Stochastic Optimization for Big Data Analytics: Algorithms and Libraries

Tianbao Yang[‡]
SDM 2014, Philadelphia, Pennsylvania

collaborators:
Rong Jin[†], Shenghuo Zhu[‡]
[‡]NEC Laboratories America, [†]Michigan State University

February 9, 2014

The updates are available here

http://www.cse.msu.edu/~yangtia1/sdm14-tutorial.pdf

Thanks.

Outline

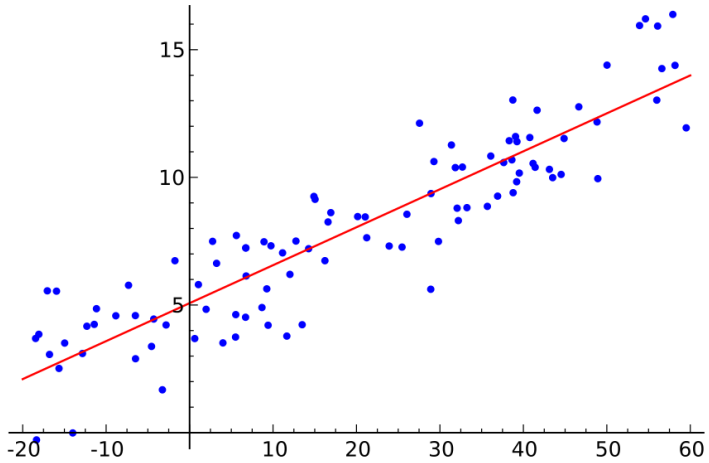
Introduction

- Machine Learning problems and Stochastic Optimization
- Coverage:
  - Classification and Regression in different forms
  - Motivation to employ STochastic OPtimization (STOP)
  - Basic Convex Optimization Knowledge

Learning as Optimization

Least Square Regression Problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

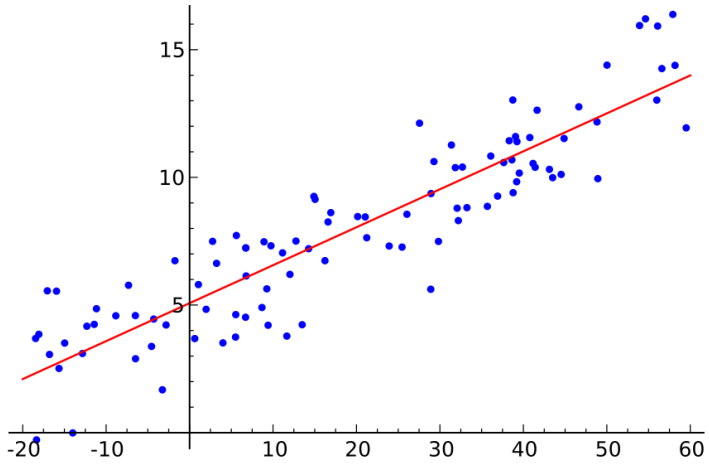


- $\mathbf{x}_i \in \mathbb{R}^d$: d-dimensional feature vector
- $y_i \in \mathbb{R}$: response variable
- $\mathbf{w} \in \mathbb{R}^d$: unknown parameters
- n: number of data points

Learning as Optimization

Least Square Regression Problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}_{\text{Empirical Loss}} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

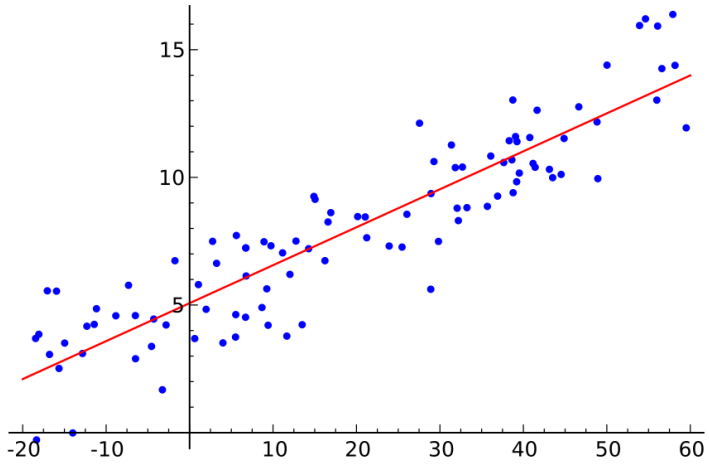


- $\mathbf{x}_i \in \mathbb{R}^d$: d-dimensional feature vector
- $y_i \in \mathbb{R}$: response variable
- $\mathbf{w} \in \mathbb{R}^d$: unknown parameters
- n: number of data points

Learning as Optimization

Least Square Regression Problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{\text{Regularization}}$$

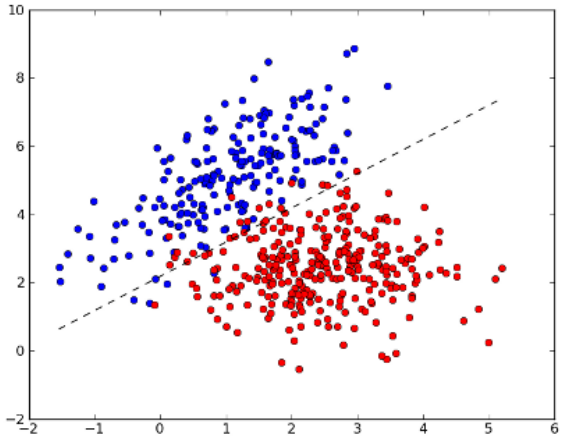


- $\mathbf{x}_i \in \mathbb{R}^d$: d-dimensional feature vector
- $y_i \in \mathbb{R}$: response variable
- $\mathbf{w} \in \mathbb{R}^d$: unknown parameters
- n: number of data points

Learning as Optimization

Classification Problems:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \mathbf{w}^\top \mathbf{x}_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

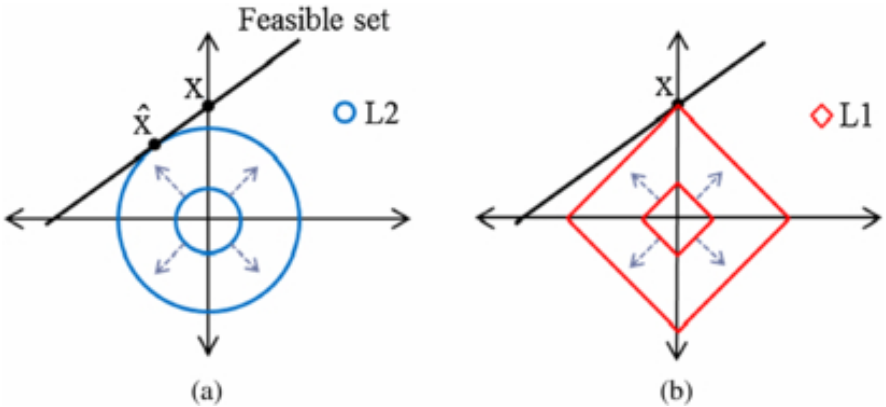


- $y_i \in \{+1, -1\}$: label
- Loss function $\ell(z)$
  1. SVMs: hinge loss $\ell(z) = \max(0, 1-z)^p$, where $p = 1, 2$
  2. Logistic Regression: $\ell(z) = \log(1 + \exp(-z))$

Learning as Optimization

Feature Selection:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \mathbf{w}^\top \mathbf{x}_i) + \lambda \|\mathbf{w}\|_1$$

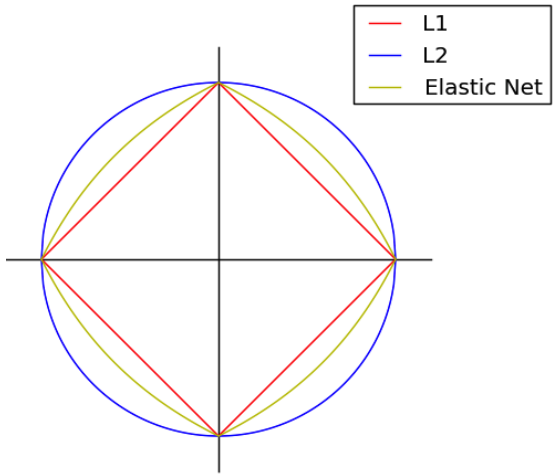


- ℓ_1 regularization $\|\mathbf{w}\|_1 = \sum_{i=1}^{d} |w_i|$
- λ controls sparsity level

Learning as Optimization

Feature Selection using Elastic Net:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \mathbf{w}^\top \mathbf{x}_i) + \lambda \left(\|\mathbf{w}\|_1 + \gamma \|\mathbf{w}\|_2^2 \right)$$

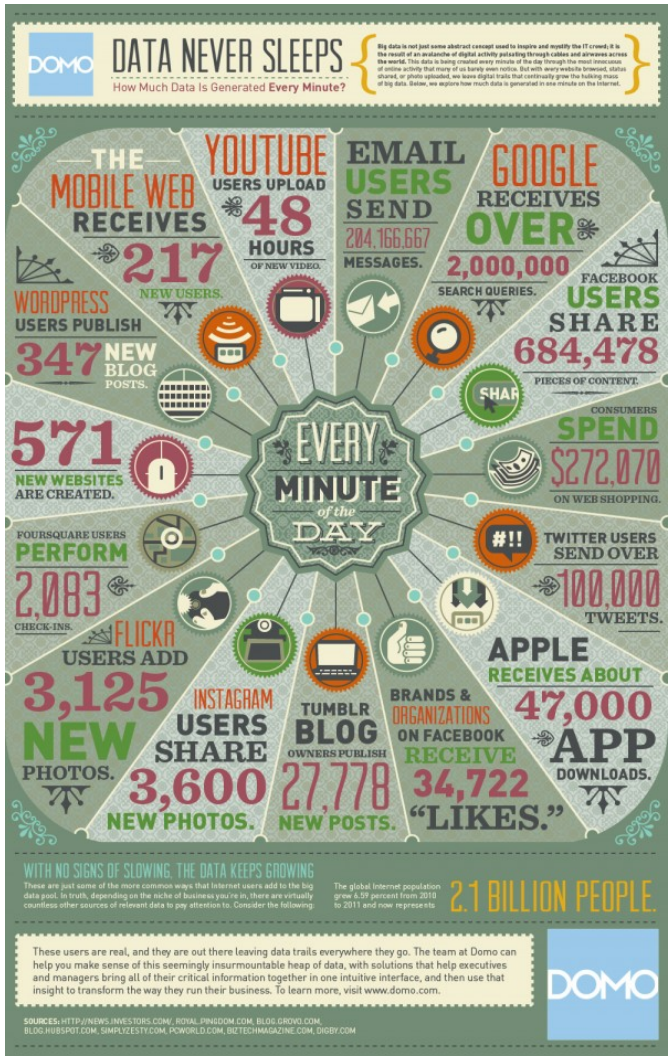


- Elastic net regularizer, more robust than ℓ_1 regularizer

Big Data Challenge

Huge amount of data generated on the internet every day

- Facebook users upload 3 million photos
- Goolge receives 3000 million queries
- GE 100 Million Hours of operational data on the World's Largest Gas Turbine Fleet
- The global internet population is 2.1 billion people



http://www.visualnews.com/2012/06/19/how-much-data-created-every-minute/

Why Learning from Big Data is Hard?

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \lambda R(\mathbf{w})}_{\text{empirical loss + regularizer}}$$

Too many data points

- Issue: can't afford go through data set many times
- Solution: Stochastic Optimization

Why Learning from Big Data is Hard?

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{\frac{1}{n}\sum_{i=1}^{n}\ell(\mathbf{w}^\top\mathbf{x}_i, y_i) + \lambda R(\mathbf{w})}_{\text{empirical loss + regularizer}}$$

High Dimensional Data

- Issue: can't afford second order optimization (e.g. Newton's method)
- Solution: first order method (i.e, gradient based method)

Why Learning from Big Data is Hard?

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \lambda R(\mathbf{w})}_{\text{empirical loss + regularizer}}$$

Data are distributed over many machines

- Issue: expensive (if not impossible) to move data
- Solution: Distributed Optimization

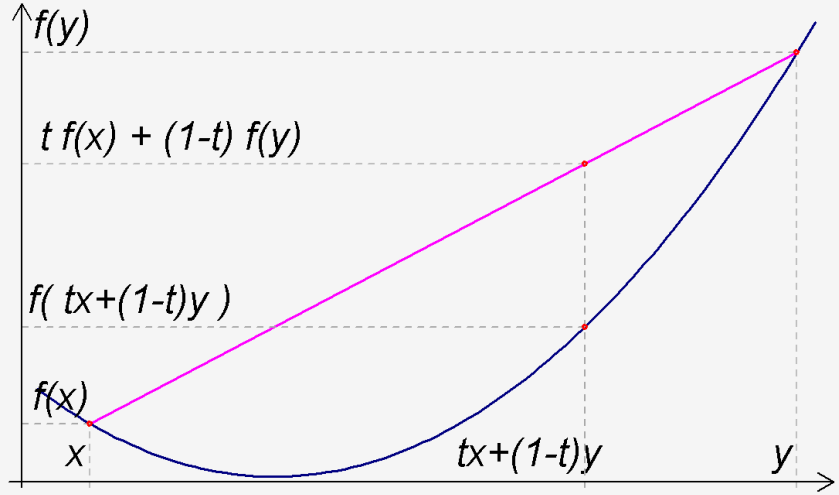
Warm-up: Vector, Norm, Inner product, Dual Norm

- bold letters $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{w} \in \mathbb{R}^d$: d-dimensional vectors
- norm $\|\mathbf{x}\|$: $\mathbb{R}^d \to \mathbb{R}_+$. e.g. ℓ_2 norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$, ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$, ℓ_∞ norm $\|\mathbf{x}\|_\infty = \max_i |x_i|$
- inner product $< \mathbf{x}, \mathbf{y} >= \mathbf{x}^\top \mathbf{y} = \sum_{i=1}^d x_i y_i$
- dual norm $\|\mathbf{x}\|_* = \max_{\|\mathbf{y}\| \leq 1} \mathbf{x}^\top \mathbf{y}$. $\|\mathbf{x}\|_2 \Longleftrightarrow \|\mathbf{x}\|_2$, $\|\mathbf{x}\|_1 \Longleftrightarrow \|\mathbf{x}\|_\infty$

Warm-up: Convex Optimization

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

- $f(\mathbf{x})$ is a convex function
- $\mathcal{X}$ is a convex domain



$$f(\alpha\mathbf{x}+(1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}), \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}, \alpha \in [0,1]$$

Warm-up: Convergence Measure

- Most optimization algorithms are iterative

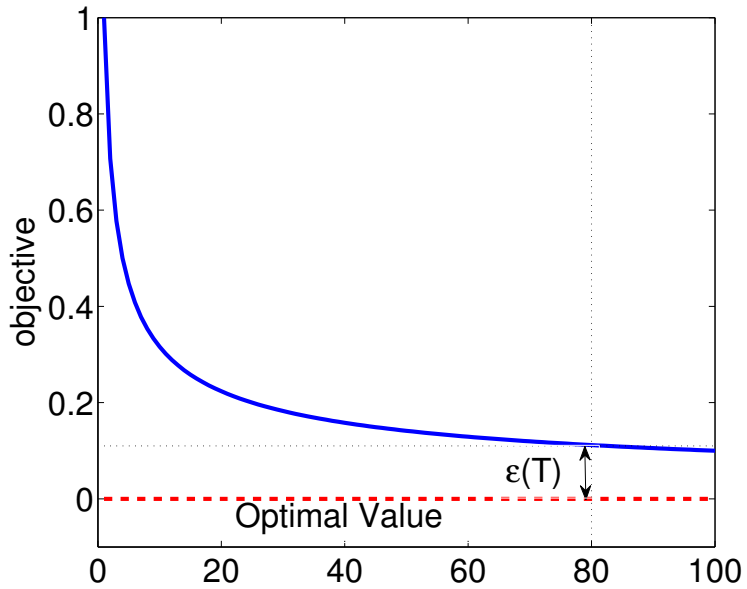
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta \mathbf{w}_t$$

- Iteration Complexity: the number of iterations $T(\epsilon)$ needed to have

$$f(\mathbf{x}_T) - \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \leq \epsilon \quad (\epsilon \ll 1)$$

- Convergence rate: after T iterations, how good is the solution

$$f(\mathbf{x}_T) - \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \leq \epsilon(T)$$



More on Convergence Measure

	Convergence Rate	Iteration Complexity
linear	$\mu^T \quad (\mu < 1)$	$O\left(\log(\frac{1}{\epsilon})\right)$
sub-linear	$\frac{1}{T^\alpha} \quad \alpha > 0$	$O\left(\frac{1}{\epsilon^{1/\alpha}}\right)$

Why are we interested in Bounds?

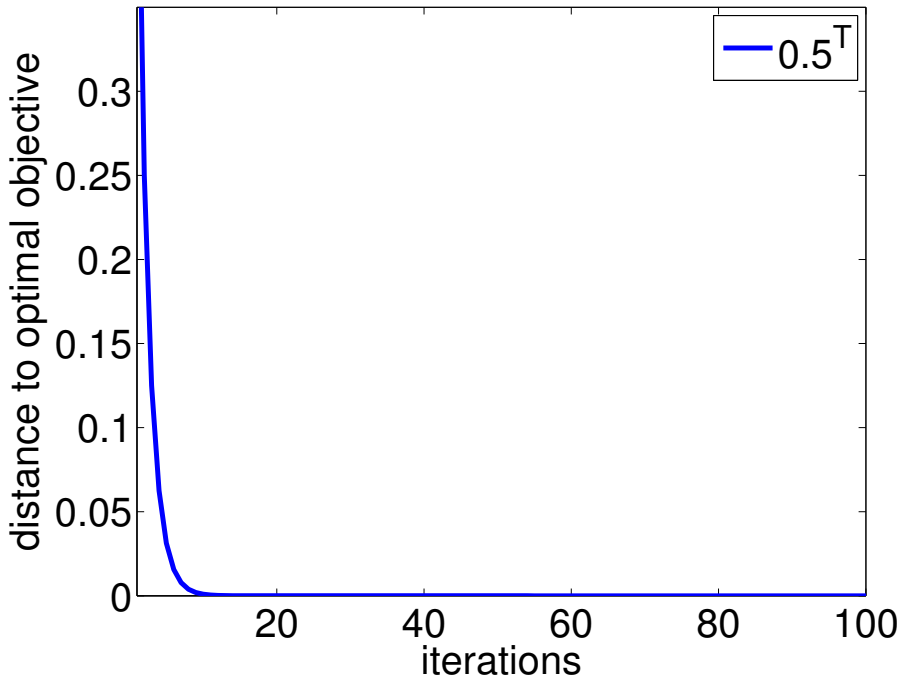
More on Convergence Measure

	Convergence Rate	Iteration Complexity
linear	$\mu^T \quad (\mu < 1)$	$O\left(\log(\frac{1}{\epsilon})\right)$
sub-linear	$\frac{1}{T^\alpha} \quad \alpha > 0$	$O\left(\frac{1}{\epsilon^{1/\alpha}}\right)$

Why are we interested in Bounds?

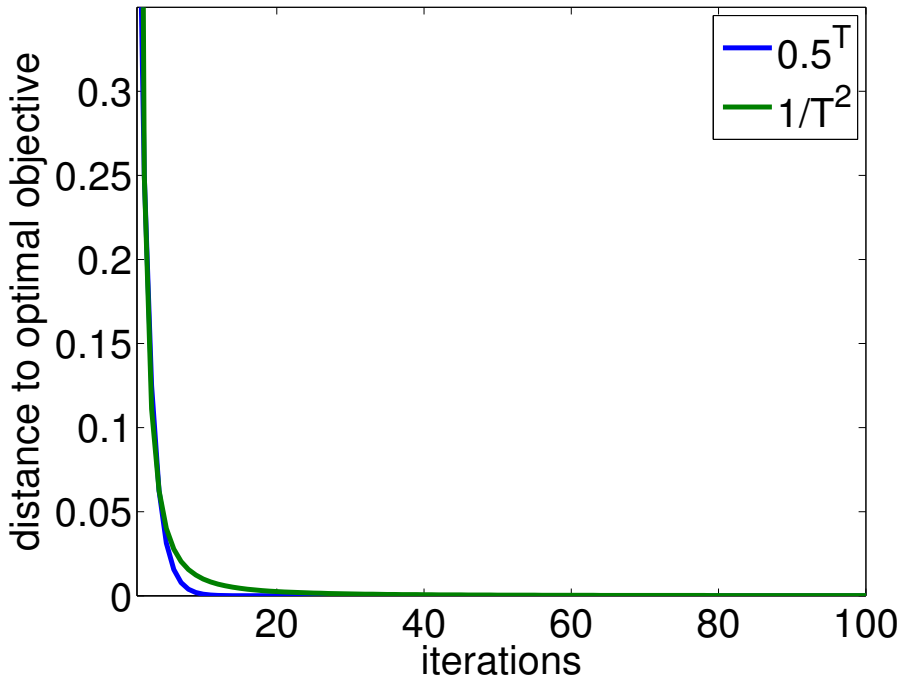
More on Convergence Measure

	Convergence Rate	Iteration Complexity
linear	$\mu^T \quad (\mu < 1)$	$O\left(\log(\frac{1}{\epsilon})\right)$
sub-linear	$\frac{1}{T^\alpha} \quad \alpha > 0$	$O\left(\frac{1}{\epsilon^{1/\alpha}}\right)$



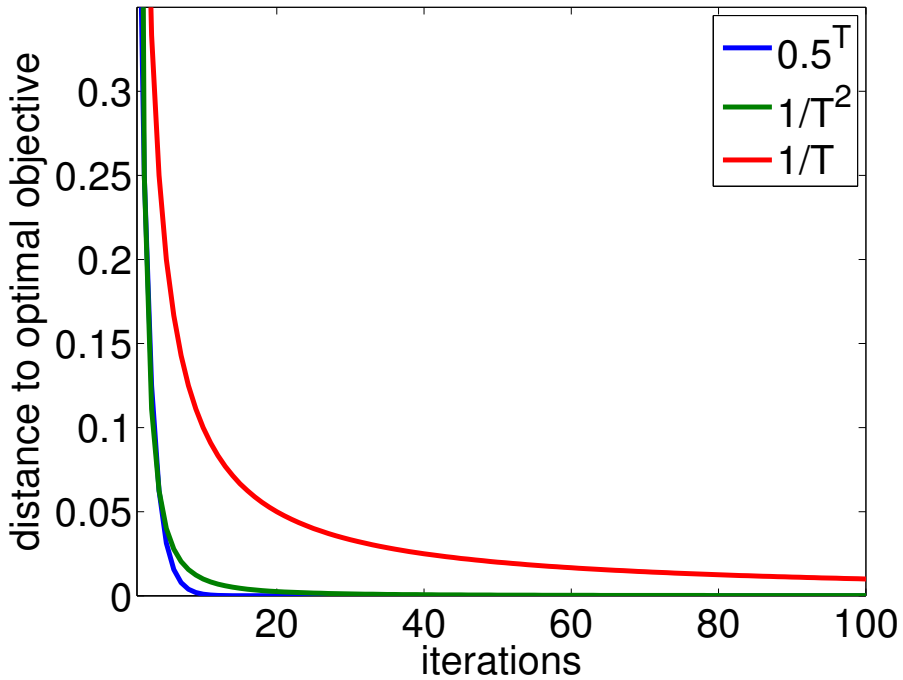
More on Convergence Measure

	Convergence Rate	Iteration Complexity
linear	$\mu^T \quad (\mu < 1)$	$O\left(\log(\frac{1}{\epsilon})\right)$
sub-linear	$\frac{1}{T^\alpha} \quad \alpha > 0$	$O\left(\frac{1}{\epsilon^{1/\alpha}}\right)$



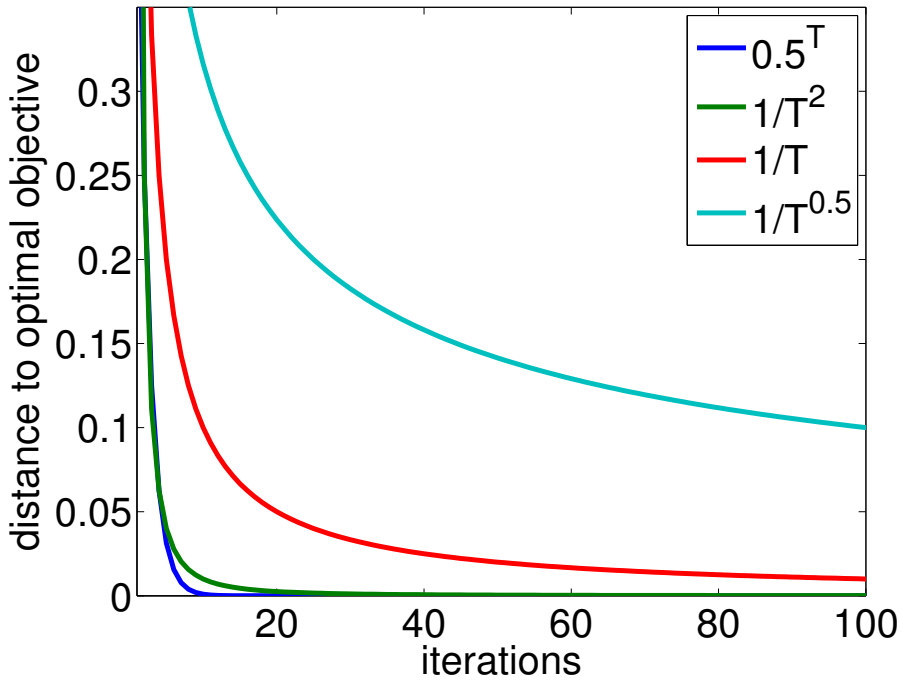
More on Convergence Measure

	Convergence Rate	Iteration Complexity
linear	$\mu^T \quad (\mu < 1)$	$O\left(\log(\frac{1}{\epsilon})\right)$
sub-linear	$\frac{1}{T^\alpha} \quad \alpha > 0$	$O\left(\frac{1}{\epsilon^{1/\alpha}}\right)$



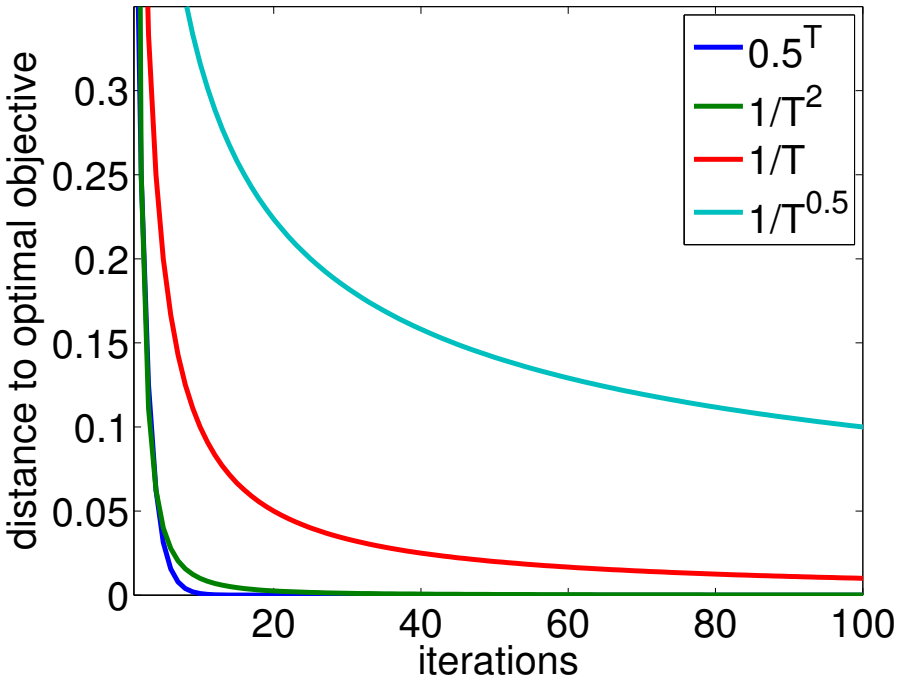
More on Convergence Measure

	Convergence Rate	Iteration Complexity
linear	$\mu^T \quad (\mu < 1)$	$O\left(\log(\frac{1}{\epsilon})\right)$
sub-linear	$\frac{1}{T^\alpha} \quad \alpha > 0$	$O\left(\frac{1}{\epsilon^{1/\alpha}}\right)$



More on Convergence Measure

	Convergence Rate	Iteration Complexity
linear	$\mu^T \quad (\mu < 1)$	$O\left(\log(\frac{1}{\epsilon})\right)$
sub-linear	$\frac{1}{T^\alpha} \quad \alpha > 0$	$O\left(\frac{1}{\epsilon^{1/\alpha}}\right)$



$$\mu^T < \frac{1}{T^2} < \frac{1}{T} < \frac{1}{\sqrt{T}}$$

$$\log\left(\frac{1}{\epsilon}\right) < \frac{1}{\sqrt{\epsilon}} < \frac{1}{\epsilon} < \frac{1}{\epsilon^2}$$

Factors that affect Iteration Complexity

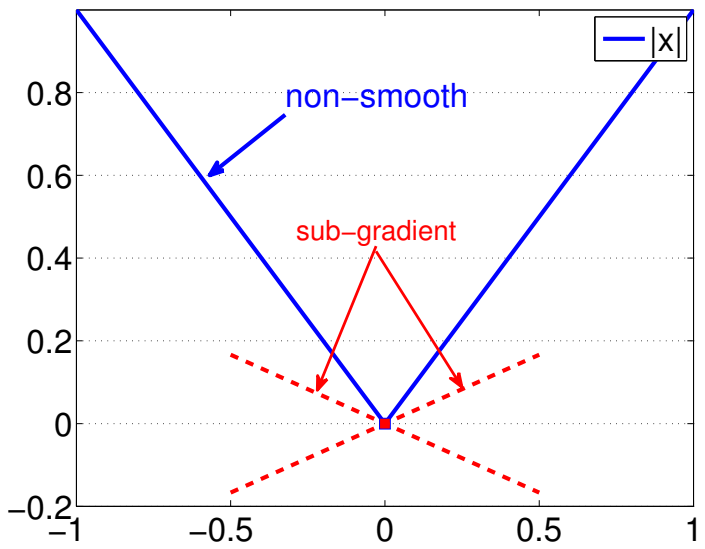
- Domain $\mathcal{X}$: size and geometry
- Size of problem: dimension and number of data points
- Property of function: smoothness of function

Warm-up: Non-smooth function

- Lipschitz continuous: e.g. absolute

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|$$

$$f(\mathbf{x}) \geq f(\mathbf{y}) + \partial f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})$$



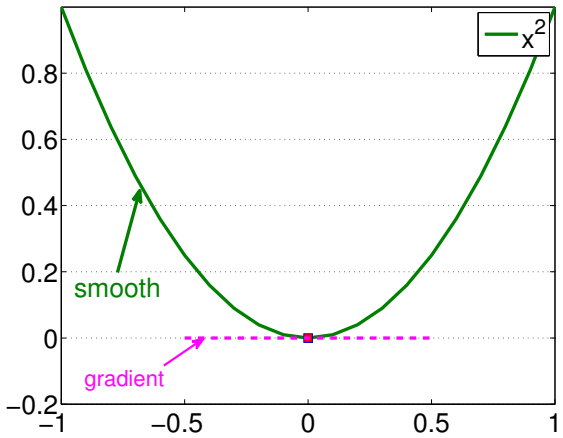
- Iteration complexity $O\left(\frac{1}{\epsilon^2}\right)$ or convergence rate $O\left(\frac{1}{\sqrt{T}}\right)$

Warm-up: Smooth Convex function

- smooth: e.g. logistic loss $f(z) = \log(1 + \exp(-z))$

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_* \leq \beta \|\mathbf{x} - \mathbf{y}\|$$

where $\beta > 0$



Second Order Derivative is upper bounded

- Full gradient descent $O\left(\frac{1}{\sqrt{\epsilon}}\right)$ or $O\left(\frac{1}{T^2}\right)$
- Stochastic gradient descent $O\left(\frac{1}{\epsilon^2}\right)$ or $O\left(\frac{1}{\sqrt{T}}\right)$

Warm-up: Strongly Convex function

- strongly convex: e.g. ℓ_2^2 norm $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2$

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_* \geq \lambda \|\mathbf{x} - \mathbf{y}\|, \quad \lambda > 0$$

- Second Order Derivative is lower bounded
- Iteration complexity $O(1/\epsilon)$ or convergence rate $O(1/T)$

Warm-up: Smooth and Strongly Convex function

- smooth and strongly convex:

$$\lambda\|\mathbf{x}-\mathbf{y}\| \leq \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_* \leq \beta\|\mathbf{x}-\mathbf{y}\|, \quad \beta \geq \lambda > 0$$

- e.g. square loss: $f(z) = \frac{1}{2}(1-z)^2$
- Iteration complexity $O\left(\log\left(\frac{1}{\epsilon}\right)\right)$ or convergence rate $O(\mu^T)$

Warm-up: Stochastic Gradient

$$\text{Loss: } L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i)$$

- stochastic gradient: $\nabla \ell(\mathbf{w}^\top \mathbf{x}_{i_t}, y_{i_t})$
- where $i_t \in \{1, \ldots, n\}$ randomly sampled
- key equation: $\mathrm{E}_{i_t}[\nabla \ell(\mathbf{w}^\top \mathbf{x}_{i_t}, y_{i_t})] = \nabla L(\mathbf{w})$
- computation is very cheap

Warm-up: Dual

Convex Conjugate: $\ell^*(\alpha) \Longleftrightarrow \ell(\mathbf{z})$

$$\ell(z) = \max_{\alpha \in \Omega} \alpha z - \ell^*(\alpha), \quad \textit{e.g.}\, \max(0, 1-z) = \max_{\alpha \leq 0} \alpha z - \alpha$$

Dual Objective (classification):

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\underbrace{y_i \mathbf{w}^\top \mathbf{x}_i}_{z}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

$$= \min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \left[\max_{\alpha_i \in \Omega} \alpha_i (y_i \mathbf{w}^\top \mathbf{x}_i) - \ell^*(\alpha_i) \right] + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

$$= \max_{\alpha \in \Omega} \frac{1}{n} \sum_{i=1}^{n} -\ell^*(\alpha_i) - \frac{\lambda}{2} \left\| \frac{1}{\lambda n} \sum_{i=1} \alpha_i y_i \mathbf{x}_i \right\|_2^2$$

Outline

STOP Algorithms for Big Data Classification and Regression

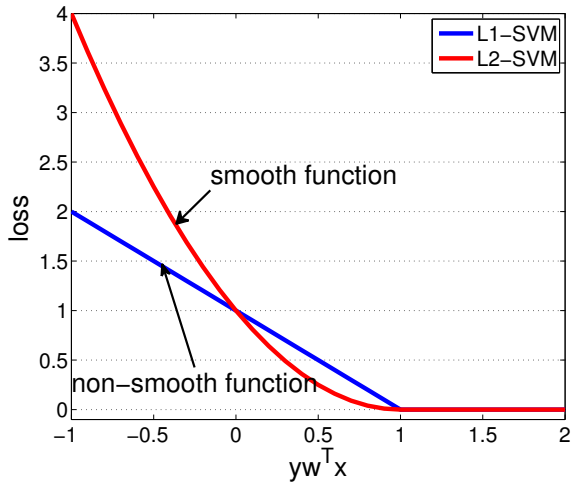
In this section, we present several well-known STOP algorithms for solving classification and regression problems

- Coverage:
  - Stochastic Gradient Descent (Pegasos) for L1-SVM (primal)
  - Stochastic Dual Coordinate Ascent (SDCA) for L2-SVM (dual)
  - Stochastic Average Gradient (SAG) for Logistic Regression/Regression (primal)

STOP Algorithms for Big Data Classification

Support Vector Machines:

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell(\mathbf{w}^\top \mathbf{x}_i, y_i)}_{loss} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

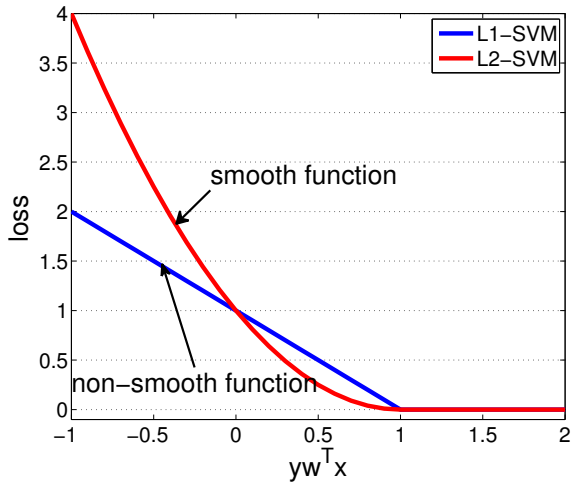


- L_1 SVM: $\ell(\mathbf{w}^\top \mathbf{x}, y) = \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$
- L_2 SVM: $\ell(\mathbf{w}^\top \mathbf{x}, y) = \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)^2$
- Equivalent to C-SVM formulations $C = n\lambda$

STOP Algorithms for Big Data Classification

Support Vector Machines:

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell(\mathbf{w}^\top \mathbf{x}_i, y_i)}_{loss} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

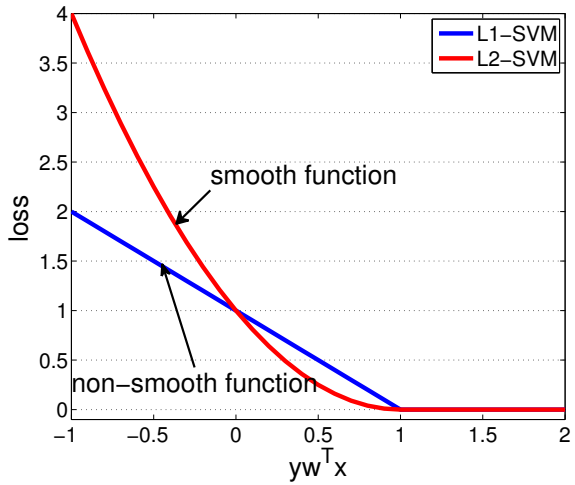


- L_1 SVM: $\ell(\mathbf{w}^\top \mathbf{x}, y) = \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$
- L_2 SVM: $\ell(\mathbf{w}^\top \mathbf{x}, y) = \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)^2$
- Equivalent to C-SVM formulations $C = n\lambda$

STOP Algorithms for Big Data Classification

Support Vector Machines:

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell(\mathbf{w}^\top \mathbf{x}_i, y_i)}_{loss} + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$



- L_1 SVM: $\ell(\mathbf{w}^\top \mathbf{x}, y) = \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$
- L_2 SVM: $\ell(\mathbf{w}^\top \mathbf{x}, y) = \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)^2$
- Equivalent to C-SVM formulations $C = n\lambda$

Pegasos (Shai et al. ICML '07, primal): STOP for L_1 SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i) + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{\text{strongly convex}}$$

Pegasos (SGD for strongly convex) :

$$\text{stochastic gradient: } \partial \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) = \begin{cases} -y_i \mathbf{x}_i, & 1 - y_i \mathbf{w}^\top \mathbf{x}_i < 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\partial f(\mathbf{w}_t; i_t) = \partial \ell(\mathbf{w}_t^\top \mathbf{x}_{i_t}, y_{i_t}) + \lambda \mathbf{w}_t$$

Stochastic Gradient Descent Updates

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{\lambda t}\partial f(\mathbf{w}_t; i_t)$$

SDCA (C.J. Hsieh, ICML '08, Shai&Tong JMLR'12, dual): STOP for L_2 SVM

convex conjugate: $\max(0, 1 - y\mathbf{w}^\top \mathbf{x})^2 = \max_{\alpha \geq 0} \left(\alpha - \frac{1}{4}\alpha^2 \right) - \alpha y \mathbf{w}^\top \mathbf{x}$

Dual SVM: $\max_{\alpha \geq 0} D(\alpha) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\left(\alpha_i - \frac{\alpha_i^2}{4} \right)}_{\phi^*(\alpha_i)} - \frac{\lambda}{2} \left\| \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \right\|_2^2$

primal solution: $\mathbf{w}_t = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i^t y_i \mathbf{x}_i$

SDCA: stochastic dual coordinate ascent

Dual Coordinate Updates

$$\Delta \alpha_i = \max_{\alpha_i^t + \Delta \alpha_i \geq 0} \phi^*(\alpha_i^t + \Delta \alpha_i) - \frac{\lambda}{2} \left\| \mathbf{w}_t + \frac{1}{\lambda n} \Delta \alpha_i y_i \mathbf{x}_i \right\|_2^2$$

SAG (Schmidt et al. '11, primal): Stopt for Logit Reg./ Reg.

- smooth and strongly convex objective

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{\text{smooth and strongly convex}}$$

- e.g. logistic regression, least square regression

SAG (stochastic average gradient):

$$\mathbf{w}_{t+1} = (1-\eta\lambda)\mathbf{w}_t - \boxed{\frac{\eta}{n}\sum_{i=1}^{n} g_i^t}, \quad \text{where}$$

$$g_i^t = \begin{cases} \partial\ell(\mathbf{w}_t^\top \mathbf{x}_i, y_i), & \text{if } i \text{ is selected} \\ g_i^{t-1}, & \text{otherwise} \end{cases}$$

OK, tell me which one is the best?

Let us compare Theoretically first

alg.	Pegasos	SAG	SDCA
$f(\mathbf{w})$	sc	sm & sc	sm loss & sc reg.
e.g.	SVMs, LR, LSR	L_2-SVM, LR, LSR	SVMs, LR, LSR
com. cost	$O(d)$	$O(d)$	$O(d)$
mem. cost	$O(d)$	$O(n+d)$	$O(n+d)$
Iteration	$O\left(\frac{1}{\lambda\epsilon}\right)$	$O\left((n+\frac{1}{\lambda})\log\left(\frac{1}{\epsilon}\right)\right)$	$O\left((n+\frac{1}{\lambda})\log\left(\frac{1}{\epsilon}\right)\right)$

Table : sc: strongly convex; sm: smooth; LR: logistic regression; LSR: least square regression

What about ℓ_1 regularization?

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \sigma \underbrace{\sum_{g=1}^{K} \|\mathbf{w}_g\|_1}_{\text{Lasso or Group Lasso}}$$

Issue: Not Strongly Convex Solution: Add ℓ_2^2 regularization

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \sigma \sum_{g=1}^{K} \|\mathbf{w}_g\|_1 + \boxed{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}$$

setting $\lambda = \Theta(1/\epsilon)$

Algorithm: Pegasos, SDCA

Outline

General Strategies for Stochastic Optimization

General strategies for Stochastic Gradient Descent

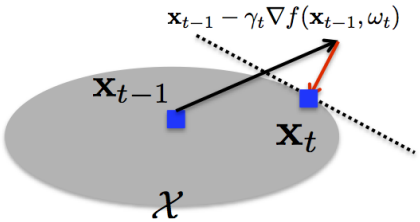
- Coverage:
  - SGD for various objectives
  - Accelerated SGD for Stochastic Composite Optimization
  - Variance Reduced SGD
  - Parallel and Distributed Optimization

Stochastic Gradient Descent

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

- stochastic gradient $\nabla f(\mathbf{x};\omega)$: ω is a random variable
- SGD updates:

$$x_t \leftarrow \Pi_{\mathcal{X}}\left[x_{t-1} - \gamma_t \partial f(x_{t-1};\omega_t)\right]$$



$$\Pi_{\mathcal{X}}[\hat{\mathbf{x}}] = \min_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$$

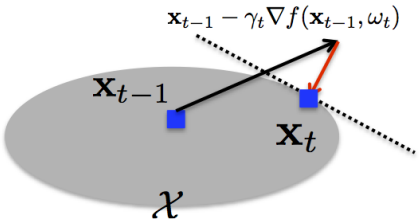
Issues: How to determine learning rate γ_t? key: decreasing

Stochastic Gradient Descent

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

- stochastic gradient $\nabla f(\mathbf{x};\omega)$: ω is a random variable
- SGD updates:

$$x_t \leftarrow \Pi_{\mathcal{X}}\left[x_{t-1} - \gamma_t \partial f(x_{t-1};\omega_t)\right]$$

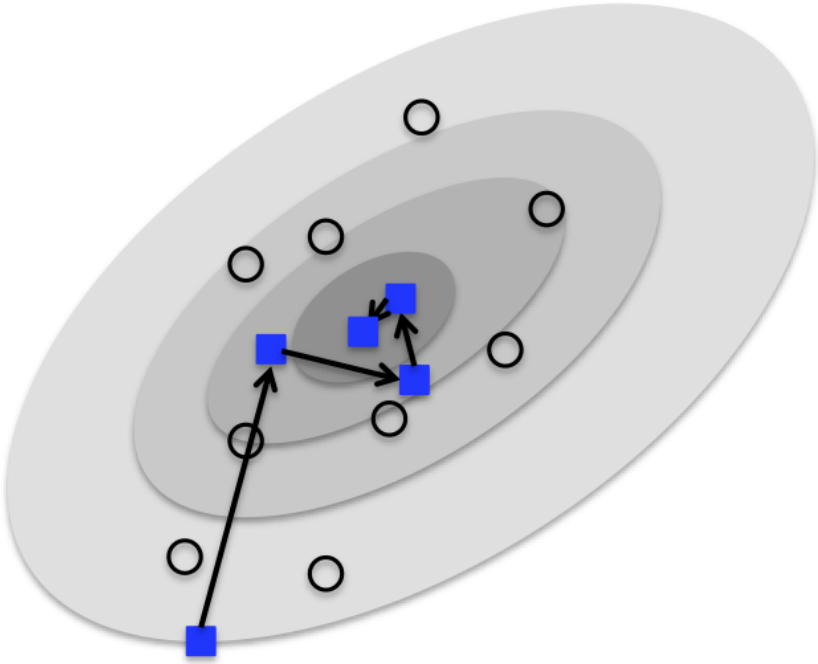


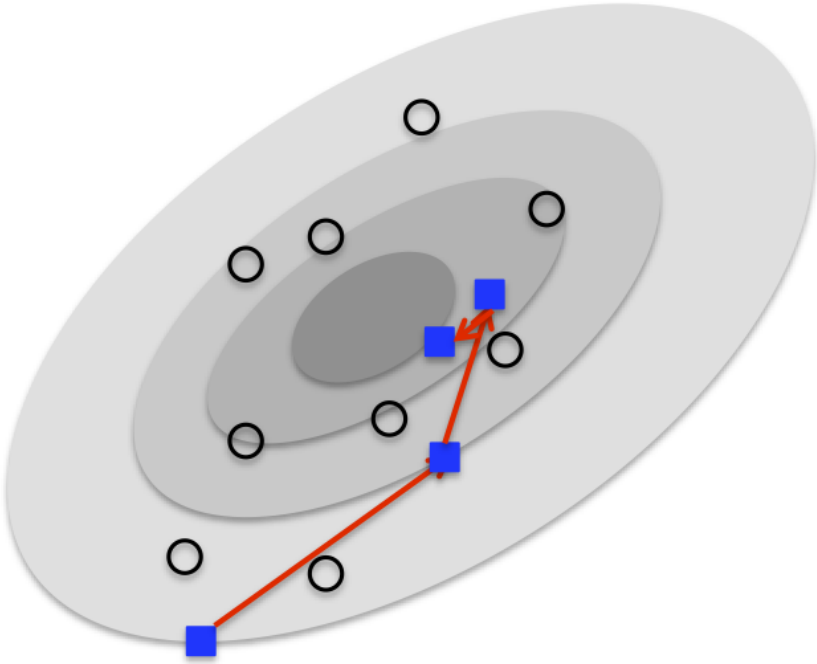
$$\Pi_{\mathcal{X}}[\hat{\mathbf{x}}] = \min_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$$

Issues: How to determine learning rate γ_t? key: decreasing

Why Stochastic Gradient Descent Works?

Why decreasing step size? GD vs SGD





$$\mathbf{x}_t = \mathbf{x}_{t-1} - \eta \nabla f(\mathbf{x}_{t-1})$$

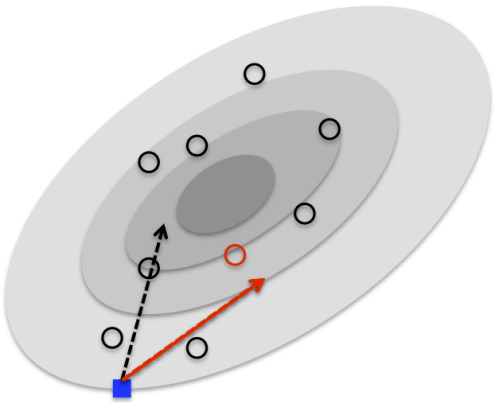
$$\nabla f(\mathbf{x}_{t-1}) \to 0$$

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \eta_t \nabla f(\mathbf{x}_{t-1}; \omega_t)$$

$$\nabla f(\mathbf{x}_{t-1}; \omega_t) \not\to 0, \quad \eta_t \to 0$$

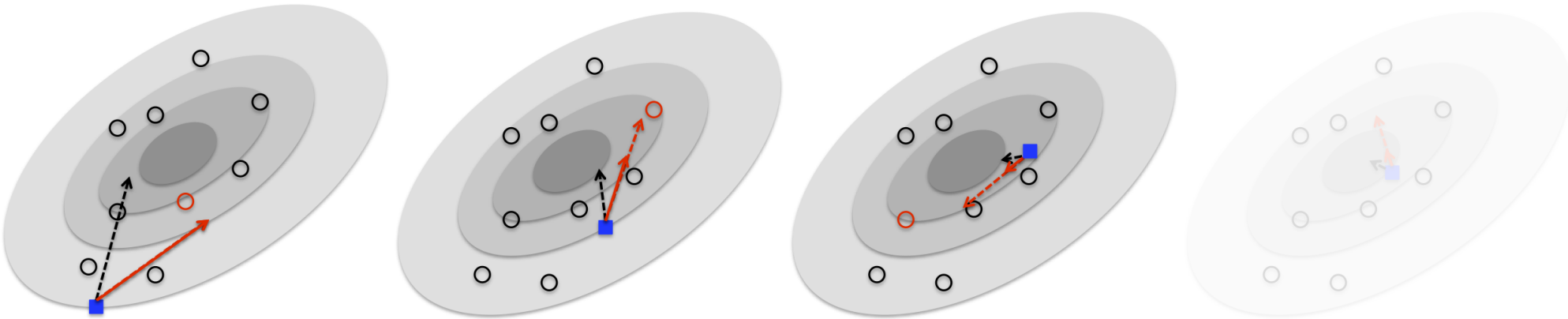
Why Stochastic Gradient Descent Works?

Why Stochastic Gradient Descent Works?

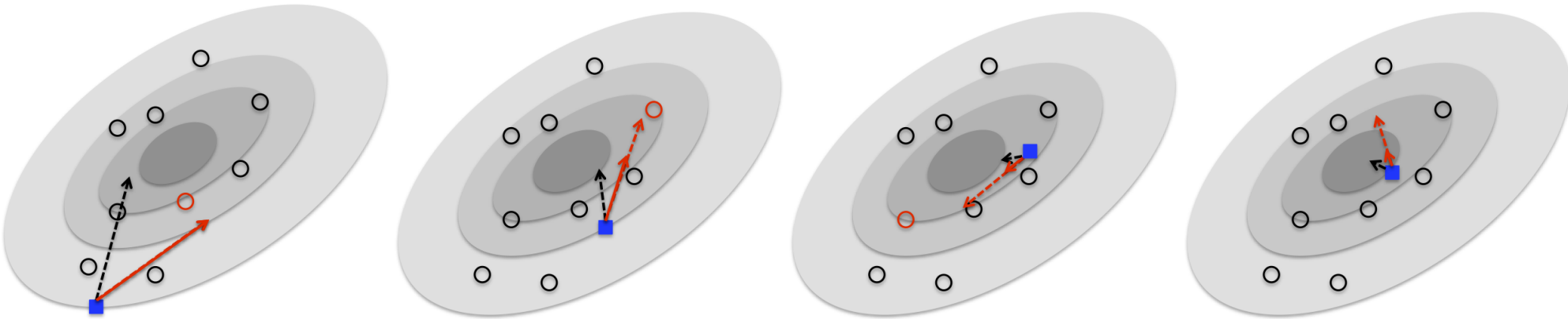


Why Stochastic Gradient Descent Works?

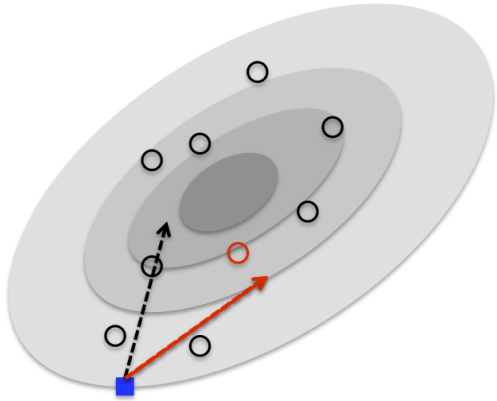
Why Stochastic Gradient Descent Works?



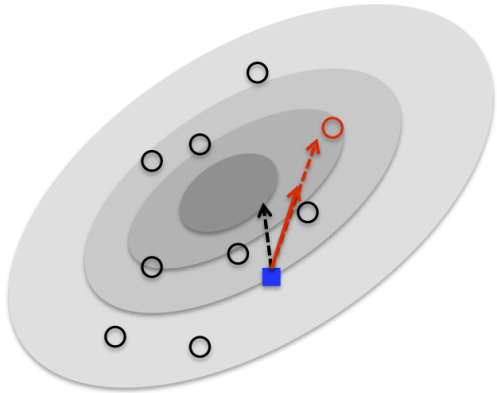
Why Stochastic Gradient Descent Works?



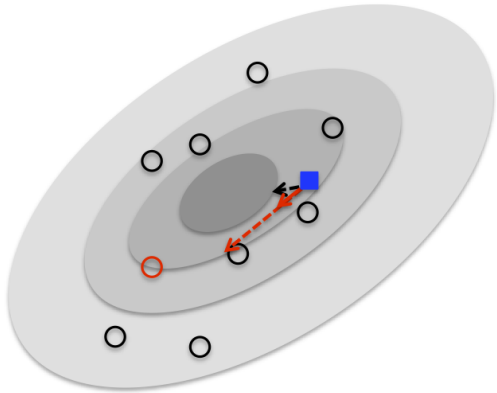
Why Stochastic Gradient Descent Works?



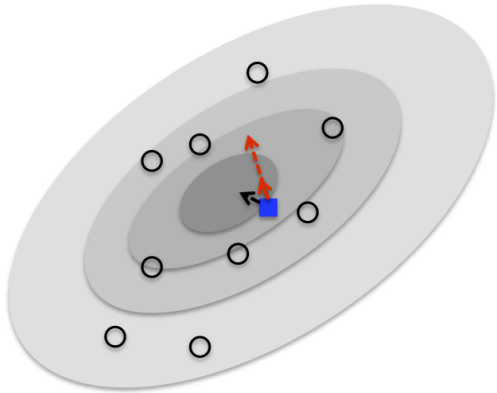
Why Stochastic Gradient Descent Works?



Why Stochastic Gradient Descent Works?



Why Stochastic Gradient Descent Works?



How to decrease the Step size?

Step size Depends on the property of the function.

SGD for Non-smooth Function: Convergence Rate

- $\|\mathbf{x} - \mathbf{y}\| \leq D$ and $\|\partial f(x;\omega)\|_* \leq G$, $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$.
- Step size: $\gamma_t = \frac{c}{\sqrt{t}}$, c usually needed to be tuned
- Convergence rate of final solution $\mathbf{x}_T$: $O\left(\left(\frac{D^2}{c} + cG^2\right)\frac{\log T}{T}\right)$
- Are we good enough? Close but not yet
- minimax optimal rate : $O\left(\frac{1}{\sqrt{T}}\right)$

SGD for Non-smooth Function: Convergence Rate

- $\|\mathbf{x} - \mathbf{y}\| \leq D$ and $\|\partial f(x;\omega)\|_* \leq G$, $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$.
- Step size: $\gamma_t = \frac{c}{\sqrt{t}}$, c usually needed to be tuned
- Convergence rate of final solution $\mathbf{x}_T$: $O\left(\left(\frac{D^2}{c} + cG^2\right)\frac{\log T}{T}\right)$
- Are we good enough? Close but not yet
- minimax optimal rate : $O\left(\frac{1}{\sqrt{T}}\right)$

SGD for Non-smooth Function: Convergence Rate

- $\|\mathbf{x} - \mathbf{y}\| \leq D$ and $\|\partial f(x;\omega)\|_* \leq G$, $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$.
- Step size: $\gamma_t = \frac{c}{\sqrt{t}}$, c usually needed to be tuned
- Convergence rate of final solution $\mathbf{x}_T$: $O\left(\left(\frac{D^2}{c} + cG^2\right)\frac{\log T}{T}\right)$
- Are we good enough? Close but not yet
- minimax optimal rate : $O\left(\frac{1}{\sqrt{T}}\right)$

SGD for Non-smooth Function: Convergence Rate

- $\|\mathbf{x} - \mathbf{y}\| \leq D$ and $\|\partial f(x;\omega)\|_* \leq G$, $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$.
- Step size: $\gamma_t = \frac{c}{\sqrt{t}}$, c usually needed to be tuned
- Convergence rate of final solution $\mathbf{x}_T$: $O\left(\left(\frac{D^2}{c} + cG^2\right)\frac{\log T}{T}\right)$
- Are we good enough? Close but not yet
- minimax optimal rate : $O\left(\frac{1}{\sqrt{T}}\right)$

SGD for Non-smooth Function: Convergence Rate

- minimax optimal rate : $O\left(\frac{1}{\sqrt{T}}\right)$
- Averaging:

$$\bar{\mathbf{x}}_t = \left(1 - \frac{1+\eta}{t+\eta}\right)\bar{\mathbf{x}}_{t-1} + \frac{1+\eta}{t+\eta}\mathbf{x}_t, \quad \eta \geq 0$$

- $\eta = 0$ standard average

$$\bar{\mathbf{x}}_T = \frac{\mathbf{x}_1 + \cdots + \mathbf{x}_T}{T}$$

- Convergence Rate of $\bar{\mathbf{x}}_T$: $O\left(\eta\left(\frac{D^2}{c} + cG^2\right)\frac{1}{T}\right)$

SGD for Non-smooth Strongly Convex Function: Convergence Rate

- λ-Strongly Convex minimax optimal rate : $O(\frac{1}{T})$
- Step size: $\gamma_t = \frac{1}{\lambda t}$
- Convergence Rate of $\mathbf{x}_T$: $O\left(\frac{G^2 \log T}{\lambda T}\right)$
- Averaging:

$$\bar{\mathbf{x}}_t = \left(1 - \frac{1+\eta}{t+\eta}\right)\bar{\mathbf{x}}_{t-1} + \frac{1+\eta}{t+\eta}\mathbf{x}_t, \quad \eta \geq 0$$

- Convergence Rate of $\bar{\mathbf{x}}_T$: $O\left(\frac{\eta G^2}{\lambda T}\right)$

SGD for Smooth function

- Generally No Further improvement upon $O\left(\frac{1}{\sqrt{T}}\right)$
- Special structure may yield better convergence
- Stochastic Composite Optimization: Smooth + Non-Smooth

General Strategies for Stochastic Optimization

General strategies for Stochastic Gradient Descent

- Coverage:
  - SGD for various objectives
  - Accelerated SGD for Stochastic Composite Optimization
  - Variance Reduced SGD
  - Parallel and Distributed Optimization

Stochastic Composite Optimization

$$\min_{\mathbf{x}\in\mathcal{X}} \underbrace{f(\mathbf{x})}_{\textit{smooth}} + \underbrace{g(\mathbf{x})}_{\textit{nonsmooth}}$$

- Example 1

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i\mathbf{w}^\top\mathbf{x}_i)) + \lambda\|\mathbf{w}\|_1$$

- Example 2

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(y_i-\mathbf{w}^\top\mathbf{x}_i)^2 + \lambda\|\mathbf{w}\|_1$$

Stochastic Composite Optimization

- Accelerated Stochastic Gradient
  - auxiliary sequence of search solutions
  - maintain three solutions: $\mathbf{x}_t$ (action solution), $\bar{\mathbf{x}}_t$ (averaged solution), $\mathbf{x}_t^m$ (auxiliary solution)
  - Similar to Nesterov's deterministic method
  - several variants

- Stochastic Dual Coordinate Ascent

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \sigma \sum_{g=1}^{K} \|\mathbf{w}_g\|_1 + \boxed{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}$$

G. Lan's Accelerated Stochastic Approximation

Iterate $t = 1, \ldots, T$

1. compute auxiliary solution

$$\boxed{\mathbf{x}_t^m = \beta_t^{-1}\mathbf{x}_t + (1-\beta_t^{-1})\bar{\mathbf{x}}_t}$$

2. update solution

$$\mathbf{x}_{t+1} = P_{\mathbf{x}_t}(\gamma_t G(\mathbf{x}_t^m; \omega_t)) \text{ (think about projection)}$$

3. update averaged solution

$$\bar{\mathbf{x}}_{t+1} = (1-\beta_t^{-1})\bar{\mathbf{x}}_t + \beta_t^{-1}\mathbf{x}_{t+1} \text{ (Similar to SGD)}$$

γ_t, β_t, Convergence Rate?

Accelerated Stochastic Approximation

1. $\beta_t^{-1} = \frac{2}{t+1}$:

$$\bar{\mathbf{x}}_{t+1} = \left(1 - \frac{2}{t+1}\right)\bar{\mathbf{x}}_t + \frac{2}{t+1}\mathbf{x}_{t+1}$$

2. $\gamma_t = \frac{t+1}{2}\gamma$: Increasing, Never like before

3. Convergence rate of $\bar{\mathbf{x}}_T$:

$$O\left(\frac{L}{T^2} + \frac{M+\sigma^2}{\sqrt{T}}\right)$$

Variance Reduction: Mini-batch

- Batch gradient: $g_t = \frac{1}{b}\sum_{i=1}^{b} \partial f(x_{t-1}, \omega_{ti})$
- Unbiased sampling: $\mathbb{E}g_t = \mathbb{E}\partial f(x_{t-1}, \omega)$.
- Variance reduced: $\mathbb{V}g_t = \frac{1}{b}\mathbb{V}\partial f(x_{t-1}, \omega)$.
- Tradeoff the batch computation for variance decrease.
- Batch size
  1. Constant batch size;
  2. Increasing batch size as proportion to t.
- Convergence Rate: e.g. $O(\frac{1}{b\lambda T})$ of SGD for strongly convex

Variance Reduction: Mixed-Grad (Mahdavi et. al, Johnson et al. '13)

Iterate $s = 1, \ldots,$

Iterate $t = 1, \ldots, m$

$$\mathbf{x}_t^s = \mathbf{x}_{t-1}^s - \eta \underbrace{\left(\nabla f(\mathbf{x}_{t-1}^s; \omega_t) - \nabla f(\bar{\mathbf{x}}^{s-1}; \omega_t) + \nabla f(\bar{\mathbf{x}}^{s-1})\right)}_{StoGrad - StoGrad + Grad}$$

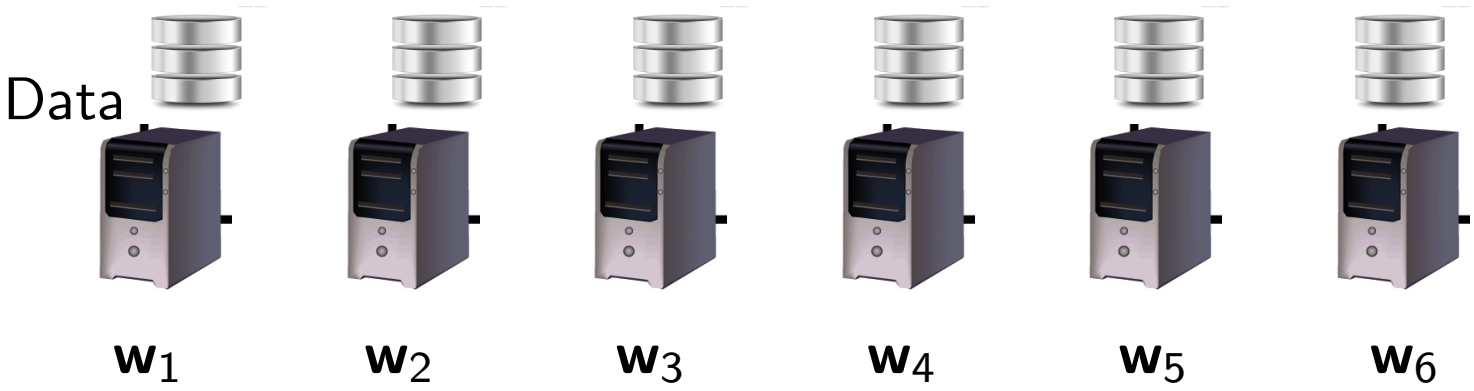
update $\bar{\mathbf{x}}^s$

- how variance is reduced?
  if $\bar{\mathbf{x}}^{s-1} \to \mathbf{x}^*$, $\nabla f(\bar{\mathbf{x}}_{t-1}) \to 0$, $\nabla f(\mathbf{x}_{t-1}; \omega_t) - \nabla f(\mathbf{x}^*; \omega_t) \to 0$
- $\bar{\mathbf{x}}^s = \mathbf{x}_m^s$ or $\bar{\mathbf{x}}^s = \sum_{t=1}^m \mathbf{x}_t^s / m$
- Constant learning rate
- Better Iteration Complexity.: $O((n + \frac{1}{\lambda}) \log\left(\frac{1}{\epsilon}\right))$ for sm&sc, $O(1/\epsilon)$ for smooth

Distributed Optimization: Why?

- data distributed over multiple machines
- moving to single machine suffers
  - low network bandwidth
  - limited disk or memory
- benefits from parallel computation
  - cluster of machines
  - GPUs

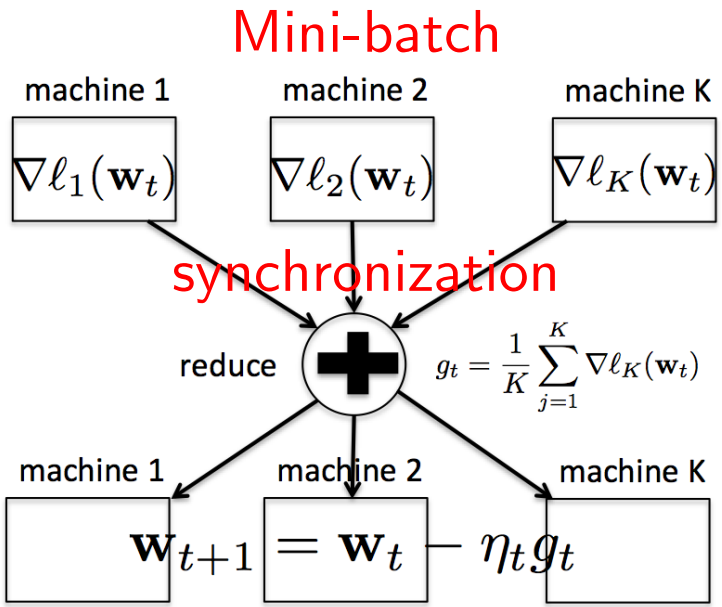
A Naive Solution



$$\mathbf{w} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{w}_i$$

Issue: Not the Optimal

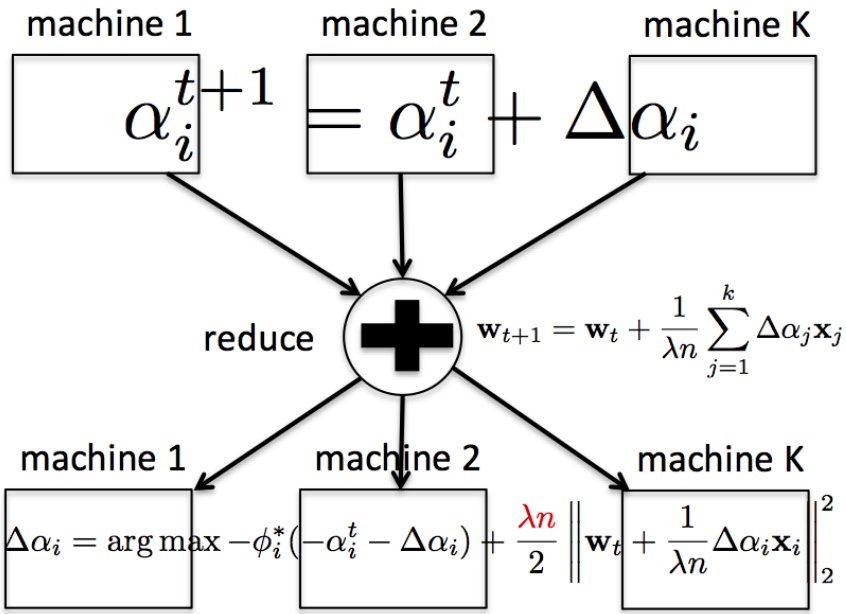
Distribute SGD is simple



Mini-batch SGD

- Good: reduced variance, faster conv.
- Bad: synchronization is expensive
- Solutions:
  - asynchronized update
    - convergence: difficult to prove
  - lesser synchronizations
    - guaranteed convergence

Distribute SDCA is not trivial



issue: data are correlated

$$\Delta\alpha_i = \arg\max -\phi_i^*(-\alpha_i^t - \Delta\alpha_i) - \frac{\lambda n}{2}\left\|\mathbf{w}_t + \frac{1}{\lambda n}\Delta\alpha_i\mathbf{x}_i\right\|_2^2$$

Not Working

Distribute SDCA (T. Yang NIPS'13)

The Basic Variant

mR-SDCA running on machine k

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

> **Iterate:** for $j = 1, \ldots, m$
>
> Randomly pick $i \in \{1, \cdots, n_k\}$ and let $i_j = i$
>
> Find $\Delta\alpha_{k,i}$ by **IncDual**($\mathbf{w} = \mathbf{w}_{t-1}$, $scl = mK$)
>
> Set $\alpha_{k,i}^t = \alpha_{k,i}^{t-1} + \Delta\alpha_{k,i}$

Reduce: $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \frac{1}{\lambda n}\sum_{j=1}^m \Delta\alpha_{k,i_j} x_{k,i_j}$

$$\Delta\alpha_i = \arg\max -\phi_i^*(-\alpha_i^t - \Delta\alpha_i) - \frac{\lambda n}{2mK}\left\|\mathbf{w}_t + \frac{mK}{\lambda n}\Delta\alpha_i \mathbf{x}_i\right\|_2^2$$

Distribute SDCA (T. Yang NIPS'13)

The Practical Variant

Initialize: $\mathbf{u}_t^0 = \mathbf{w}^{t-1}$

Iterate: for $j = 1, \ldots, m$

Randomly pick $i \in \{1, \cdots, n_k\}$ and let $i_j = i$

Find $\Delta\alpha_{k,i_j}$ by **IncDual**($\mathbf{w} = \mathbf{u}_t^{j-1}$, $scl = k$)

Update $\alpha_{k,i_j}^t = \alpha_{k,i_j}^{t-1} + \Delta\alpha_{k,i_j}$

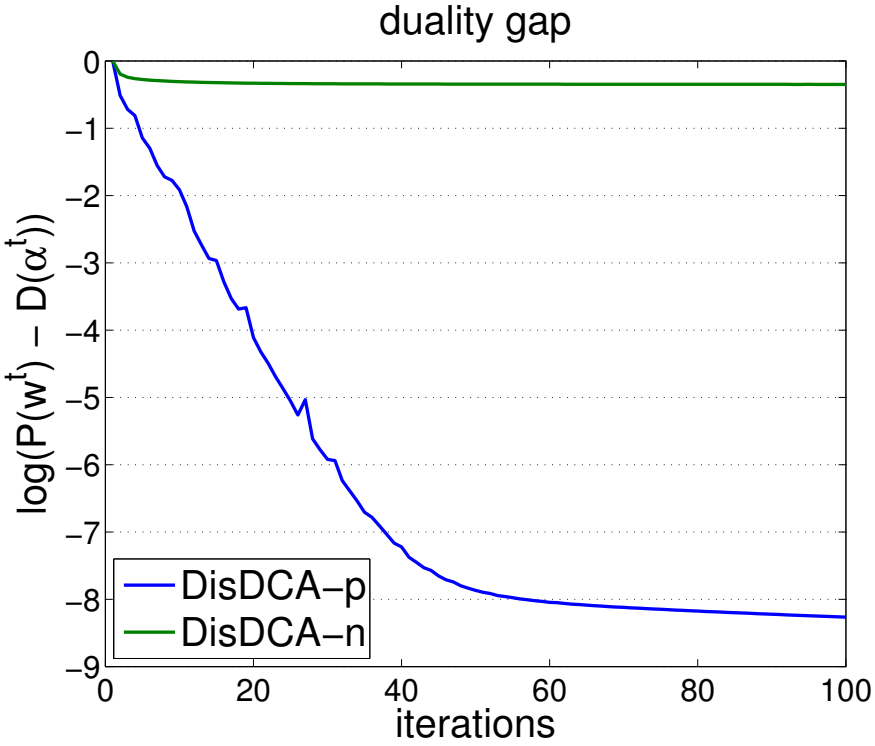
Update $\mathbf{u}_t^j = \mathbf{u}_t^{j-1} + \frac{1}{\lambda n_k}\Delta\alpha_{k,i_j}\mathbf{x}_{k,i_j}$

$$\Delta\alpha_{i_j} = \arg\max -\phi_{i_j}^*(-\alpha_{i_j}^t - \Delta\alpha_{i_j}) - \frac{\lambda n}{2K}\left\|\mathbf{u}_t^{j-1} + \frac{K}{\lambda n}\Delta\alpha_{i_j}\mathbf{x}_{i_j}\right\|_2^2$$

Using Updated Information and Large Step Size

Distribute SDCA (T. Yang NIPS'13)

The Practical Variant is much faster than the Basic Variant



Outline

Implementations and A Library

In this section, we present some techniques for efficient implementations and a practical library

- Coverage:
  - efficient averaging
  - Gradient sparsification
  - pre-optimization: screening
  - pre-optimization: data preconditioning
  - distributed (parallel) optimization library

Efficient Averaging

- Update rule:

$$x_t = (1 - \gamma_t \lambda) x_{t-1} + \gamma_t g_t$$
$$\bar{x}_t = (1 - \alpha_t) \bar{x}_{t-1} + \alpha_t x_t$$

- Efficient update when g_t has many 0, or g_t is sparse,

$$S_t = \begin{pmatrix} 1 - \lambda\gamma_t & 0 \\ \alpha_t(1 - \lambda\gamma_t) & 1 - \alpha_t \end{pmatrix} S_{t-1}, \quad S_1 = I$$
$$y_t = y_{t-1} - [S_t^{-1}]_{11} \gamma_t g_t$$
$$\tilde{y}_t = \tilde{y}_{t-1} - ([S_t^{-1}]_{21} + [S_t^{-1}]_{22} \alpha_t) \gamma_t g_t$$

$$x_T = [S_T]_{11} y_T$$
$$\bar{x}_T = [S_T]_{21} y_T + [S_T]_{22} \tilde{y}_T$$

When Gradient is Sparse

Gradient sparsification

- Sparsification by importance sampling

$$R_{ti} = \mathrm{unif}(0, 1)$$
$$\tilde{g}_{ti} = g_{ti}[|g_{ti}| \geq \hat{g}_i] + \hat{g}\mathrm{sign}(g_{ti})[\hat{g}_i R_{ti} \leq |g_{ti}| < \hat{g}_i]$$

- Unbiased sample: $\mathbb{E}\tilde{g}_t = g_t$.
- Tradeoff variance increase for the efficient computation.

Especially useful for Logistic Regression

Pre-optimization: Screening

- Screening for Lasso (Wang et al., 2012)
- Screening for SVM (Ogawa et al., 2013)

Distributed Optimization Library: Birds

- The birds library implements distributed stochastic dual coordinate ascent (DisDCA) for classification and regression with a broad support.
- For technical details see:
  - "Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent." Tianbao Yang. NIPS 2013.
  - "On the Theoretical Analysis of Distributed Stochastic Dual Coordinate Ascent" Tianbao Yang, etc. Tech Report 2013.
- The code is distributed under GNU General Publich License (see license.txt for details).

Distributed Optimization Library: Birds

What problems does it Solve?

- Classification and Regression
- Loss
  1. Hinge loss (SVM): L-1 hinge loss, L-2 hinge loss
  2. Logistic loss (Logistic Regression)
  3. Least Square Regression (Ridge Regression)
- Regularizer
  1. ℓ_2 norm: SVM, Logistic Regression, Ridge Regression
  2. ℓ_1 norm: Lasso, SVM, LR with ℓ_1 norm
- Multi-class : one-vs-all

Distributed Optimization Library: Birds

What data does it Support?

- dense, sparse
- txt, binary

What environment does it Support?

- Prerequisites: Boost Library
- Tested on A cluster of Linux Servers (up to hundreds of machines)
- Tested on Cygwin in Windows with multi-core

How about kernel methods?

- Stochastic/Online Optimization Approaches: (Jin et al., 2010; Orabona et al., 2012),

- Linearization + STOP for linear methods
  - the Nyström method (Drineas & Mahoney, 2005)
  - Random Fourier Features (Rahimi & Recht, 2007)
  - Comparison of two (Yang et al., 2012)

References I

Drineas, Petros and Mahoney, Michael W. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175, 2005.

Jin, Rong, Hoi, Steven C. H., and Yang, Tianbao. Online multiple kernel learning: Algorithms and mistake bounds. In *Proceedings of the 21st International Conference on Algorithmic Learning Theory*, pp. 390–404, 2010.

Ogawa, Kohei, Suzuki, Yoshiki, and Takeuchi, Ichiro. Safe screening of non-support vectors in pathwise svm computation. In *ICML (3)*, pp. 1382–1390, 2013.

Orabona, Francesco, Luo, Jie, and Caputo, Barbara. Multi kernel learning with online-batch optimization. *Journal of Machine Learning Research*, 13:227–253, 2012.

References II

Rahimi, Ali and Recht, Benjamin. Random features for large-scale kernel machines. In *NIPS*, 2007.

Wang, Jie, Lin, Binbin, Gong, Pinghua, Wonka, Peter, and Ye, Jieping. Lasso screening rules via dual polytope projection. *CoRR*, abs/1211.3966, 2012.

Yang, Tianbao, Li, Yu-Feng, Mahdavi, Mehrdad, Jin, Rong, and Zhou, Zhi-Hua. "nystrom method vs random fourier features: A theoretical and empirical comparison". In *NIPS*, pp. 485–493, 2012.