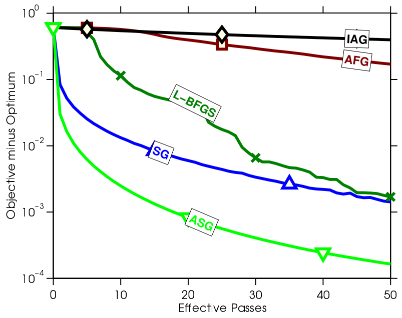
- Adaptivity to potentially hidden strong convexity
- No need to know the local/global strong-convexity constant

Experimental results (logistic regression)

`quantum` **dataset**
$(n = 50\,000, d = 78)$

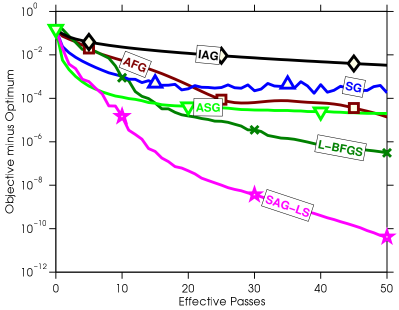
`rcv1` **dataset**
$(n = 697\,641, d = 47\,236)$



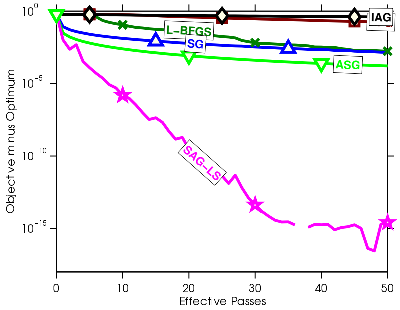


Experimental results (logistic regression)

`quantum` dataset
$(n = 50\,000, d = 78)$



`rcv1` dataset
$(n = 697\,641, d = 47\,236)$



Key Idea: Variance reduction

Principle: reducing variance of sample of X by using a sample from another random variable Y with known expectation

$$Z_\alpha = \alpha(X - Y) + \mathbb{E}Y$$

- $\mathbb{E}Z_\alpha = \alpha\mathbb{E}X + (1-\alpha)\mathbb{E}Y$
- $\text{var}(Z_\alpha) = \alpha^2\big[\text{var}(X) + \text{var}(Y) - 2\text{cov}(X, Y)\big]$
- $\alpha = 1$: no bias, $\alpha < 1$: potential bias (but reduced variance)
- Useful if Y positively correlated with X

Key Idea: Variance reduction

Principle: reducing variance of sample of X by using a sample from another random variable Y with known expectation

$$Z_\alpha = \alpha(X - Y) + \mathbb{E}Y$$

- $\mathbb{E}Z_\alpha = \alpha\mathbb{E}X + (1-\alpha)\mathbb{E}Y$
- $\mathrm{var}(Z_\alpha) = \alpha^2\big[\mathrm{var}(X) + \mathrm{var}(Y) - 2\mathrm{cov}(X,Y)\big]$
- $\alpha = 1$: no bias, $\alpha < 1$: potential bias (but reduced variance)
- Useful if Y positively correlated with X

Application to gradient estimation (Johnson and Zhang, 2013; Zhang, Mahdavi, and Jin, 2013)

- SVRG: $X = f'_{i(t)}(\theta_{t-1})$, $Y = f'_{i(t)}(\tilde{\theta})$, $\alpha = 1$, with $\tilde{\theta}$ stored
- $\mathbb{E}Y = \frac{1}{n}\sum_{i=1}^{n} f'_i(\tilde{\theta})$ full gradient at $\tilde{\theta}$;
  $X - Y = f'_{i(t)}(\theta_{t-1}) - f'_{i(t)}(\tilde{\theta})$

Stochastic variance reduced gradient (SVRG)

```
- Initialize $\tilde{\theta} \in \mathbb{R}^d$
- For $i_{\text{epoch}} = 1$ to # of epochs
    - Compute all gradients $f_i'(\tilde{\theta})$ ; store $g'(\tilde{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i'(\tilde{\theta})$
    - Initialize $x_0 = \tilde{\theta}$
    - For $t = 1$ to length of epochs
          $\theta_t = \theta_{t-1} - \gamma\Big[g'(\tilde{\theta}) + \big(f_{i(t)}'(\theta_{t-1}) - f_{i(t)}'(\tilde{\theta})\big)\Big]$
    - Update $\tilde{\theta} = \theta_t$
- Output: $\tilde{\theta}$
```

- two gradient evaluations per inner step; no need to store gradients (SAG needs storage)
- Two parameters: length of epochs + step-size γ
- Same linear convergence rate as SAG, simpler proof

SVRG vs. SAGA

- **SAGA update**:
$$\theta_t = \theta_{t-1} - \gamma\left[\frac{1}{n}\sum_{i=1}^{n} y_i^{t-1} + \left(f'_{i(t)}(\theta_{t-1}) - y_{i(t)}^{t-1}\right)\right]$$
- **SVRG update**:
$$\theta_t = \theta_{t-1} - \gamma\left[\frac{1}{n}\sum_{i=1}^{n} f'_i(\tilde{\theta}) + \left(f'_{i(t)}(\theta_{t-1}) - f'_{i(t)}(\tilde{\theta})\right)\right]$$

	SAGA	SVRG
Storage of gradients	**yes**	**no**
Epoch-based	no	yes
Parameters	step-size	step-size & epoch lengths
Gradient evaluations per step	1	at least 2
Adaptivity to strong-convexity	yes	no
Robustness to ill-conditioning	yes	no

Proximal extensions

– **Composite optimization problems**: $\min_{x \in \mathbb{R}^d} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\theta) + h(\theta)$

- f_i smooth and convex
- h convex, potentially non-smooth
- Constrained optimization: h an indicator function
- Sparsity-inducing norms, e.g., $h(\theta) = \|\theta\|_1$

– **Proximal methods (a.k.a. splitting methods)**

- Projection / soft-thresholding step after gradient update
- See, e.g., Combettes and Pesquet (2011); Bach, Jenatton, Mairal, and Obozinski (2012); Parikh and Boyd (2014)

– **Directly extends to variance-reduced gradient techniques**
Same rates of convergence

SGD minimizes the testing cost!

- **Goal**: minimize $g(\theta) = \mathbb{E}_{p(x,y)} \ell(y, \theta^\top \Phi(x))$
  - Given n independent samples $(x_i, y_i)_{i=1}^n$, from $p(x, y)$
  - Given a single pass of stochastic gradient descent
  - Bounds on the excess testing cost $\mathbb{E} g(\bar{\theta}_n) - \inf_{x \in \mathbb{R}^d} g(\theta)$

- **Optimal convergence rates**: $O(1/\sqrt{n})$ and $O(1/(n\mu))$
  - Optimal for non-smooth (Nemirovski and Yudin, 1983)
  - Attained by averaged SGD with decaying step-sizes

References

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4 (1):1–106, 2012.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Adv. NIPS*, 2008.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014a.

A. Defazio, J. Domke, and T. S. Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *Proc. ICML*, 2014b.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 2013.

N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

A. S. Nemirovski and D. B. Yudin. *Problem complexity and method efficiency in optimization.* Wiley & Sons, 1983.

Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Kluwer, 2004.

N. Parikh and S. P. Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14 (Feb):567–599, 2013.

L. Zhang, M. Mahdavi, and R. Jin. Linear convergence with condition number independent access of full gradients. In *Advances in Neural Information Processing Systems*, 2013.