Recent Progresses in Stochastic Algorithms for Big Data Optimization

Tong Zhang

Rutgers University & Baidu Inc.

collaborators: Shai Shalev-Shwartz, Rie Johnson, Lin Xiao, Ohad Shamir and Nathan Srebro

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Strategies for distributed computing
  - algorithm 4: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

Mathematical Problem

Big Data Optimization Problem in machine learning:

$$\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

Special structure: sum over data.

Big data (n large) requires distrubuted training.

Assumptions on loss function

- λ-strong convexity:

$$f(w') \geq f(w) + \nabla f(w)^\top (w' - w) + \frac{\lambda}{2}\|w' - w\|_2^2$$

- L-smoothness:

$$f_i(w') \leq f_i(w) + \nabla f_i(w)^\top (w' - w) + \frac{L}{2}\|w' - w\|_2^2$$

Example: Computational Advertizing

- Large scale regularized logistic regression

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \left[\underbrace{\ln(1+e^{-w^\top x_i y_i}) + \frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}\right]$$

  - data (x_i, y_i) with $y_i \in \{\pm 1\}$
  - parameter vector w.
- λ strongly convex and $L = 0.25\max_i \|x_i\|_2^2 + \lambda$ smooth.

Example: Computational Advertizing

- Large scale regularized logistic regression

$$\min_w \frac{1}{n}\sum_{i=1}^{n}\left[\underbrace{\ln(1+e^{-w^\top x_i y_i})+\frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}\right]$$

  - data (x_i, y_i) with $y_i \in \{\pm 1\}$
  - parameter vector w.
- λ strongly convex and $L = 0.25\max_i \|x_i\|_2^2 + \lambda$ smooth.

- big data: $n \sim 10 - 100$ billion
- high dimension: $\dim(x_i) \sim 10 - 100$ billion

Example: Computational Advertizing

- Large scale regularized logistic regression

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \left[\underbrace{\ln(1+e^{-w^\top x_i y_i}) + \frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}\right]$$

  - data (x_i, y_i) with $y_i \in \{\pm 1\}$
  - parameter vector w.
- λ strongly convex and $L = 0.25\max_i \|x_i\|_2^2 + \lambda$ smooth.

- big data: $n \sim 10 - 100$ billion
- high dimension: $\dim(x_i) \sim 10 - 100$ billion

How to solve big optimization problems efficiently?

Statistical Thinking: sampling

- Objective function:

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

sample objective function: only optimize approximate objective

Statistical Thinking: sampling

- Objective function:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

  sample objective function: only optimize approximate objective
- 1st order gradient

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w)$$

  sample 1st order gradient (stochastic gradient): converges to exact optimal – variance reduction: fast rate

Statistical Thinking: sampling

- Objective function:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

  sample objective function: only optimize approximate objective
- 1st order gradient

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w)$$

  sample 1st order gradient (stochastic gradient): converges to exact optimal – variance reduction: fast rate
- 2nd order gradient

$$\nabla^2 f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(w)$$

  sample 2nd order gradient (stochastic Newton): converges to exact optimal with fast rate, distributed computing

Batch Optimization Method: Gradient Descent

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

Gradient Descent (GD):

$$w_k = w_{k-1} - \eta_k \nabla f(w_{k-1}).$$

How fast does this method converge to the optimal solution?

Batch Optimization Method: Gradient Descent

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

Gradient Descent (GD):

$$w_k = w_{k-1} - \eta_k \nabla f(w_{k-1}).$$

How fast does this method converge to the optimal solution?

- General result: converge to local minimum under suitable conditions.
- Convergence rate depends on conditions of $f(\cdot)$. For λ-strongly convex and L-smooth problems, it is linear rate:

$$f(w_k) - f(w_*) = O((1-\rho)^k),$$

where $\rho = O(\lambda/L)$ is the inverse condition number

Stochastic Approximate Gradient Computation

If

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w),$$

GD requires the computation of full gradient, which is extremely costly

$$\nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w)$$

Stochastic Approximate Gradient Computation

If

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w),$$

GD requires the computation of full gradient, which is extremely costly

$$\nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w)$$

Idea: stochastic optimization employs random sample (mini-batch) B to approximate

$$\nabla f(w) \approx \frac{1}{|B|} \sum_{i \in B} \nabla f_i(w)$$

- It is an unbiased estimator
- more efficient computation but introduces variance

SGD versus GD

SGD:

- faster computation per step
- Sublinear convergence: due to the variance of gradient approximation.

$$f(w_t) - f(w_*) = \tilde{O}(1/t).$$

GD:

- slower computation per step
- Linear convergence:

$$f(w_t) - f(w_*) = O((1-\rho)^t).$$

Improving SGD via Variance Reduction

- GD converges fast but computation is slow
- SGD computation is fast but converges slowly
  - slow convergence due to inherent variance
- SGD as a statistical estimator of gradient:
  - let $\mathbf{g}_i = \nabla f_i$.
  - unbaisedness: $\mathbf{E}\, \mathbf{g}_i = \frac{1}{n} \sum_{i=1}^n \mathbf{g}_i = \nabla f$.
  - error of using $\mathbf{g}_i$ to approx ∇f: variance $\mathbf{E}\|\mathbf{g}_i - \mathbf{E}\mathbf{g}_i\|_2^2$.

Improving SGD via Variance Reduction

- GD converges fast but computation is slow
- SGD computation is fast but converges slowly
  - slow convergence due to inherent variance
- SGD as a statistical estimator of gradient:
  - let $\mathbf{g}_i = \nabla f_i$.
  - unbaisedness: $\mathbf{E}\, \mathbf{g}_i = \frac{1}{n}\sum_{i=1}^n \mathbf{g}_i = \nabla f$.
  - error of using $\mathbf{g}_i$ to approx ∇f: variance $\mathbf{E}\|\mathbf{g}_i - \mathbf{E}\mathbf{g}_i\|_2^2$.
- Statistical thinking:
  - relating variance to optimization
  - design other unbiased gradient estimators with smaller variance

Relating Statistical Variance to Optimization

Want to optimize

$$\min_w f(w)$$

Full gradient $\nabla f(w)$.

Relating Statistical Variance to Optimization

Want to optimize

$$\min_w f(w)$$

Full gradient $\nabla f(w)$.

Given unbiased random estimator $\mathbf{g}_i$ of $\nabla f(w)$, and SGD rule

$$w \to w - \eta \mathbf{g}_i,$$

reduction of objective is

$$\mathbf{E} f(w - \eta \mathbf{g}_i) \leq \underbrace{f(w) - (\eta - \eta^2 L/2)\|\nabla f(w)\|_2^2}_{\text{non-random}} + \frac{\eta^2 L}{2} \underbrace{\mathbf{E}\|\mathbf{g} - \mathbf{E}\mathbf{g}\|_2^2}_{\text{variance}}.$$

Relating Statistical Variance to Optimization

Want to optimize

$$\min_w f(w)$$

Full gradient $\nabla f(w)$.

Given unbiased random estimator $\mathbf{g}_i$ of $\nabla f(w)$, and SGD rule

$$w \to w - \eta \mathbf{g}_i,$$

reduction of objective is

$$\mathbf{E} f(w - \eta \mathbf{g}_i) \leq \underbrace{f(w) - (\eta - \eta^2 L/2)\|\nabla f(w)\|_2^2}_{\text{non-random}} + \frac{\eta^2 L}{2} \underbrace{\mathbf{E}\|\mathbf{g} - \mathbf{E}\mathbf{g}\|_2^2}_{\text{variance}}.$$

Smaller variance implies bigger reduction

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Strategies for distributed computing
  - algorithm 4: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

Stochastic Variance Reduced Gradient (SVRG) I

Objective function

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \frac{1}{n}\sum_{i=1}^{n} \tilde{f}_i(w),$$

where

$$\tilde{f}_i(w) = f_i(w) - \underbrace{(\nabla f_i(\tilde{w}) - \nabla f(\tilde{w}))^\top w}_{\text{sum to zero}}.$$

Pick $\tilde{w}$ to be an approximate solution (close to w_*).
The SVRG rule (control variates) is

$$w_t = w_{t-1} - \eta_t \nabla \tilde{f}_i(w_{t-1}) = w_{t-1} - \eta_t \underbrace{[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})]}_{\text{small variance}}.$$

Stochastic Variance Reduced Gradient (SVRG) II

Assume that $\tilde{w} \approx w_*$ and $w_{t-1} \approx w_*$. Then

$$\nabla f(\tilde{w}) \approx \nabla f(w_*) = 0 \qquad \nabla f_i(w_{t-1}) \approx \nabla f_i(\tilde{w}).$$

This means

$$\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w}) \to 0.$$

It is possible to choose a constant step size $\eta_t = \eta$ instead of requiring $\eta_t \to 0$.
One can achieve comparable linear convergence with SVRG:

$$Ef(w_t) - f(w_*) = O((1 - \tilde{\rho})^t),$$

where $\tilde{\rho} = O(\lambda n/(L + \lambda n)$; convergence is faster than GD.

Compare SVRG to Batch Gradient Descent Algorithm

Number of examples needed to achieve ϵ accuracy:

- Batch GD: $\tilde{O}(n \cdot L/\lambda \log(1/\epsilon))$
- SVRG: $\tilde{O}((n + L/\lambda) \log(1/\epsilon))$

Assume L-smooth loss:

$$\|\nabla f_i(w) - \nabla f_i(w')\| \leq L\|w - w'\|$$

and λ strong convex objective:

$$\|\nabla f(w) - \nabla f(w')\| \geq \lambda\|w - w'\|$$

The gain of SVRG over batch algorithm is significant when n is large.

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Strategies for distributed computing
  - algorithm 4: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

Motivation of SDCA: regularized loss minimization

Assume we want to solve the Lasso problem:

$$\min_w \left[\frac{1}{n} \sum_{i=1}^{n} (w^\top x_i - y_i)^2 + \lambda \|w\|_1 \right]$$

Motivation of SDCA: regularized loss minimization

Assume we want to solve the Lasso problem:

$$\min_w \left[\frac{1}{n} \sum_{i=1}^{n} (w^\top x_i - y_i)^2 + \lambda \|w\|_1 \right]$$

or the ridge regression problem:

$$\min_w \left[\underbrace{\frac{1}{n} \sum_{i=1}^{n} (w^\top x_i - y_i)^2}_{\text{loss}} + \underbrace{\frac{\lambda}{2} \|w\|_2^2}_{\text{regularization}} \right]$$

Our goal: solve regularized loss minimization problems as fast as we can.

Motivation of SDCA: regularized loss minimization

Assume we want to solve the Lasso problem:

$$\min_w \left[\frac{1}{n} \sum_{i=1}^{n} (w^\top x_i - y_i)^2 + \lambda \|w\|_1 \right]$$

or the ridge regression problem:

$$\min_w \left[\underbrace{\frac{1}{n} \sum_{i=1}^{n} (w^\top x_i - y_i)^2}_{\text{loss}} + \underbrace{\frac{\lambda}{2} \|w\|_2^2}_{\text{regularization}} \right]$$

Our goal: solve regularized loss minimization problems as fast as we can.

- A good solution leads to stochastic algorithm called *proximal Stochastic Dual Coordinate Ascent* (Prox-SDCA).
- We show: fast convergence of SDCA for many regularized loss minimization problems in machine learning.

General Problem

Want to solve:

$$\min_w P(w) := \left[\frac{1}{n}\sum_{i=1}^{n}\phi_i(X_i^\top w) + \lambda g(w)\right],$$

where X_i are matrices; $g(\cdot)$ is strongly convex.
Examples:

- Multi-class logistic loss

$$\phi_i(X_i^\top w) = \ln\sum_{\ell=1}^{K}\exp(w^\top X_{i,\ell}) - w^\top X_{i,y_i}.$$

- $L_1 - L_2$ regularization

$$g(w) = \frac{1}{2}\|w\|_2^2 + \frac{\sigma}{\lambda}\|w\|_1$$

Dual Formulation

Primal:

$$\min_w P(w) := \left[\frac{1}{n}\sum_{i=1}^n \phi_i(X_i^\top w) + \lambda g(w)\right],$$

Dual:

$$\max_\alpha D(\alpha) := \left[\frac{1}{n}\sum_{i=1}^n -\phi_i^*(-\alpha_i) - \lambda g^*\left(\frac{1}{\lambda n}\sum_{i=1}^n X_i\alpha_i\right)\right]$$

with the relationship

$$w = \nabla g^*\left(\frac{1}{\lambda n}\sum_{i=1}^n X_i\alpha_i\right).$$

The convex conjugate (dual) is defined as $\phi_i^*(a) = \sup_z(az - \phi_i(z))$.

SDCA: randomly pick i and optimize $D(\alpha)$ by varying α_i while keeping other dual variables fixed.

Example: $L_1 - L_2$ Regularized Logistic Regression

Primal:

$$P(w) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ln(1 + e^{-w^\top X_i Y_i})}_{\phi_i(w)} + \underbrace{\frac{\lambda}{2} w^\top w + \sigma \|w\|_1}_{\lambda g(w)}.$$

Dual: with $\alpha_i Y_i \in [0, 1]$

$$D(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{-\alpha_i Y_i \ln(\alpha_i Y_i) - (1 - \alpha_i Y_i)\ln(1 - \alpha_i Y_i)}_{\phi_i^*(-\alpha_i)} - \frac{\lambda}{2}\|\mathrm{trunc}(v, \sigma/\lambda)\|_2^2$$

$$\text{s.t. } v = \frac{1}{\lambda n}\sum_{i=1}^{n} \alpha_i X_i; \qquad w = \mathrm{trunc}(v, \sigma/\lambda)$$

where

$$\mathrm{trunc}(u, \delta)_j = \begin{cases} u_j - \delta & \text{if } u_j > \delta \\ 0 & \text{if } |u_j| \le \delta \\ u_j + \delta & \text{if } u_j < -\delta \end{cases}$$

Proximal-SDCA for L_1-L_2 Regularization

Algorithm:

Keep dual α and $v = (\lambda n)^{-1} \sum_i \alpha_i X_i$

- Randomly pick i
- Find Δ_i by approximately maximizing:

$$-\phi_i^*(\alpha_i + \Delta_i) - \mathrm{trunc}(v, \sigma/\lambda)^\top X_i \, \Delta_i - \frac{1}{2\lambda n} \|X_i\|_2^2 \Delta_i^2,$$

  where $\phi_i^*(\alpha_i + \Delta) = (\alpha_i + \Delta) Y_i \ln((\alpha_i + \Delta) Y_i) + (1 - (\alpha_i + \Delta) Y_i) \ln(1 - (\alpha_i + \Delta) Y_i)$

- $\alpha = \alpha + \Delta_i \cdot e_i$
- $v = v + (\lambda n)^{-1} \Delta_i \cdot X_i$.

Let $w = \mathrm{trunc}(v, \sigma/\lambda)$.

Convergence rate

The number of iterations needed to achieve ϵ accuracy

- For $(1/\gamma)$-smooth loss:

$$\tilde{O}\left(\left(n+\frac{1}{\gamma\lambda}\right)\log\frac{1}{\epsilon}\right)$$

- For L-Lipschitz loss:

$$\tilde{O}\left(n+\frac{L^2}{\lambda\,\epsilon}\right)$$

Solving L_1 with Smooth Loss

Assume we want to solve L_1 regularization to accuracy ϵ with smooth ϕ_i:

$$\frac{1}{n}\sum_{i=1}^{n}\phi_i(w) + \sigma\|w\|_1.$$

Apply Prox-SDCA with extra term $0.5\lambda\|w\|_2^2$, where $\lambda = O(\epsilon)$:

- number of iterations needed is $\tilde{O}(n + 1/\epsilon)$.

Solving L_1 with Smooth Loss

Assume we want to solve L_1 regularization to accuracy ϵ with smooth ϕ_i:

$$\frac{1}{n}\sum_{i=1}^{n}\phi_i(w) + \sigma\|w\|_1.$$

Apply Prox-SDCA with extra term $0.5\lambda\|w\|_2^2$, where $\lambda = O(\epsilon)$:

- number of iterations needed is $\tilde{O}(n + 1/\epsilon)$.

Compare to Dual Averaging SGD (Xiao):

- number of iterations needed is $\tilde{O}(1/\epsilon^2)$.

Solving L_1 with Smooth Loss

Assume we want to solve L_1 regularization to accuracy ϵ with smooth ϕ_i:

$$\frac{1}{n}\sum_{i=1}^{n}\phi_i(w) + \sigma\|w\|_1.$$

Apply Prox-SDCA with extra term $0.5\lambda\|w\|_2^2$, where $\lambda = O(\epsilon)$:

- number of iterations needed is $\tilde{O}(n + 1/\epsilon)$.

Compare to Dual Averaging SGD (Xiao):

- number of iterations needed is $\tilde{O}(1/\epsilon^2)$.

Compare to batch FISTA (Nesterov's accelerated proximal gradient):

- number of iterations needed is $\tilde{O}(n/\sqrt{\epsilon})$.

Prox-SDCA wins in the statistically interesting regime: $\epsilon > \Omega(1/n^2)$

Solving L_1 with Smooth Loss

Assume we want to solve L_1 regularization to accuracy ϵ with smooth ϕ_i:

$$\frac{1}{n}\sum_{i=1}^{n}\phi_i(w) + \sigma\|w\|_1.$$

Apply Prox-SDCA with extra term $0.5\lambda\|w\|_2^2$, where $\lambda = O(\epsilon)$:

- number of iterations needed is $\tilde{O}(n + 1/\epsilon)$.

Compare to Dual Averaging SGD (Xiao):

- number of iterations needed is $\tilde{O}(1/\epsilon^2)$.

Compare to batch FISTA (Nesterov's accelerated proximal gradient):

- number of iterations needed is $\tilde{O}(n/\sqrt{\epsilon})$.

Prox-SDCA wins in the statistically interesting regime: $\epsilon > \Omega(1/n^2)$

Can design an accelerated prox-SDCA procedure always superior to FISTA

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Strategies for distributed computing
  - algorithm 4: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

Accelerated Prox-SDCA

Solving:

$$P(w) := \frac{1}{n}\sum_{i=1}^{n} \phi_i(X_i^\top w) + \lambda g(w)$$

- Convergence rate of Prox-SDCA depends on $O(1/\lambda)$
- Inferior to acceleration when λ is very small $\ll O(1/n)$, which has $O(1/\sqrt{\lambda})$ dependency

Accelerated Prox-SDCA

Solving:

$$P(w) := \frac{1}{n}\sum_{i=1}^{n} \phi_i(X_i^\top w) + \lambda g(w)$$

- Convergence rate of Prox-SDCA depends on $O(1/\lambda)$
- Inferior to acceleration when λ is very small $\ll O(1/n)$, which has $O(1/\sqrt{\lambda})$ dependency

Inner-outer Iteration Accelerated Prox-SDCA

- Pick a suitable $\kappa = \Theta(1/n)$ and β
- For $t = 2, 3 \ldots$ (outer iter)
  - Let $\tilde{g}_t(w) = \lambda g(w) + 0.5\kappa\|w\|_2^2 - \kappa w^\top y^{(t-1)}$ (κ-strongly convex)
  - Let $\tilde{P}_t(w) = P(w) - \lambda g(w) + \tilde{g}_t(w)$ (redefine $P(\cdot)$ – κ strongly convex)
  - Approximately solve $\tilde{P}_t(w)$ for $(w^{(t)}, \alpha^{(t)})$ with prox-SDCA (inner iter)
  - Let $y^{(t)} = w^{(t)} + \beta(w^{(t)} - w^{(t-1)})$ (acceleration)

Performance Comparisons

Problem	Algorithm	Runtime
SVM	SGD	$\frac{d}{\lambda\epsilon}$
	AGD (Nesterov)	$dn\sqrt{\frac{1}{\lambda\epsilon}}$
	Acc-Prox-SDCA	$d\left(n+\min\{\frac{1}{\lambda\epsilon},\sqrt{\frac{n}{\lambda\epsilon}}\}\right)$
Lasso	SGD and variants	$\frac{d}{\epsilon^2}$
	Stochastic Coordinate Descent	$\frac{dn}{\epsilon}$
	FISTA	$dn\sqrt{\frac{1}{\epsilon}}$
	Acc-Prox-SDCA	$d\left(n+\min\{\frac{1}{\epsilon},\sqrt{\frac{n}{\epsilon}}\}\right)$
Ridge Regression	Exact	d^2n+d^3
	SGD, SDCA	$d\left(n+\frac{1}{\lambda}\right)$
	AGD	$dn\sqrt{\frac{1}{\lambda}}$
	Acc-Prox-SDCA	$d\left(n+\min\{\frac{1}{\lambda},\sqrt{\frac{n}{\lambda}}\}\right)$

Summary of Sampling

- 1st order gradient

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w)$$

  sample 1st order gradient (stochastic gradient): variance reduction leads to fast linear convergence

- 2nd order gradient

$$\nabla^2 f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(w)$$

  sample 2nd order gradient (stochastic Newton): converges to exact optimal with fast rate in distributed computing setting

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization problem
  - 1st order stochastic gradient versus batch gradient: pros and cons
- Stochastic gradient algorithms with variance reduction
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Strategies for distributed computing
  - algorithm 4: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

Distributed Computing

Assume: data distributed over machines

- m processors
- each has n/m examples

Simple Computational Strategy — One Shot Averaging (OSA)

- run optimization on m machines separately
  - obtaining parameters $w^{(1)}, \ldots, w^{(m)}$
- average the parameters: I
  - $\bar{w} = m^{-1} \sum_{i=1}^{m} w^{(i)}$

Improvement

OSA Strategy's advantage:

- machines run independently
- simple and computationally efficient; asymptotically good in theory

Disadvantage:

- practically inferior to training all examples on a single machine

Improvement

OSA Strategy's advantage:

- machines run independently
- simple and computationally efficient; asymptotically good in theory

Disadvantage:

- practically inferior to training all examples on a single machine

Traditional solution in optimization: ADMM

New Idea: via 2nd order gradient sampling

- Distributed Approximate NEwton (DANE)

Distribution Scheme

Assume: data distributed over machines with decomposed problem

$$f(w) = \sum_{\ell=1}^{m} f^{(\ell)}(w).$$

- m processors
- each $f^{(\ell)}(w)$ has n/m randomly partitioned examples
- each machine holds a complete set of parameters

DANE

Starting with $\tilde{w}$ using OSA
Iterate

- Take $\tilde{w}$ and define

$$\tilde{f}^{(\ell)}(w) = f^{(\ell)}(w) - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w$$

- on each machine solves

$$w^{(\ell)} = \arg\min_w \tilde{f}^{(\ell)}(w)$$

  independently.
- Take partial average as the next $\tilde{w}$

DANE

Starting with $\tilde{w}$ using OSA
Iterate

- Take $\tilde{w}$ and define

$$\tilde{f}^{(\ell)}(w) = f^{(\ell)}(w) - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w$$

- on each machine solves

$$w^{(\ell)} = \arg\min_w \tilde{f}^{(\ell)}(w)$$

  independently.
- Take partial average as the next $\tilde{w}$

The gradient correction is similar to SVRG

Relationship to 2nd Order Gradient Sampling

Each $f^{(\ell)}(w)$ takes m out of n sample from $f(w) = n^{-1}\sum_i f_i(w)$.

$$\tilde{f}^{(\ell)}(w) = f^{(\ell)}(w) - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w$$

Expand $f^{(\ell)}(w)$ around $\tilde{w}$, we have

$$\begin{aligned}\tilde{f}^{(\ell)}(w) \approx & f^{(\ell)}(\tilde{w}) + \nabla f^{(\ell)}(\tilde{w})^\top (w-\tilde{w}) + \frac{1}{2}(w-\tilde{w})^\top \nabla^2 f^{(\ell)}(\tilde{w})(w-\tilde{w}) \\ & - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w.\end{aligned}$$

Relationship to 2nd Order Gradient Sampling

Each $f^{(\ell)}(w)$ takes m out of n sample from $f(w) = n^{-1}\sum_i f_i(w)$.

$$\tilde{f}^{(\ell)}(w) = f^{(\ell)}(w) - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w$$

Expand $f^{(\ell)}(w)$ around $\tilde{w}$, we have

$$\begin{aligned}\tilde{f}^{(\ell)}(w) \approx & f^{(\ell)}(\tilde{w}) + \nabla f^{(\ell)}(\tilde{w})^\top (w-\tilde{w}) + \frac{1}{2}(w-\tilde{w})^\top \nabla^2 f^{(\ell)}(\tilde{w})(w-\tilde{w}) \\ & - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w.\end{aligned}$$

$\min_w \tilde{f}^{(\ell)}(w)$ is equivalent to approximate minimization of

$$\min_w \left[f^{(\ell)}(\tilde{w}) + \nabla f(\tilde{w})^\top (w-\tilde{w}) + \frac{1}{2} \underbrace{(w-\tilde{w})^\top \nabla^2 f^{(\ell)}(\tilde{w})(w-\tilde{w})}_{\text{2nd order gradient sampling from } \nabla^2 f(\tilde{w})} \right]$$

Relationship to 2nd Order Gradient Sampling

Each $f^{(\ell)}(w)$ takes m out of n sample from $f(w) = n^{-1}\sum_i f_i(w)$.

$$\tilde{f}^{(\ell)}(w) = f^{(\ell)}(w) - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w$$

Expand $f^{(\ell)}(w)$ around $\tilde{w}$, we have

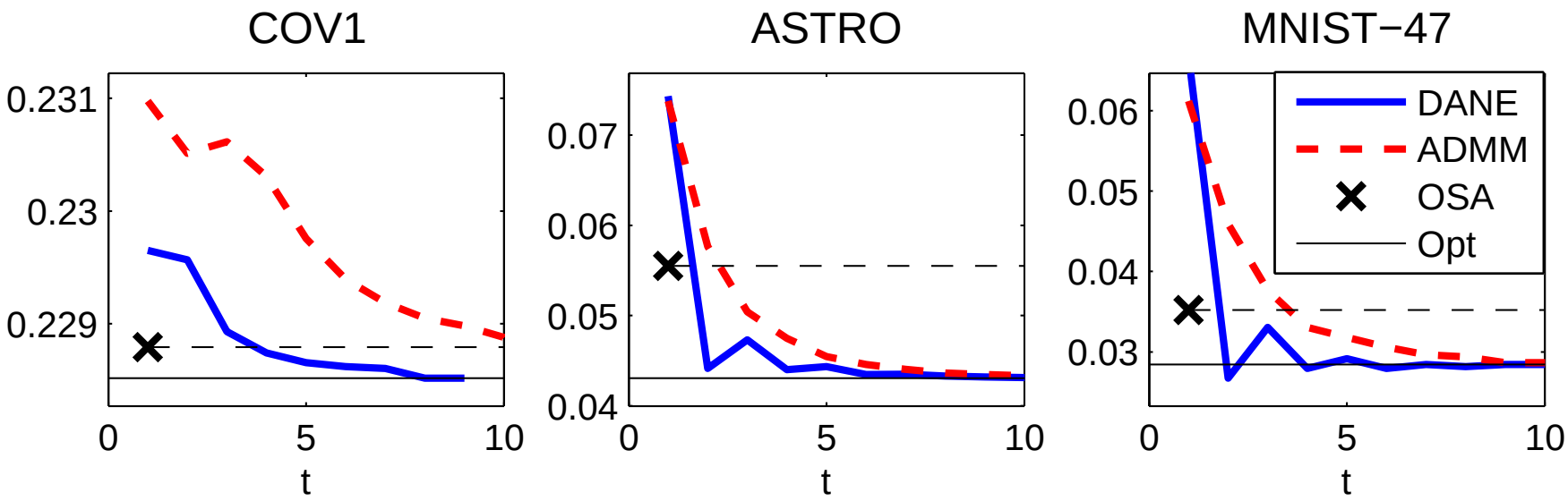
$$\begin{aligned}\tilde{f}^{(\ell)}(w) \approx & f^{(\ell)}(\tilde{w}) + \nabla f^{(\ell)}(\tilde{w})^\top (w-\tilde{w}) + \frac{1}{2}(w-\tilde{w})^\top \nabla^2 f^{(\ell)}(\tilde{w})(w-\tilde{w}) \\ & - (\nabla f^{(\ell)}(\tilde{w}) - \nabla f(\tilde{w}))^\top w.\end{aligned}$$

$\min_w \tilde{f}^{(\ell)}(w)$ is equivalent to approximate minimization of

$$\min_w \left[f^{(\ell)}(\tilde{w}) + \nabla f(\tilde{w})^\top (w-\tilde{w}) + \frac{1}{2} \underbrace{(w-\tilde{w})^\top \nabla^2 f^{(\ell)}(\tilde{w})(w-\tilde{w})}_{\text{2nd order gradient sampling from } \nabla^2 f(\tilde{w})} \right]$$

Lead to fast convergence

Comparisons



Summary

- Optimization in machine learning: sum over data structure
  - suitable for statistical sampling
- Traditional methods: gradient based batch algorithms
  - do not take advantage of special structure
- Recent progress: stochastic optimization (focusing on my own work)

Summary

- Optimization in machine learning: sum over data structure
  - suitable for statistical sampling
- Traditional methods: gradient based batch algorithms
  - do not take advantage of special structure
- Recent progress: stochastic optimization (focusing on my own work)
- Sampling of 1st order gradient
  - variance reduction leads to fast linear convergence rate
- Sampling of 2nd order gradient
  - leads to fast distributed algorithm

References

- Rie Johnson and TZ. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction, NIPS 2013.
- Lin Xiao and TZ. A Proximal Stochastic Gradient Method with Progressive Variance Reduction, Tech Report arXiv:1403.4699, March 2014.
- Shai Shalev-Shwartz and TZ. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization, JMLR 14:567-599, 2013.
- Shai Shalev-Shwartz and TZ. Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization, Tech Report arXiv:1309.2375, Sep 2013.
- Ohad Shamir and Nathan Srebro and Tong Zhang. Communication-Efficient Distributed Optimization using an Approximate Newton-type Method, ICML 2014.

References

- Rie Johnson and TZ. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction, NIPS 2013.
- Lin Xiao and TZ. A Proximal Stochastic Gradient Method with Progressive Variance Reduction, Tech Report arXiv:1403.4699, March 2014.
- Shai Shalev-Shwartz and TZ. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization, JMLR 14:567-599, 2013.
- Shai Shalev-Shwartz and TZ. Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization, Tech Report arXiv:1309.2375, Sep 2013.
- Ohad Shamir and Nathan Srebro and Tong Zhang. Communication-Efficient Distributed Optimization using an Approximate Newton-type Method, ICML 2014.
- My friend Lin Xiao has made further improves with collaborators in multiple fronts on this thread of big optimization research ...