Stochastic Optimization Techniques for Big Data Machine Learning

Tong Zhang

Rutgers University & Baidu Inc.

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: 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: large n

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: large n

Assumptions on loss function

- λ-strong convexity:

$$f(w') \geq \underbrace{f(w) + \nabla f(w)^\top (w' - w) + \frac{\lambda}{2}\|w' - w\|_2^2}_{\text{quadratic lower bound}}$$

- L-smoothness:

$$f_i(w') \leq \underbrace{f_i(w) + \nabla f_i(w)^\top (w' - w) + \frac{L}{2}\|w' - w\|_2^2}_{\text{quadratic upper bound}}$$

Example: Computational Advertising

- 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 Advertising

- 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 Advertising

- 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?

Optimization Problem: Communication Complexity

From simple to complex

- Single machine single-core
  can employ sequential algorithms

Optimization Problem: Communication Complexity

From simple to complex

- Single machine single-core
  can employ sequential algorithms
- Single machine multi-core
  relatively cheap communication

Optimization Problem: Communication Complexity

From simple to complex

- Single machine single-core
  can employ sequential algorithms
- Single machine multi-core
  relatively cheap communication
- Multi-machine (synchronous)
  expensive communication

Optimization Problem: Communication Complexity

From simple to complex

- Single machine single-core
  can employ sequential algorithms
- Single machine multi-core
  relatively cheap communication
- Multi-machine (synchronous)
  expensive communication
- Multi-machine (asynchronous)
  break synchronization to reduce communication

Optimization Problem: Communication Complexity

From simple to complex

- Single machine single-core
  can employ sequential algorithms
- Single machine multi-core
  relatively cheap communication
- Multi-machine (synchronous)
  expensive communication
- Multi-machine (asynchronous)
  break synchronization to reduce communication

We want to solve simple problem well first, then more complex ones.

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

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?

- 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:

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

SGD versus GD

SGD:

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

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

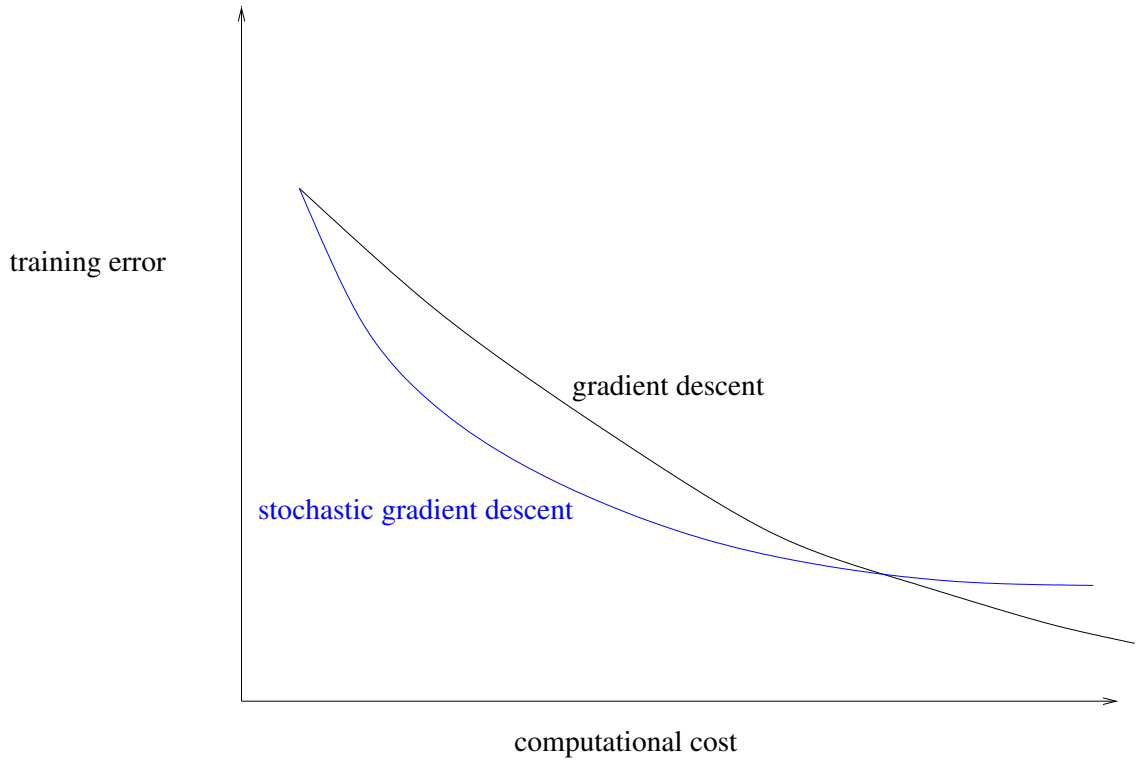
GD:

- slower computation per step
- Linear convergence:

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

Overall: sgd is fast in the beginning but slow asymptotically

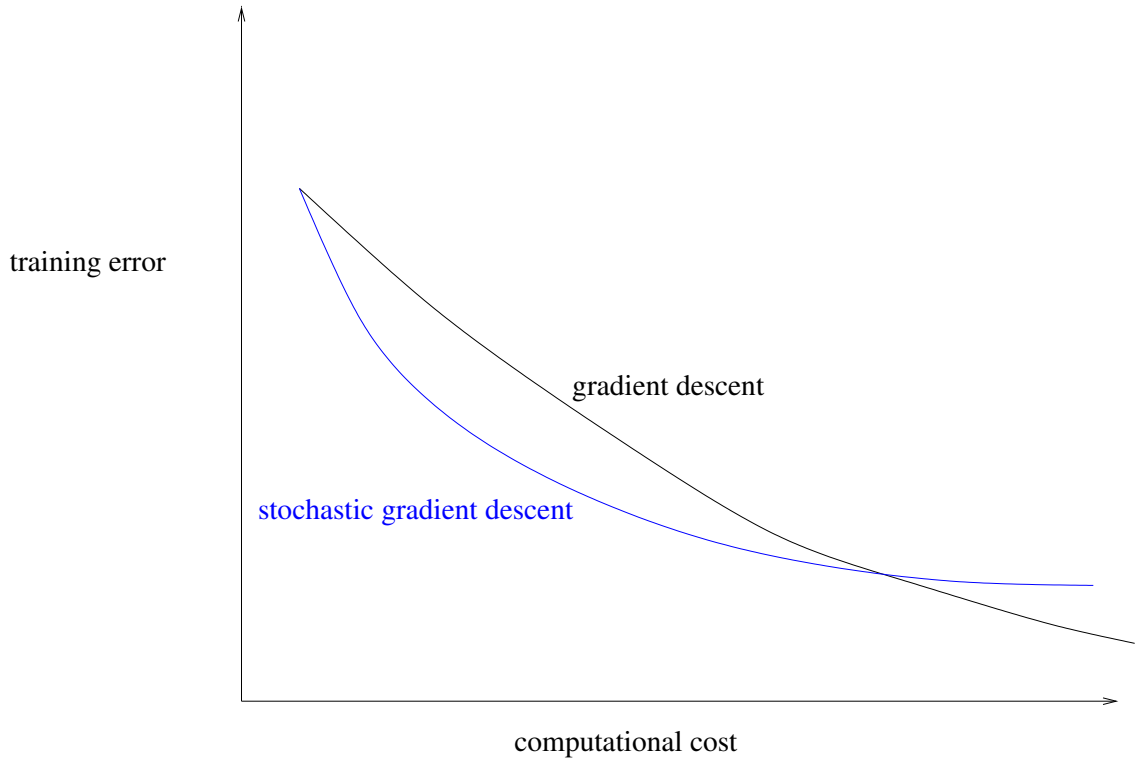
SGD versus GD



One strategy:

- use sgd first to train
- after a while switch to batch methods such as LBFGS.

SGD versus GD



One strategy:

- use sgd first to train
- after a while switch to batch methods such as LBFGS.

However, one can do better

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

Improving SGD using Variance Reduction

The idea leads to modern stochastic algorithms for big data machine learning with fast convergence rate

Improving SGD using Variance Reduction

The idea leads to modern stochastic algorithms for big data machine learning with fast convergence rate

- Collins et al (2008): For special problems, with a relatively complicated algorithm (Exponentiated Gradient on dual)
- Le Roux, Schmidt, Bach (NIPS 2012): A variant of SGD called SAG (stochastic average gradient) and later SAGA (Defazio, Bach, Lacoste-Julien, NIPS 2014)
- Johnson and Z (NIPS 2013): SVRG (Stochastic variance reduced gradient)
- Shalev-Schwartz and Z (JMLR 2013): SDCA (Stochastic Dual Coordinate Ascent) , and later a variant with Zheng Qu and Peter Richtarik

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

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

Statistical Thinking: variance reduction techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?

Statistical Thinking: variance reduction techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?
- Control variates (leads to SVRG).
  - find $\tilde{\mathbf{g}}_i \approx \mathbf{g}_i$
  - use estimator

$$\mathbf{g}_i' := \mathbf{g}_i - \tilde{\mathbf{g}}_i + \mathbf{E}\, \tilde{\mathbf{g}}_i.$$

Statistical Thinking: variance reduction techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?
- Control variates (leads to SVRG).
  - find $\tilde{\mathbf{g}}_i \approx \mathbf{g}_i$
  - use estimator
$$\mathbf{g}_i' := \mathbf{g}_i - \tilde{\mathbf{g}}_i + \mathbf{E}\,\tilde{\mathbf{g}}_i.$$
- Importance sampling (Zhao and Zhang ICML 2014)
  - sample $\mathbf{g}_i$ proportional to ρ_i ($\mathbf{E}\rho_i = 1$)
  - use estimator $\mathbf{g}_i/\rho_i$

Statistical Thinking: variance reduction techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?
- Control variates (leads to SVRG).
  - find $\tilde{\mathbf{g}}_i \approx \mathbf{g}_i$
  - use estimator
$$\mathbf{g}_i' := \mathbf{g}_i - \tilde{\mathbf{g}}_i + \mathbf{E}\,\tilde{\mathbf{g}}_i.$$
- Importance sampling (Zhao and Zhang ICML 2014)
  - sample $\mathbf{g}_i$ proportional to ρ_i ($\mathbf{E}\rho_i = 1$)
  - use estimator $\mathbf{g}_i/\rho_i$
- Stratified sampling (Zhao and Zhang)

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.

SVRG Algorithm

Procedure SVRG

Parameters update frequency m and learning rate η
Initialize $\tilde{w}_0$
Iterate: for $s = 1, 2, \ldots$
  $\tilde{w} = \tilde{w}_{s-1}$
  $\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla \psi_i(\tilde{w})$
  $w_0 = \tilde{w}$
  Iterate: for $t = 1, 2, \ldots, m$
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
      $w_t = w_{t-1} - \eta(\nabla \psi_{i_t}(w_{t-1}) - \nabla \psi_{i_t}(\tilde{w}) + \tilde{\mu})$
  end
  Set $\tilde{w}_s = w_m$
end

SVRG v.s. Batch Gradient Descent: fast convergence

Number of examples needed to achieve ϵ accuracy:

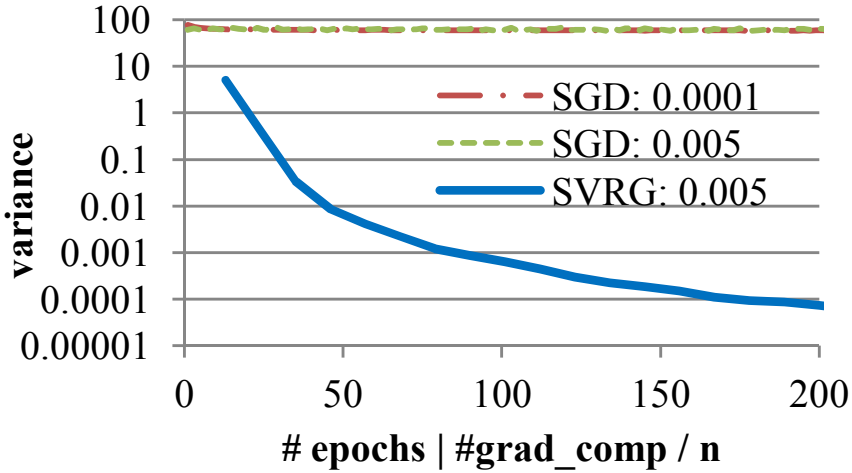
- Batch GD: $\tilde{O}(n \cdot L/\lambda \log(1/\epsilon))$
- SVRG: $\tilde{O}((n + L/\lambda) \log(1/\epsilon))$

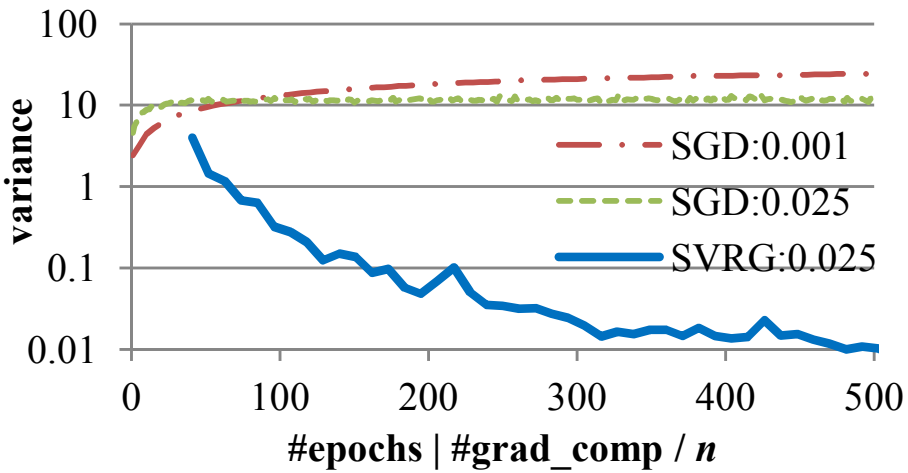
Assume L-smooth loss f_i and λ strongly convex objective function.

SVRG has fast convergence — condition number effectively reduced

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

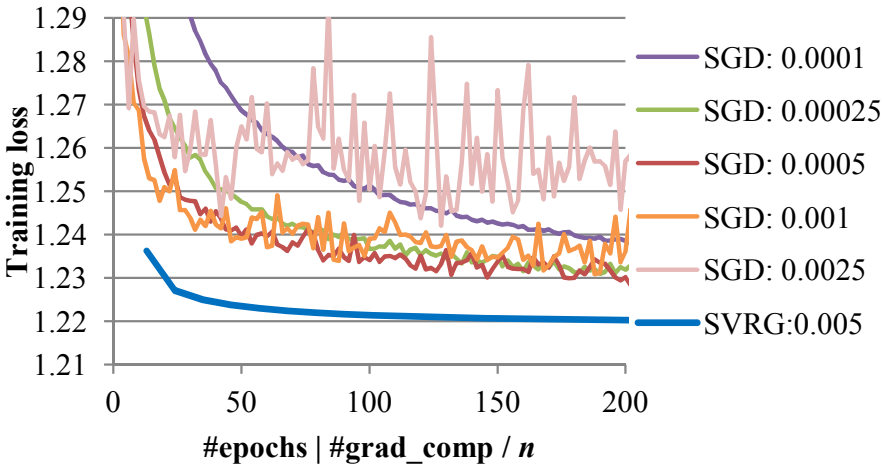
SVRG: variance

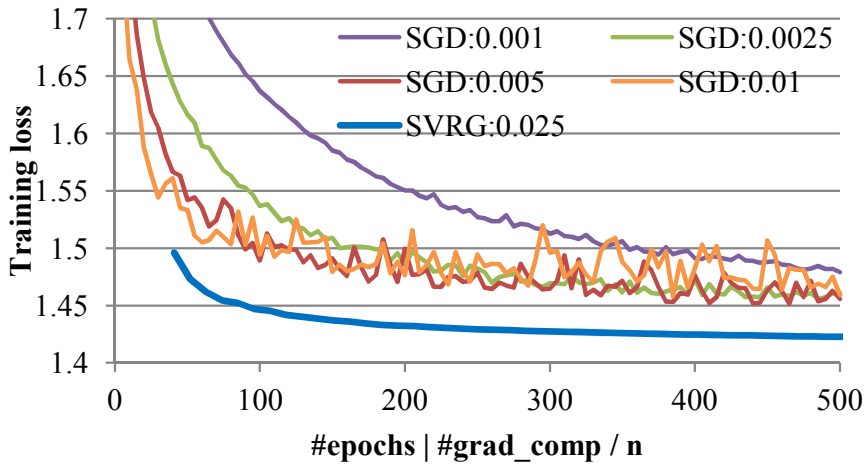




- Convex case (left): least squares on MNIST;
- Nonconvex case (right): neural nets on CIFAR-10.
- The numbers in the legends are learning rate

SVRG: convergence





- Convex case (left): least squares on MNIST;
- Nonconvex case (right): neural nets on CIFAR-10.
- The numbers in the legends are learning rate

Variance Reduction using Importance Sampling (combined with SVRG)

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

L_i: smoothness param of $f_i(w)$; λ: strong convexity param of $f(w)$

Number of examples needed to achieve ϵ accuracy:

- With uniform sampling:

$$\tilde{O}((n + L/\lambda)\log(1/\epsilon)),$$

  where $L = \max_i L_i$

- With importance sampling:

$$\tilde{O}((n + \bar{L}/\lambda)\log(1/\epsilon)),$$

  where $\bar{L} = n^{-1}\sum_{i=1}^{n} L_i$

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

Motivation

Solve

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

SGD with variance reduction via SVRG:

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

Motivation

Solve

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

SGD with variance reduction via SVRG:

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

Compute full gradient $\nabla f(\tilde{w})$ periodically at an intermediate $\tilde{w}$

Motivation

Solve

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

SGD with variance reduction via SVRG:

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

Compute full gradient $\nabla f(\tilde{w})$ periodically at an intermediate $\tilde{w}$

How to avoid computing $\nabla f(\tilde{w})$?
Answer: keeping previously calculated gradients.

Stochastic Average Gradient ameliore: SAGA

Initialize: $\tilde{\mathbf{g}}_i = \nabla f_i(w_0)$ and $\tilde{\mathbf{g}} = \frac{1}{n}\sum_{j=1}^n \tilde{\mathbf{g}}_j$

SAGA update rule: randomly select i, and

$$\begin{aligned} w_t &= w_{t-1} - \eta_t[\nabla f_i(w_{t-1}) - \tilde{\mathbf{g}}_i + \tilde{\mathbf{g}}] \\ \tilde{\mathbf{g}} &= \tilde{\mathbf{g}} + (\nabla f_i(w_{t-1}) - \tilde{\mathbf{g}}_i)/n \\ \tilde{\mathbf{g}}_i &= \nabla f_i(w_{t-1}) \end{aligned}$$

Equivalent to:

$$w_t = w_{t-1} - \eta_t \underbrace{[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_i) + \frac{1}{n}\sum_{j=1}^n \nabla f_j(\tilde{w}_j)]}_{\text{small variance}} \qquad \tilde{w}_i = w_{t-1}.$$

Compare to SVRG:

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

Variance Reduction

The gradient estimator of SAGA is unbiased:

$$\mathbf{E}\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_i) + \frac{1}{n}\sum_{j=1}^{n}\nabla f_j(\tilde{w}_j)\right] = \nabla f(w_{t-1}).$$

Since $\tilde{w}_i \to w_*$, we have

$$\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_i) + \frac{1}{n}\sum_{j=1}^{n}\nabla f_j(\tilde{w}_j)\right] \to 0.$$

Therefore variance of the gradient estimator goes to zero.

Theory of SAGA

Similar to SVRG, we have fast convergence for SAGA.
Number of examples needed to achieve ϵ accuracy:

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

Assume L-smooth loss f_i and λ strongly convex objective function.

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: 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:

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

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:

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

Goal: solve regularized loss minimization problems as fast as we can.

- solution: *proximal Stochastic Dual Coordinate Ascent* (Prox-SDCA).
- can show: fast convergence of SDCA.

Loss Minimization with L_2 Regularization

$$\min_w P(w) := \left[\frac{1}{n} \sum_{i=1}^{n} \phi_i(w^\top x_i) + \frac{\lambda}{2} \|w\|^2 \right].$$

Loss Minimization with L_2 Regularization

$$\min_w P(w) := \left[\frac{1}{n}\sum_{i=1}^{n}\phi_i(w^\top x_i) + \frac{\lambda}{2}\|w\|^2\right].$$

Examples:

	$\phi_i(z)$	Lipschitz	smooth
SVM	$\max\{0, 1 - y_i z\}$	✓	✗
Logistic regression	$\log(1 + \exp(-y_i z))$	✓	✓
Abs-loss regression	$\lvert z - y_i\rvert$	✓	✗
Square-loss regression	$(z - y_i)^2$	✗	✓

Dual Formulation

Primal problem:

$$w_* = \arg\min_w P(w) := \left[\frac{1}{n}\sum_{i=1}^n \phi_i(w^\top x_i) + \frac{\lambda}{2}\|w\|^2\right]$$

Dual problem:

$$\alpha_* = \max_{\alpha\in\mathbb{R}^n} D(\alpha) := \left[\frac{1}{n}\sum_{i=1}^n -\phi_i^*(-\alpha_i) - \frac{\lambda}{2}\left\|\frac{1}{\lambda n}\sum_{i=1}^n \alpha_i x_i\right\|^2\right],$$

and the convex conjugate (dual) is defined as:

$$\phi_i^*(a) = \sup_z(az - \phi_i(z)).$$

Relationship of Primal and Dual Solutions

Weak duality: $P(w) \geq D(\alpha)$ for all w and α
Strong duality: $P(w_*) = D(\alpha_*)$ with the relationship

$$w_* = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_{*,i} \cdot x_i, \quad \alpha_{*\,i} = -\phi_i'(w_*^\top x_i).$$

Relationship of Primal and Dual Solutions

Weak duality: $P(w) \geq D(\alpha)$ for all w and α
Strong duality: $P(w_*) = D(\alpha_*)$ with the relationship

$$w_* = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_{*,i} \cdot x_i, \quad \alpha_{*\,i} = -\phi_i'(w_*^\top x_i).$$

Duality gap: for any w and α:

$$\underbrace{P(w) - D(\alpha)}_{\text{duality gap}} \geq \underbrace{P(w) - P(w_*)}_{\text{primal sub-optimality}}.$$

Example: Linear Support Vector Machine

- Primal formulation:

$$P(w) = \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - w^\top x_i y_i) + \frac{\lambda}{2}\|w\|_2^2$$

- Dual formulation:

$$D(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \alpha_i y_i - \frac{1}{2\lambda n^2}\left\|\sum_{i=1}^{n} \alpha_i x_i y_i\right\|_2^2, \quad \alpha_i y_i \in [0, 1].$$

- Relationship:

$$w_* = \frac{1}{\lambda n}\sum_{i=1}^{n} \alpha_{*,i} x_i$$

Dual Coordinate Ascent (DCA)

Solve the dual problem using coordinate ascent

$$\max_{\alpha \in \mathbb{R}^n} D(\alpha),$$

and keep the corresponding primal solution using the relationship

$$w = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i.$$

- DCA: At each iteration, optimize $D(\alpha)$ w.r.t. a single coordinate, while the rest of the coordinates are kept in tact.
- Stochastic Dual Coordinate Ascent (SDCA): Choose the updated coordinate uniformly at random

Dual Coordinate Ascent (DCA)

Solve the dual problem using coordinate ascent

$$\max_{\alpha \in \mathbb{R}^n} D(\alpha),$$

and keep the corresponding primal solution using the relationship

$$w = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i.$$

- DCA: At each iteration, optimize $D(\alpha)$ w.r.t. a single coordinate, while the rest of the coordinates are kept in tact.
- Stochastic Dual Coordinate Ascent (SDCA): Choose the updated coordinate uniformly at random

SMO (John Platt), Liblinear (Hsieh et al) etc implemented DCA.

SDCA vs. SGD — update rule

Stochastic Gradient Descent (SGD) update rule:

$$w^{(t+1)} = \left(1 - \frac{1}{t}\right) w^{(t)} - \frac{\phi_i'(w^{(t)\top} x_i)}{\lambda t} x_i$$

SDCA update rule:

1. $\Delta_i = \operatorname*{argmax}_{\Delta \in \mathbb{R}} D(\alpha^{(t)} + \Delta_i e_i)$
2. $w^{(t+1)} = w^{(t)} + \frac{\Delta_i}{\lambda n} x_i$

- Rather similar update rules.
- SDCA has several advantages:
  - Stopping criterion: duality gap smaller than a value
  - No need to tune learning rate

SDCA vs. SGD — update rule — Example

SVM with the hinge loss: $\phi_i(w) = \max\{0, 1 - y_i w^\top x_i\}$

SGD update rule:

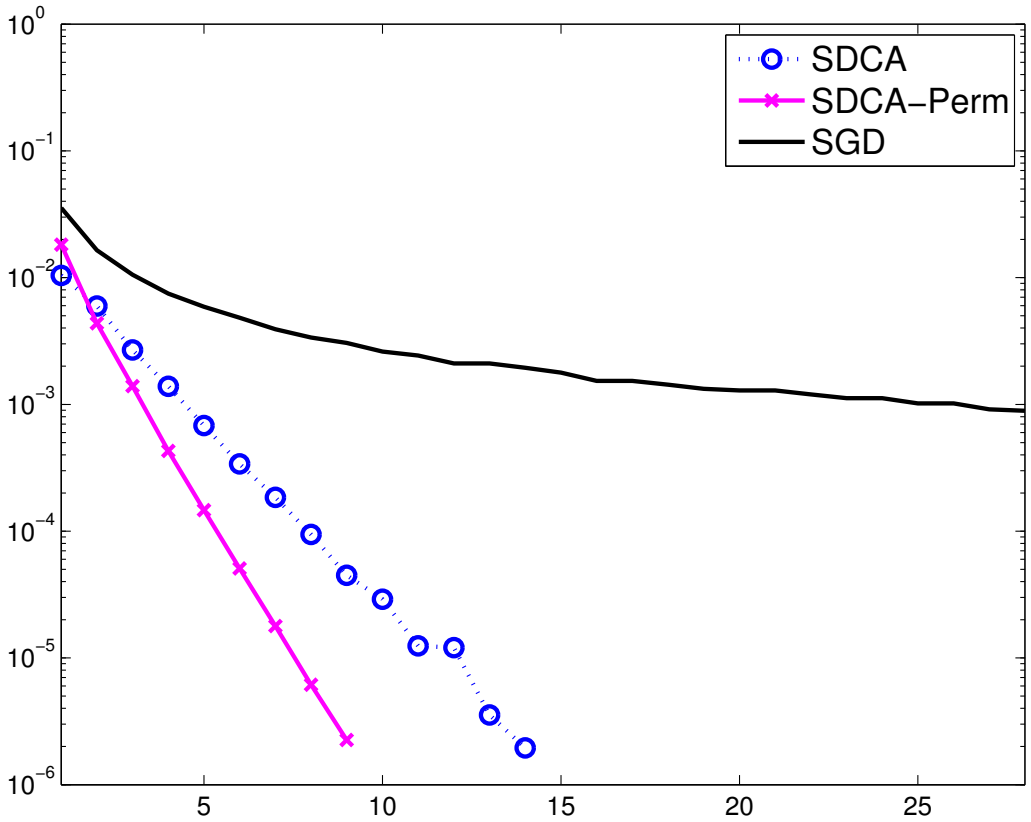
$$w^{(t+1)} = \left(1 - \tfrac{1}{t}\right) w^{(t)} - \frac{\mathbf{1}[y_i\, x_i^\top w^{(t)} < 1]}{\lambda\, t}\, x_i$$

SDCA update rule:

1. $\Delta_i = y_i \max\left(0, \min\left(1, \dfrac{1 - y_i\, x_i^\top w^{(t-1)}}{\|x_i\|_2^2/(\lambda n)} + y_i\, \alpha_i^{(t-1)}\right)\right) - \alpha_i^{(t-1)}$
1. $\alpha^{(t+1)} = \alpha^{(t)} + \Delta_i\, e_i$
2. $w^{(t+1)} = w^{(t)} + \dfrac{\Delta_i}{\lambda\, n}\, x_i$

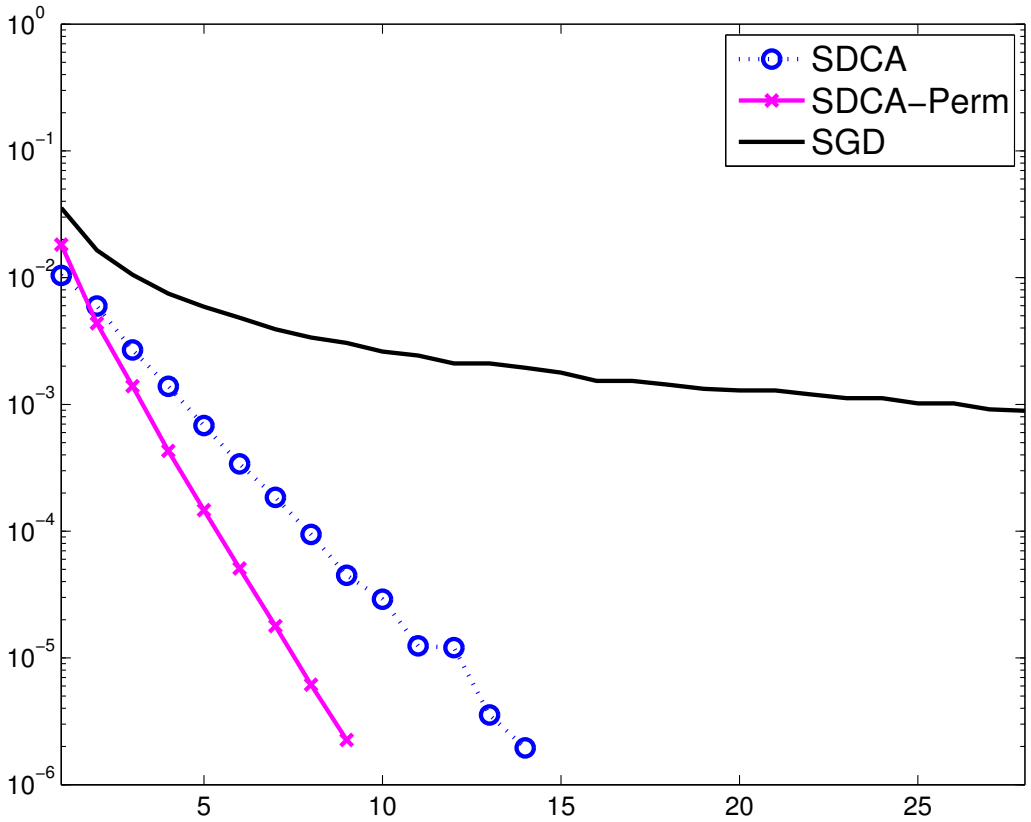
SDCA vs. SGD — experimental observations

- On CCAT dataset, $\lambda = 10^{-6}$, smoothed loss



SDCA vs. SGD — experimental observations

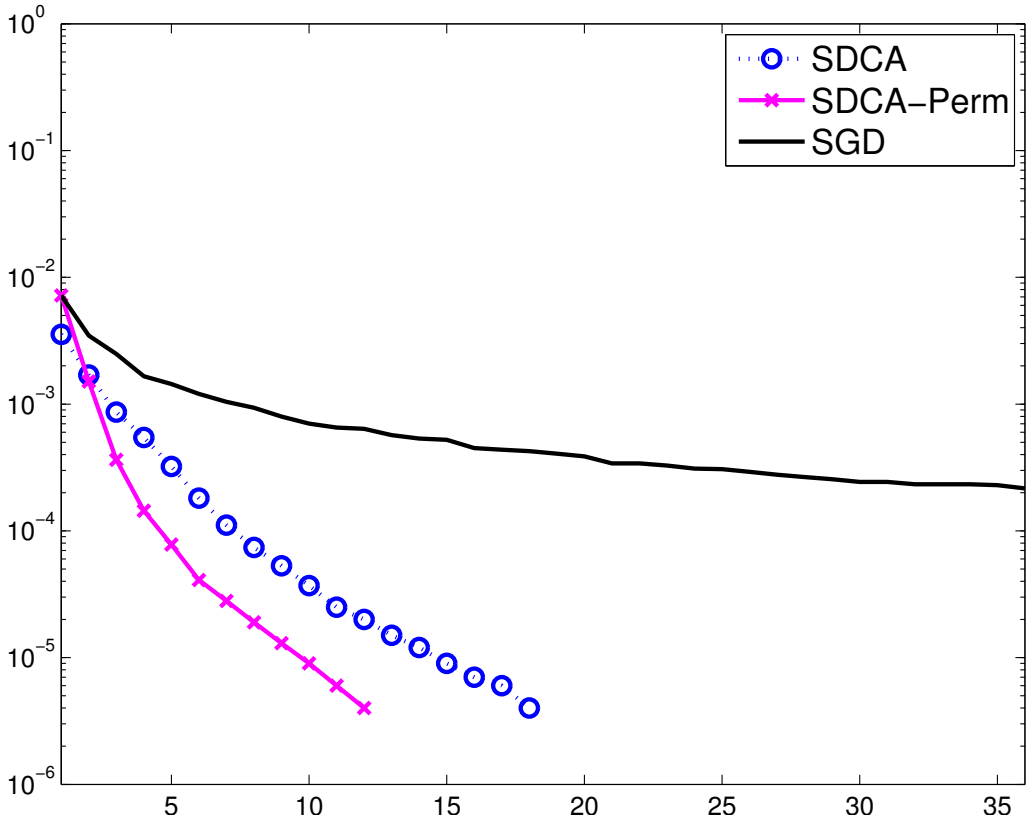
- On CCAT dataset, $\lambda = 10^{-6}$, smoothed loss



The convergence of SDCA is shockingly fast! How to explain this?

SDCA vs. SGD — experimental observations

- On CCAT dataset, $\lambda = 10^{-5}$, hinge-loss



How to understand the convergence behavior?

Derivation of SDCA I

Consider the following optimization problem

$$w_* = \arg\min_w f(w) \qquad f(w) = n^{-1} \sum_{i=1}^{n} \phi_i(w) + 0.5\lambda w^\top w.$$

The optimal condition is

$$n^{-1} \sum_{i=1}^{n} \nabla \phi_i(w_*) + \lambda w_* = 0.$$

We have dual representation:

$$w_* = \sum_{i=1}^{n} \alpha_i^* \quad \alpha_i^* = -\frac{1}{\lambda n} \nabla \phi_i(w_*)$$

Derivation of SDCA II

If we maintain a relationship: $w = \sum_{i=1}^{n} \alpha_i$, then SGD rule

$$w_t = w_{t-1} - \underbrace{\eta \nabla \phi_i(w_{t-1}) - \eta \lambda w_{t-1}}_{\text{large variance}}.$$

- property $E[w_t | w_{t-1}] = w_{t-1} - \nabla f(w)$

The dual representation of SGD rule is

$$\alpha_{t,j} = (1 - \eta\lambda)\alpha_{t-1,j} - \eta \nabla \phi_i(w_{t-1})\delta_{i,j}.$$

Derivation of SDCA III

The alternative SDCA rule is to replace $-\eta\lambda w_{t-1}$ by $-\eta\lambda n\alpha_i$: primal update is

$$w_t = w_{t-1} - \underbrace{\eta(\nabla\phi_i(w_{t-1}) + \lambda n\alpha_i)}_{\text{small variance}}.$$

and the dual update is

$$\alpha_{t,j} = \alpha_{t-1,j} - \eta(\nabla\phi_i(w_{t-1}) + \lambda n\alpha_i)\delta_{i,j}.$$

It is unbiased: $E[w_t|w_{t-1}] = w_{t-1} - \nabla f(w)$

Benefit of SDCA

Variance reduction effect: as $w \to w_*$ and $\alpha \to \alpha_*$,

$$\nabla \phi_i(w_{t-1}) + \lambda n \alpha_i \to 0,$$

thus the stochastic variance goes to zero.

Benefit of SDCA

Variance reduction effect: as $w \to w_*$ and $\alpha \to \alpha_*$,

$$\nabla \phi_i(w_{t-1}) + \lambda n \alpha_i \to 0,$$

thus the stochastic variance goes to zero.

Fast convergence rate result:

$$Ef(w_t) - f(w_*) = O(\mu^k),$$

where $\mu = 1 - O(\lambda n/(1+\lambda n))$.

- Convergence rate is fast even when $\lambda = O(1/n)$.
- Better than batch method

Fast Convergence of SDCA

The number of iterations needed to achieve ϵ accuracy

- For L-smooth loss:

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

- For non-smooth but G-Lipschitz loss (bounded gradient):

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

Fast Convergence of SDCA

The number of iterations needed to achieve ϵ accuracy

- For L-smooth loss:

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

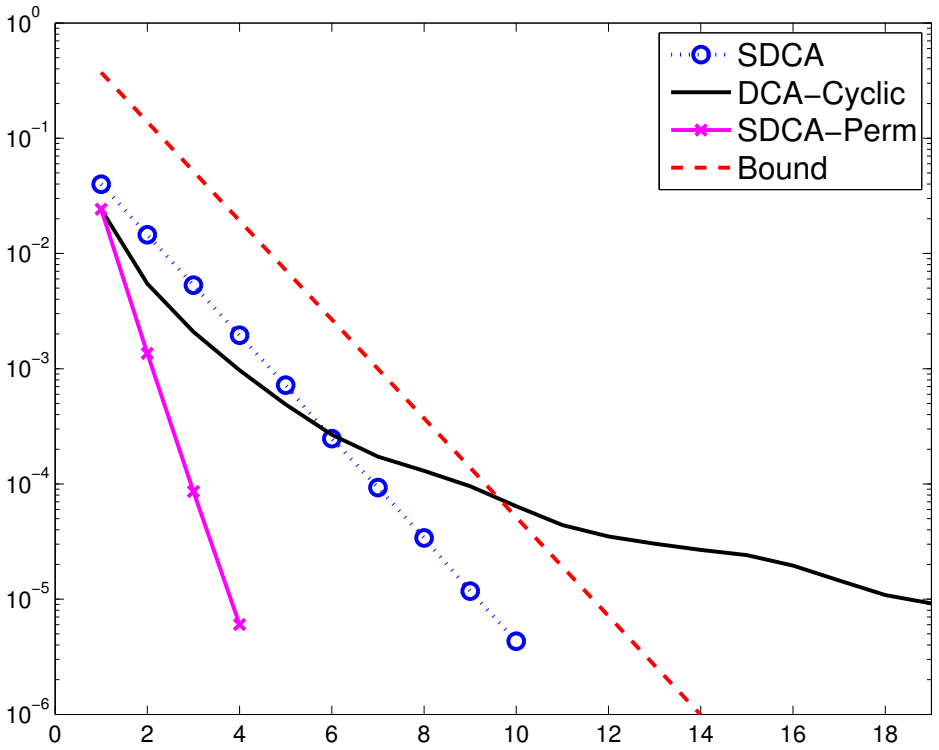
- For non-smooth but G-Lipschitz loss (bounded gradient):

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

Similar to that of SVRG; and effective when n is large

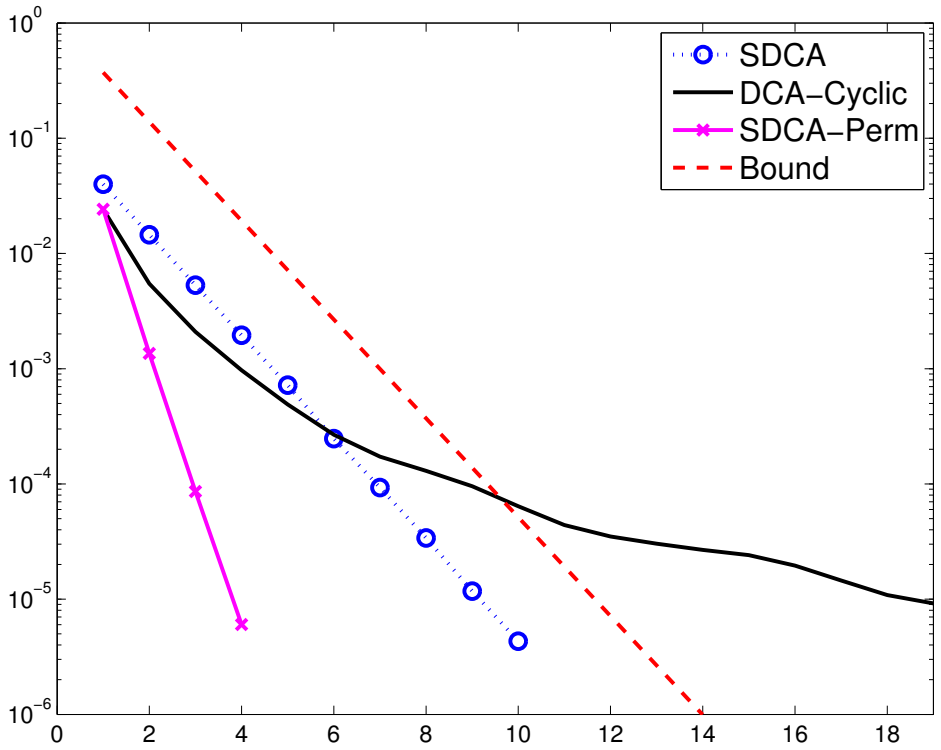
SDCA vs. DCA — Randomization is Crucial!

- On CCAT dataset, $\lambda = 10^{-4}$, smoothed hinge-loss



SDCA vs. DCA — Randomization is Crucial!

- On CCAT dataset, $\lambda = 10^{-4}$, smoothed hinge-loss



Randomization is crucial!

Proximal SDCA for General Regularizer

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

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

Prox-SDCA: extension of SDCA for arbitrarily strongly convex $g(w)$.

Prox-SDCA

Dual:

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

Assume $g(w)$ is strongly convex in norm $\|\cdot\|_P$ with dual norm $\|\cdot\|_D$.

Prox-SDCA

Dual:

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

Assume $g(w)$ is strongly convex in norm $\|\cdot\|_P$ with dual norm $\|\cdot\|_D$. For each α, and the corresponding v and w, define prox-dual

$$\tilde{D}_\alpha(\Delta\alpha) = \left[\frac{1}{n}\sum_{i=1}^{n} -\phi_i^*(-(\alpha_i + \Delta\alpha_i)) \right.$$
$$\left. -\lambda \left(\underbrace{g^*(v) + \nabla g^*(v)^\top \frac{1}{\lambda n}\sum_{i=1}^{n} X_i \Delta\alpha_i + \frac{1}{2}\left\| \frac{1}{\lambda n}\sum_{i=1}^{n} X_i \Delta\alpha_i \right\|_D^2}_{\text{upper bound of } g^*(\cdot)} \right) \right]$$

Prox-SDCA

Dual:

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

Assume $g(w)$ is strongly convex in norm $\|\cdot\|_P$ with dual norm $\|\cdot\|_D$. For each α, and the corresponding v and w, define prox-dual

$$\tilde{D}_\alpha(\Delta\alpha) = \left[\frac{1}{n}\sum_{i=1}^{n} -\phi_i^*(-(\alpha_i + \Delta\alpha_i)) \right.$$
$$\left. -\lambda\left(\underbrace{g^*(v) + \nabla g^*(v)^\top \frac{1}{\lambda n}\sum_{i=1}^{n} X_i \Delta\alpha_i + \frac{1}{2}\left\|\frac{1}{\lambda n}\sum_{i=1}^{n} X_i \Delta\alpha_i\right\|_D^2}_{\text{upper bound of } g^*(\cdot)}\right)\right]$$

Prox-SDCA: randomly pick i and update $\Delta\alpha_i$ by maximizing $\tilde{D}_\alpha(\cdot)$.

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

Solving L_1 with Smooth Loss

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 by prox-SDCA is $\tilde{O}(n + 1/\epsilon)$.

Solving L_1 with Smooth Loss

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 by prox-SDCA is $\tilde{O}(n + 1/\epsilon)$.

Compare to (number of examples needed to go through):

- Dual Averaging SGD (Xiao): $\tilde{O}(1/\epsilon^2)$.
- FISTA (Nesterov's batch accelerated proximal gradient): $\tilde{O}(n/\sqrt{\epsilon})$.

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

Solving L_1 with Smooth Loss

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 by prox-SDCA is $\tilde{O}(n + 1/\epsilon)$.

Compare to (number of examples needed to go through):

- Dual Averaging SGD (Xiao): $\tilde{O}(1/\epsilon^2)$.
- FISTA (Nesterov's batch accelerated proximal gradient): $\tilde{O}(n/\sqrt{\epsilon})$.

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

Can design accelerated prox-SDCA always superior to FISTA

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: 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 $\tilde{g}_t(w) = \lambda g(w) + 0.5\kappa \|w - y^{t-1}\|_2^2$ (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)

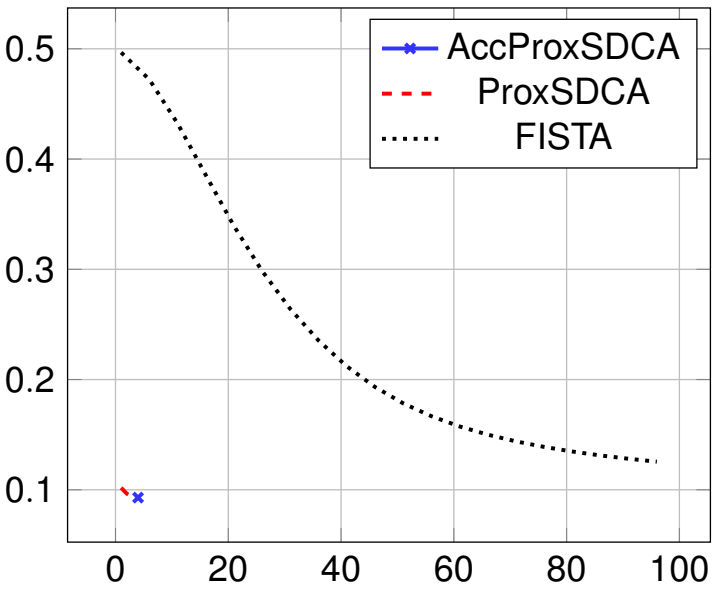
Performance Comparisons

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

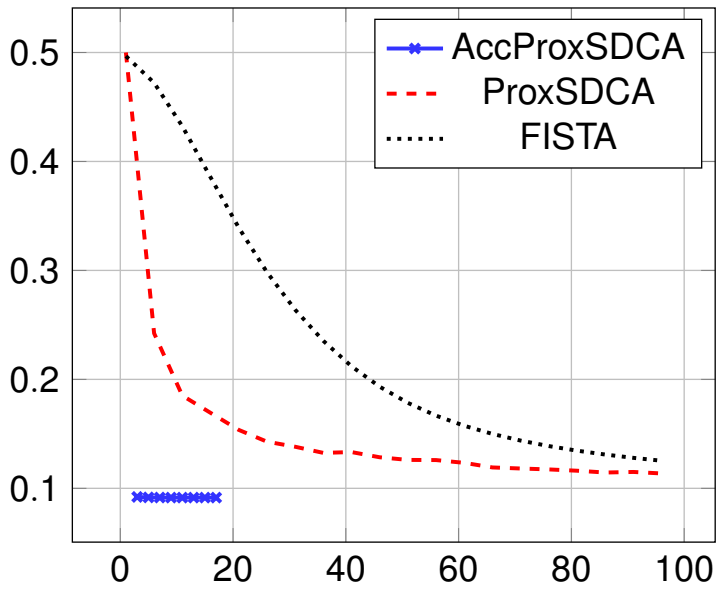
Experiments of L_1-L_2 regularization

$$\text{Smoothed hinge loss} + \frac{\lambda}{2}\|w\|_2^2 + \sigma\|w\|_1$$

on CCAT datasaet with $\sigma = 10^{-5}$



$\lambda = 10^{-7}$



$\lambda = 10^{-9}$

Additional Related Work on Acceleration

Methods achieving fast accelerated convergence comparable to Acc-Prox-SDCA

Upper bounds:

- Qihang Lin, Zhaosong Lu, Lin Xiao, An Accelerated Proximal Coordinate Gradient Method and its Application to Regularized Empirical Risk Minimization, 2014, arXiv
- Yuchen Zhang, Lin Xiao, Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization, ICML 2015 (APCG — accelerated proximal coordinate gradeint)

Matching Lower bound:

- Alekh Agarwal and Leon Bottou, A Lower Bound for the Optimization of Finite Sums, ICML 2015

Distributed Computing: Distribution Schemes

Distribute data (data parallelism)

- all machines have the same parameters
- each machine has a different set of data

Distribute features (model parallelism)

- all machines have the same data
- each machine has a different set of parameters

Distribute data and features (data & model parallelism)

- each machine has a different set of data
- each machine has a different set of parameters

Main Issues in Distributed Large Scale Learning

System Design and Network Communication

- data parallelism: need to transfer a reasonable size chunk of data each time (mini batch)
- model parallelism: distributed parameter vector

Main Issues in Distributed Large Scale Learning

System Design and Network Communication

- data parallelism: need to transfer a reasonable size chunk of data each time (mini batch)
- model parallelism: distributed parameter vector

Model Update Strategy

- synchronous
- asynchronous

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

MiniBatch

Vanilla SDCA (or SGD) is difficult to parallelize
Solution: use minibatch (thousands to hundreds of thousands)

MiniBatch

Vanilla SDCA (or SGD) is difficult to parallelize
Solution: use minibatch (thousands to hundreds of thousands)

Problem: simple minibatch implementation slows down convergence

- limited gain for using parallel computing

MiniBatch

Vanilla SDCA (or SGD) is difficult to parallelize
Solution: use minibatch (thousands to hundreds of thousands)

Problem: simple minibatch implementation slows down convergence

- limited gain for using parallel computing

Solution:

- use Nesterov acceleration
- use second order information (e.g. approximate Newton steps)

MiniBatch SDCA with Acceleration

Parameters scalars λ, γ and $\theta \in [0,1]$; mini-batch size m

Initialize $\alpha_1^{(0)} = \cdots = \alpha_n^{(0)} = \bar{\alpha}^{(0)} = 0$, $w^{(0)} = 0$

Iterate: for $t = 1, 2, \ldots$

$u^{(t-1)} = (1-\theta) w^{(t-1)} + \theta \bar{\alpha}^{(t-1)}$

Randomly pick subset $I \subset \{1, \ldots, n\}$ of size m and update

$\alpha_i^{(t)} = (1-\theta)\alpha_i^{(t-1)} - \theta \nabla f_i(u^{(t-1)})/(\lambda n)$ for $i \in I$

$\alpha_j^{(t)} = \alpha_j^{(t-1)}$ for $j \notin I$

$\bar{\alpha}^{(t)} = \bar{\alpha}^{(t-1)} + \sum_{i \in I} (\alpha_i^{(t)} - \alpha_i^{(t-1)})$

$w^{(t)} = (1-\theta) w^{(t-1)} + \theta \bar{\alpha}^{(t)}$

end

Better than vanilla block SDCA, and allow large batch.

Theory

Generally when minibatch size m increases, it is easier to parallelize, but convergence slows down

Theorem

If $\theta \leq \frac{1}{4} \min \left\{ 1 \ , \ \sqrt{\frac{\gamma \lambda n}{m}} \ , \ \gamma \lambda n \ , \ \frac{(\gamma \lambda n)^{2/3}}{m^{1/3}} \right\}$ *then after performing*

$$t \geq \frac{n/m}{\theta} \log \left(\frac{m \Delta P(x^{(0)}) + n \Delta D(\alpha^{(0)})}{m \epsilon} \right)$$

iterations, we have that $\mathcal{E}[P(x^{(t)}) - D(\alpha^{(t)})] \leq \epsilon$.

Interesting Cases

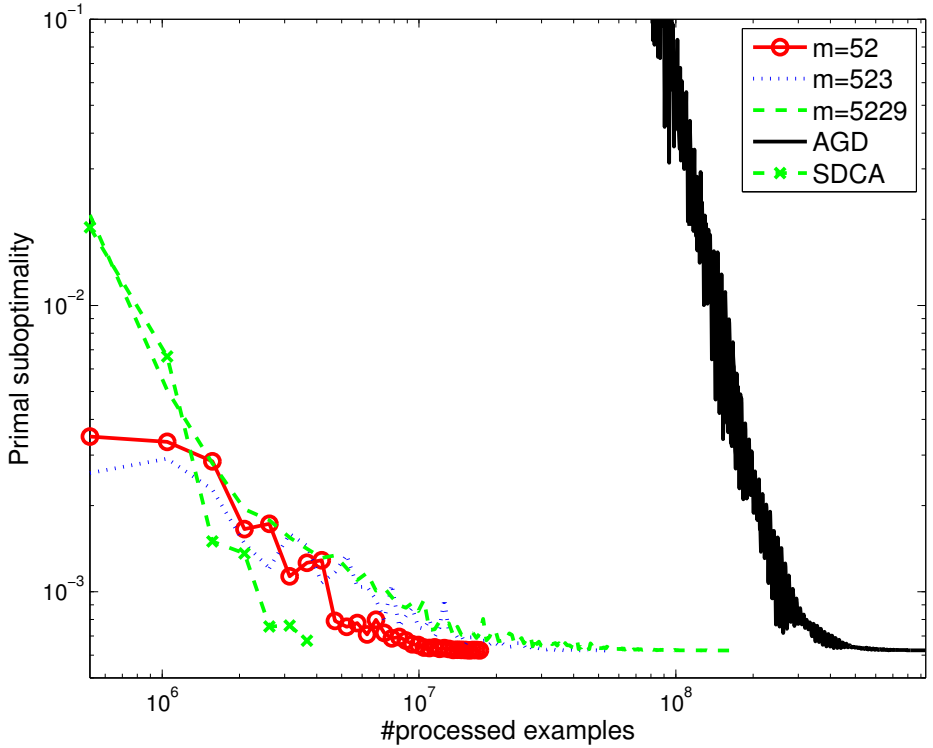
The number of **iterations** required in several interesting regimes:

Algorithm	$\gamma\lambda n = \Theta(1)$	$\gamma\lambda n = \Theta(1/m)$	$\gamma\lambda n = \Theta(m)$
SDCA	n	nm	n
ASDCA	$n/\sqrt{m}$	n	n/m
AGD	$\sqrt{n}$	$\sqrt{nm}$	$\sqrt{n/m}$

The number of **examples** processed in several interesting regimes:

Algorithm	$\gamma\lambda n = \Theta(1)$	$\gamma\lambda n = \Theta(1/m)$	$\gamma\lambda n = \Theta(m)$
SDCA	n	nm	n
ASDCA	$n\sqrt{m}$	nm	n
AGD	$n\sqrt{n}$	$n\sqrt{nm}$	$n\sqrt{n/m}$

Example



MiniBatch SDCA with acceleration can employ large minibatch size.

Outline

- Background:
  - big data optimization in machine learning: special structure

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)

Outline

- Background:
  - big data optimization in machine learning: special structure
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SAGA (Stochastic Average Gradient ameliore)
  - algorithm 3: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 4: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 5: accelerated minibatch SDCA
  - algorithm 6: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling

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 Algorithm

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 Algorithm

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 $\tilde{w}$

Lead to fast convergence: $O((1-\rho)^\ell)$ with $\rho \approx 1$

Reason: Approximate Newton Step

On each machine, we solve:

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

It can be regarded as approximate minimization of

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

Approximate Newton Step with sampled approximation of Hessian

Quadratc Loss

Newton:

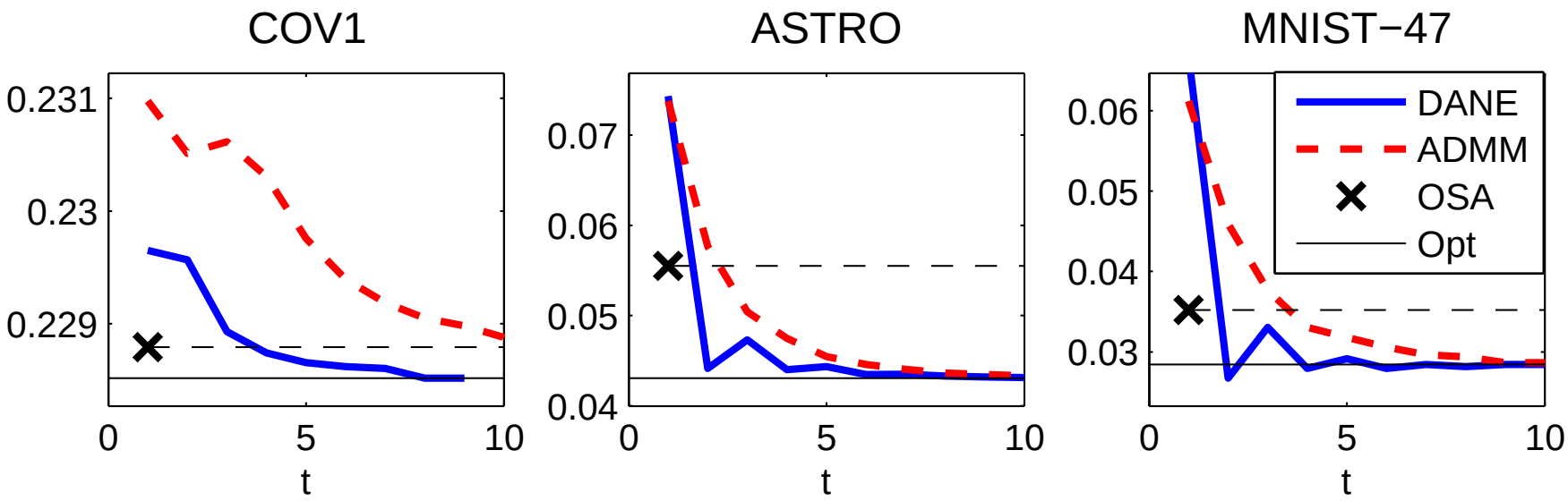
$$w^{(t)} = w^{(t-1)} - \underbrace{\left(\frac{1}{m}\sum_{\ell}\nabla^2 f^{(\ell)}\right)^{-1}}_{\text{inverse Hessian}} \nabla f(w^{(t-1)})$$

DANE:

$$w^{(t)} = w^{(t-1)} - \left(\frac{1}{m}\sum_{\ell} \underbrace{\left(\nabla^2 f^{(\ell)} + \mu I\right)^{-1}}_{\text{inverse Hessian on machine } \ell}\right) \nabla f(w^{(t-1)}),$$

where a small μ is added to regularize the Hessian.

Comparisons



Summary

- Optimization in machine learning: sum over data structure
- Traditional methods: gradient based batch algorithms
  - do not take advantage of special structure
- Recent progress: stochastic optimization with fast rate
  - take advantage of special structure: suitable for single machine

Summary

- Optimization in machine learning: sum over data structure
- Traditional methods: gradient based batch algorithms
  - do not take advantage of special structure
- Recent progress: stochastic optimization with fast rate
  - take advantage of special structure: suitable for single machine

- Distributed computing (data parallelism and synchronous update)
  - minibatch SDCA
  - DANE (batch algorithm on each machine + synchronization)

Other Developments

- Distributed large scale computing
  - algorithmic side: ADMM, Asynchronous updates (Hogwild), etc
  - system side: distributed vector computing
- Nonconvex methods
  - nonconvex regularization and loss
  - neural networks and complex models
- Closer Integration of Optimization and Statistics

References

- SVRG:
  - 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, SIAM J. Optimization, 2014.
- SAGA:

  Defazio and Bach and Lacoste-Julien, SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives, NIPS 2014.
- SDCA:
  - Shai Shalev-Shwartz and TZ. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization, JMLR 2013.
  - Shai Shalev-Shwartz and TZ. Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization, Math Programming, 2015.
  - Zheng Qu and Peter Richtarik and TZ. Randomized Dual Coordinate Ascent with Arbitrary Sampling, arXiv, 2014.

References (continued)

- mini-batch SDCA with acceleration:
  Shai Shalev-Schwartz and TZ. Accelerated Mini-Batch Stochastic Dual Coordinate Ascent, NIPS 2013.
- DANE:
  Ohad Shamir and Nathan Srebro and TZ. Communication-Efficient Distributed Optimization using an Approximate Newton-type Method, ICML 2014.