Modern 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: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: 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: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 4: Minibatch SDCA
  - algorithm 5: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling
  - other methods

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
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 4: Minibatch SDCA
  - algorithm 5: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling
  - other methods

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

Improve SGD via variance reduction:

- SGD: unbiased statistical estimator of gradient with large variance.
- Smaller variance implies faster convergence
- Idea: design other unbiased gradient estimators with small 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)
- Johnson and Z (NIPS 2013): SVRG (Stochastic variance reduced gradient)
- Shalev-Schwartz and Z (JMLR 2013): SDCA (Stochastic Dual Coordinate Ascent)

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: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 4: Minibatch SDCA
  - algorithm 5: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling
  - other methods

Stochastic Variance Reduced Gradient: Derivation

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_*).

Stochastic Variance Reduced Gradient: Derivation

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_*).
SVRG rule:

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

Compare to SGD rule:

$$w_t = w_{t-1} - \eta_t \underbrace{\nabla f_i(w_{t-1})}_{\text{large variance}}$$

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.

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: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 4: Minibatch SDCA
  - algorithm 5: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling
  - other methods

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.

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

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

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
- Single machine optimization
  - stochastic gradient (1st order) versus batch gradient: pros and cons
  - algorithm 1: SVRG (Stochastic variance reduced gradient)
  - algorithm 2: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 4: Minibatch SDCA
  - algorithm 5: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling
  - other methods

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	$\left(n+\min\{\frac{1}{\lambda}, \sqrt{\frac{n}{\lambda}}\}\right)$

Additional Related Work on Acceleration

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

- 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, 2014, arXiv

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 (parameter server)

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 (parameter server)

Model Update Strategy

- synchronous
- asynchronous

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: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 4: Minibatch SDCA
  - algorithm 5: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling
  - other methods

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 b

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 b 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 $j \notin I$

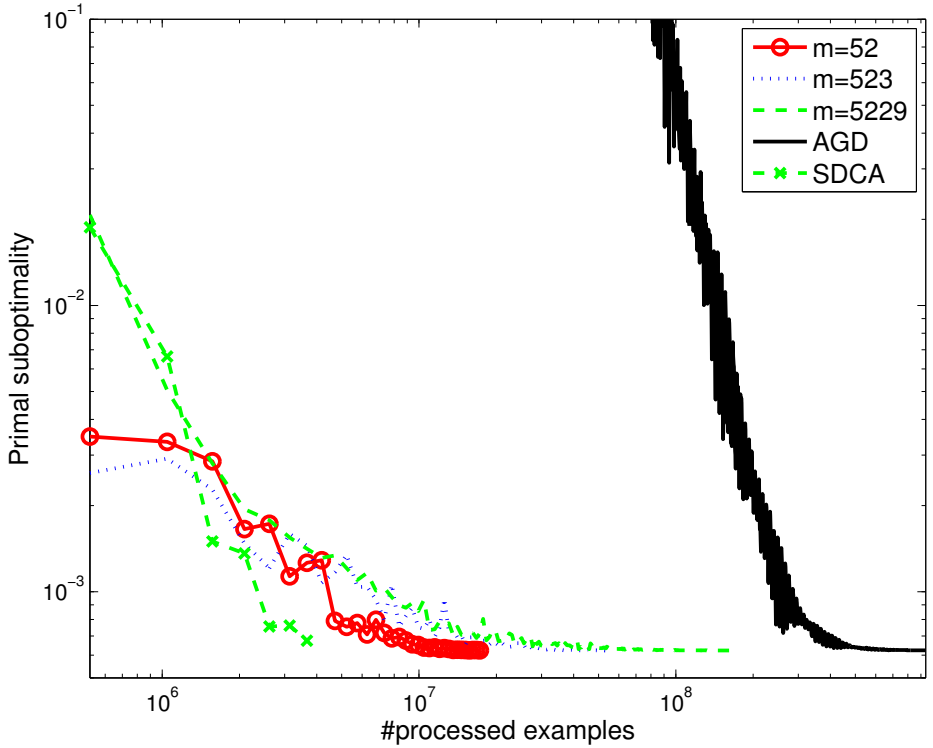
$\bar{\alpha}^{(t)} = \bar{\alpha}^{(t-1)} + \sum_{i \in I} (\alpha_i^{(t)} - \alpha_i^{(t-1)})$

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

end

Better than vanilla block SDCA, and allow large batch.

Example



MiniBatch SDCA with acceleration can employ large minibatch size.

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: SDCA (Stochastic Dual Coordinate Ascent)
  - algorithm 3: accelerated SDCA (with Nesterov acceleration)
- Distributed optimization
  - algorithm 4: Minibatch SDCA
  - algorithm 5: DANE (Distributed Approximate NEwton-type method) behaves like 2nd order stochastic sampling
  - other methods

Communication Efficient 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: l
  - $\bar{w} = m^{-1} \sum_{i=1}^{m} w^{(i)}$

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 $\tilde{w}$

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

Reason: Approximate Newton Step

On each machine, we solve:

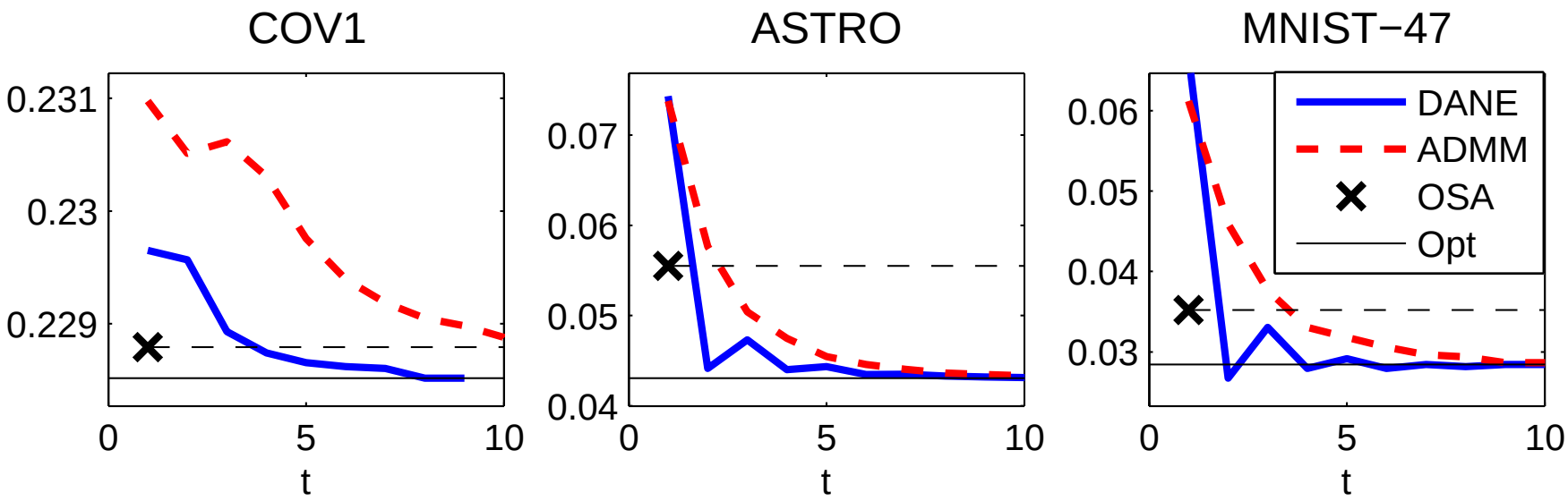
$$\min_w \tilde{f}^{(\ell)}(w).$$

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

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)

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 approaches
  - algorithmic side: ADMM, Asynchronous updates (Hogwild), etc
  - system side: distributed vector computing (parameter servers) – Baidu has industrial leading solution

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 approaches
  - algorithmic side: ADMM, Asynchronous updates (Hogwild), etc
  - system side: distributed vector computing (parameter servers) – Baidu has industrial leading solution

Fast developing field; many exciting new ideas

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, SIAM J. Opt, to appear.
- 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, Math Programming, to appear.
- Shai Shalev-Schwartz and TZ. Accelerated Mini-Batch Stochastic Dual Coordinate Ascent, NIPS 2013.
- Ohad Shamir and Nathan Srebro and TZ. Communication-Efficient Distributed Optimization using an Approximate Newton-type Method, ICML 2014.