Lecture 5:
Semi-Stochastic Methods

Peter Richtárik

Graduate School in Systems, Optimization, Control and Networks
Belgium 2015

S2GD

Jakub Konecny and P.R.
Semi-Stochastic Gradient Descent Methods
arXiv:1312.1666, 2013

Further Papers

Rie Johnson and Tong Zhang
Accelerating Stochastic Gradient Descent using Predictive Variance Reduction
Neural Information Processing Systems, 2013

Jakub Konecny, Zheng Qu and P.R.
Semi-Stochastic Coordinate Descent
arXiv:1412.6293, 2014

Jakub Konecny, Jie Liu, P.R. and Martin Takac
Mini-Batch Semi-Stochastic Gradient Descent in the Proximal Setting
IEEE Journal of Selected Topics in Signal Processing, 2015

The Problem

Minimizing Average Loss

- Problems are often structured

n is big

$$\min_{x \in \mathbb{R}^d} \left\{ F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$$

- Arising in machine learning, signal processing, engineering, …

Examples

- Linear regression (least squares)

$$f_i(x) = (a_i^T x - b_i)^2$$

a_i, b_i are data

- Logistic regression (classification)

$$f_i(x) = \log\left(\frac{1}{1+\exp(y_i a_i^T x)}\right)$$

a_i are data, y_i labels

Assumptions

Lipschitz constant

- *L*-smoothness

$$f_i(x+h) \leq f_i(x) + \langle f_i(x), h \rangle + \frac{L}{2}\|h\|^2$$

Strong convexity constant

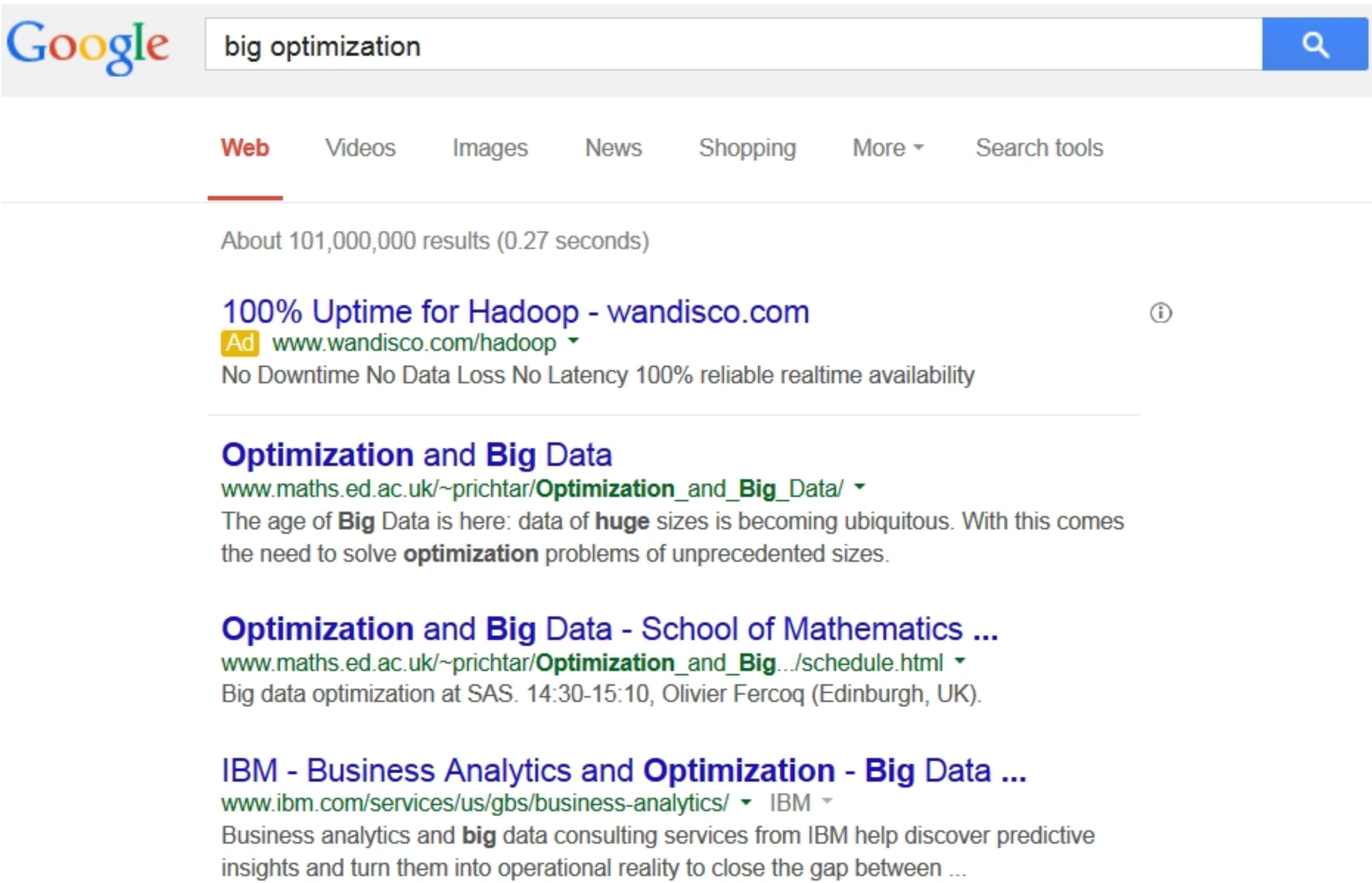
- Strong convexity

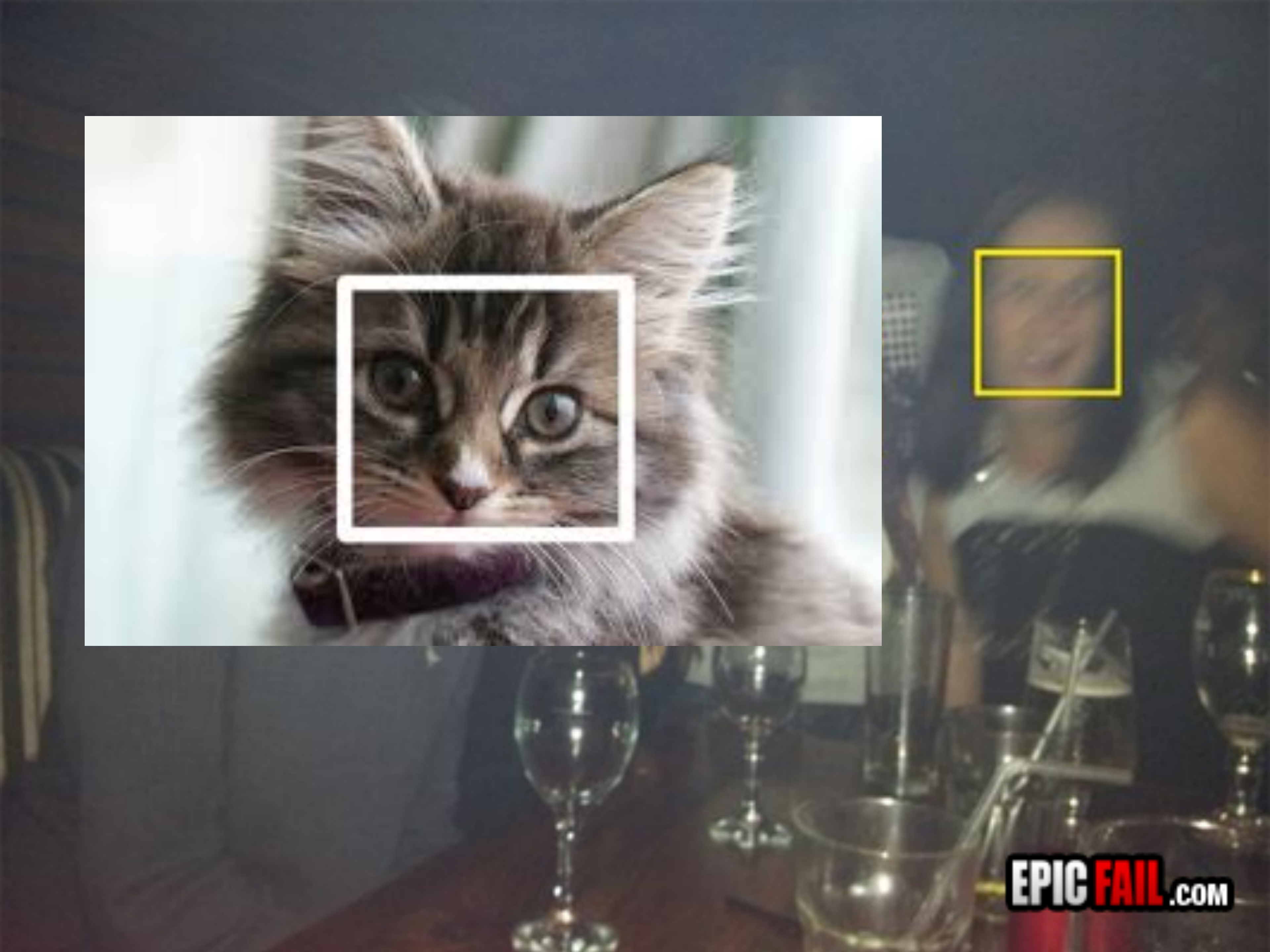
$$F(x+h) \geq F(x) + \langle F(x), h \rangle + \frac{\mu}{2}\|h\|^2$$

Applications

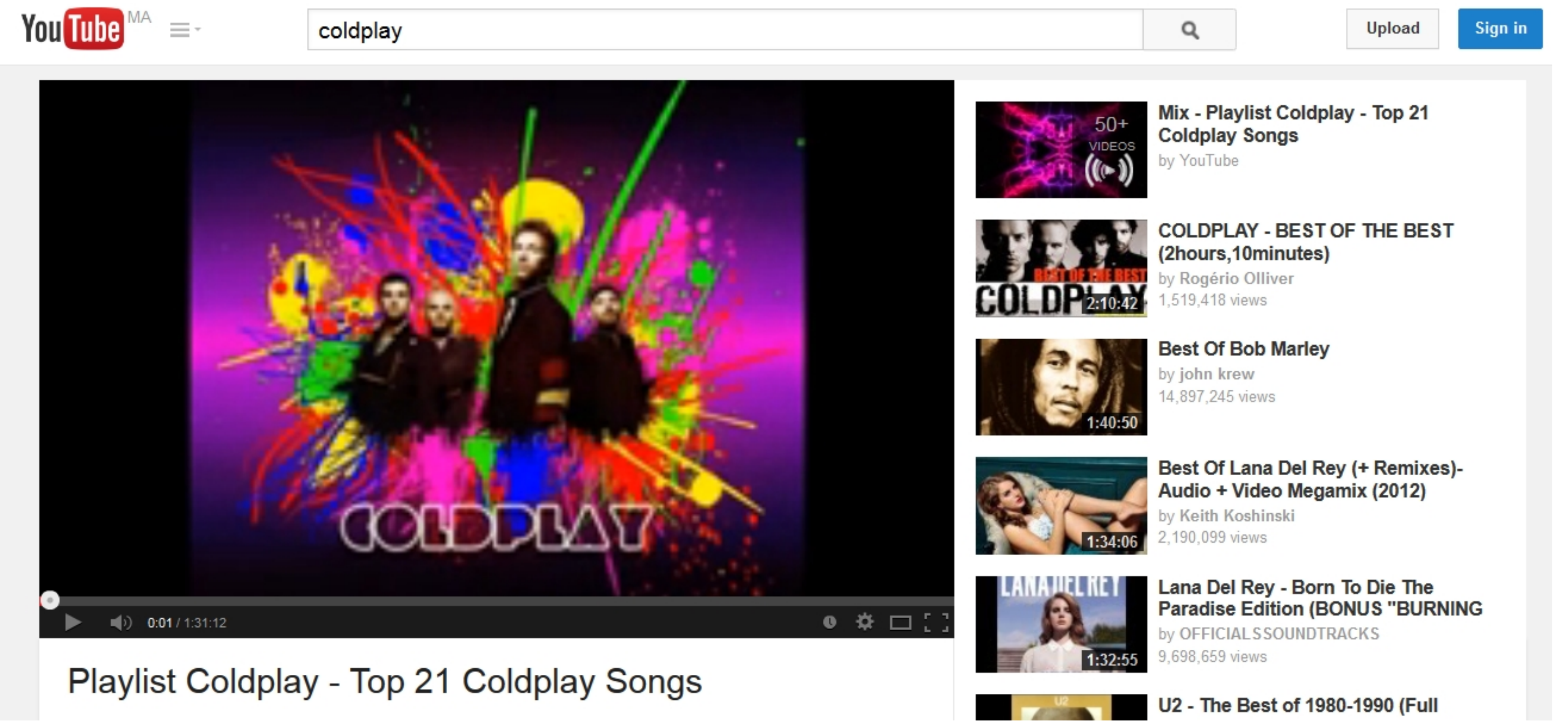

STOP
SPAM

Page Rank


Google
big optimization
Web
Videos
Images
News
Shopping
More
Search tools
About 101,000,000 results (0.27 seconds)
100% Uptime for Hadoop - wandisco.com
Ad
www.wandisco.com/hadoop
No Downtime No Data Loss No Latency 100% reliable realtime availability
Optimization and Big Data
www.maths.ed.ac.uk/~prichtar/Optimization_and_Big_Data/
The age of Big Data is here: data of huge sizes is becoming ubiquitous. With this comes the need to solve optimization problems of unprecedented sizes.
Optimization and Big Data - School of Mathematics ...
www.maths.ed.ac.uk/~prichtar/Optimization_and_Big.../schedule.html
Big data optimization at SAS. 14:30-15:10, Olivier Fercoq (Edinburgh, UK).
IBM - Business Analytics and Optimization - Big Data ...
www.ibm.com/services/us/gbs/business-analytics/
IBM
Business analytics and big data consulting services from IBM help discover predictive insights and turn them into operational reality to close the gap between ...


EPIC FAIL.COM

Recommender Systems



Geotagging One Hundred Million Twitter Accounts with Total Variation Minimization

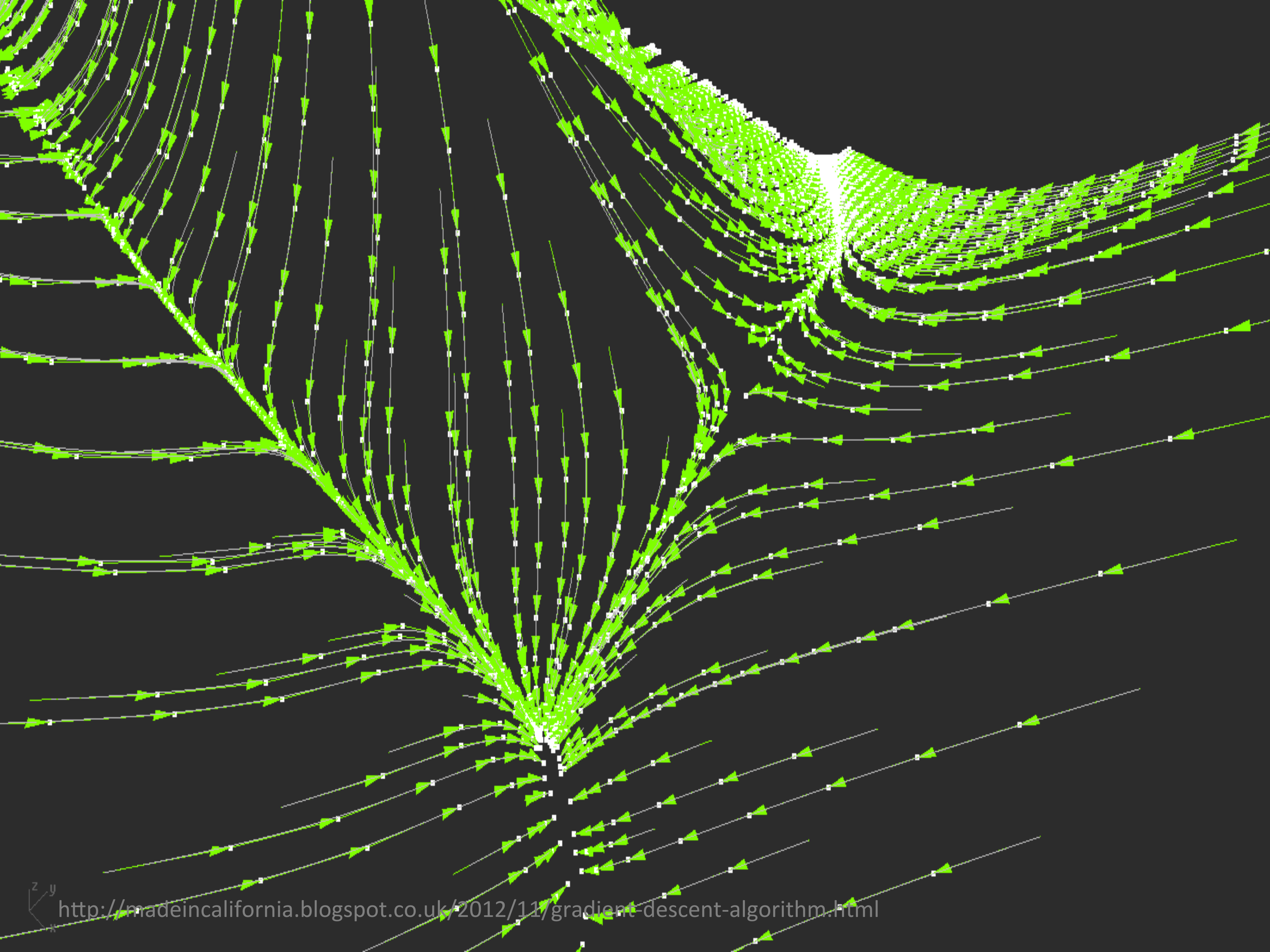
Ryan Compton, David Jurgens, David Allen

(Submitted on 28 Apr 2014)

Geographically annotated social media is extremely valuable for modern information retrieval. However, when researchers can only access publicly-visible data, one quickly finds that social media users rarely publish location information. In this work, we provide a method which can geolocate the overwhelming majority of active Twitter users, independent of their location sharing preferences, using only publicly-visible Twitter data.

Our method infers an unknown user's location by examining their friend's locations. We frame the geotagging problem as an optimization over a social network with a total variation-based objective and provide a scalable and distributed algorithm for its solution. Furthermore, we show how a robust estimate of the geographic dispersion of each user's ego network can be used as a per-user accuracy measure, allowing us to discard poor location inferences and

GD vs SGD



http://madeincalifornia.blogspot.co.uk/2012/11/gradient-descent-algorithm.html

Gradient Descent (GD)

- Update rule:

$$x_{k+1} = x_k - \frac{1}{L}\nabla F(x_k)$$

- Complexity:

$$\mathcal{O}\left(\frac{L}{\mu}\log(1/\epsilon)\right)$$

iterations

- Cost of a single iteration: n

stochastic gradient evaluations

Stochastic Gradient Descent (SGD)

stepsize

- Update rule:

$$x_{k+1} = x_k - h_k \nabla f_i(x_k)$$

$\mathbb{E}[\nabla f_i(x)] = \nabla F(x)$

i = chosen uniformly at random

- Complexity:

$$\mathcal{O}\left(\frac{L}{\mu}\frac{1}{\epsilon}\right)$$

- Cost of a single iteration: 1

stochastic gradient evaluations

Dream...

GD

Fast convergence

~~Expensive iterations~~

SGD

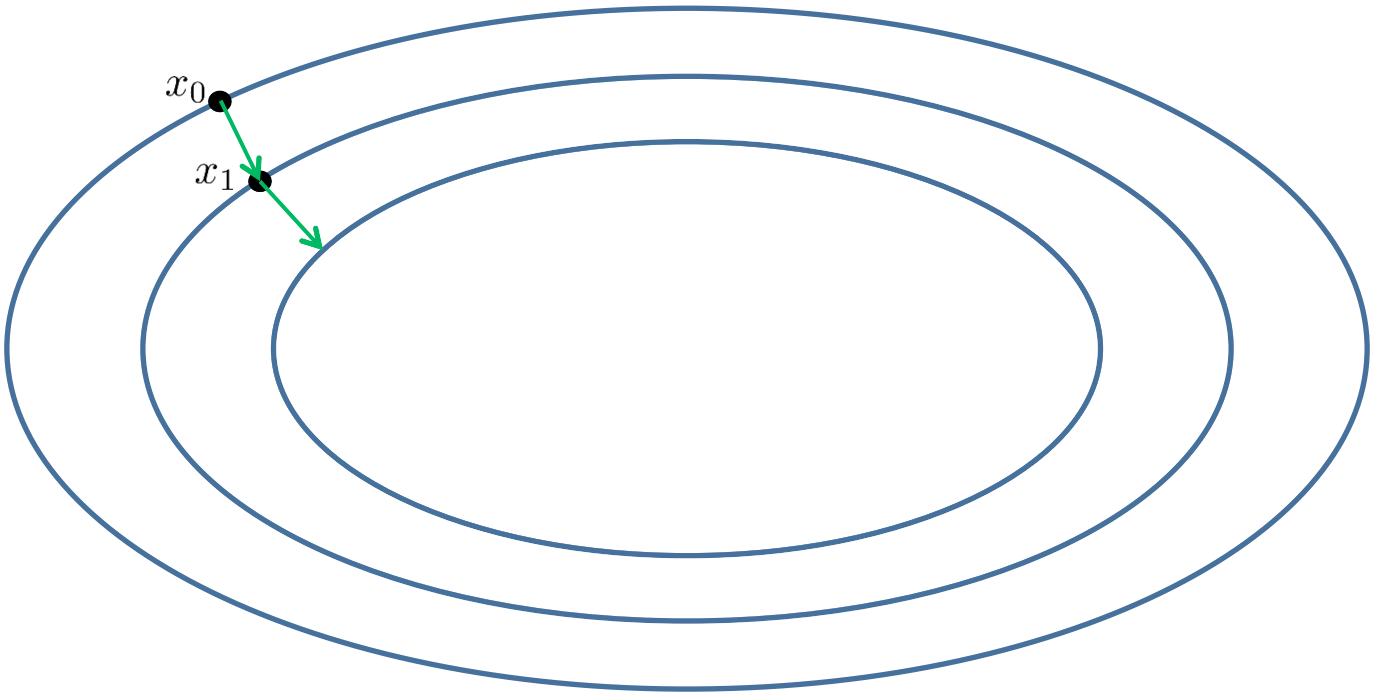
~~Slow convergence~~

Cheap iterations

Combine the good stuff in a single algorithm

S2GD:
Semi-Stochastic Gradient Descent

Intuition



Gradient does not change drastically…

Can recycle older information?

Gradient Approximation

$$x \approx \tilde{x}$$

$$\nabla F(x) = \nabla F(x) - \nabla F(\tilde{x}) + \nabla F(\tilde{x})$$

Gradient change
We can try to estimate

Already computed gradient

$$\nabla F(x) \approx \nabla f_i(x) - \nabla f_i(\tilde{x}) + \nabla F(\tilde{x})$$

The S2GD Algorithm

for $t = 0$ to $m - 1$ **do**
  Pick $i \in \{1, 2, \ldots, n\}$, uniformly at random
  $x \leftarrow x - h\left(\nabla f_i(x) - \nabla f_i(\tilde{x}_j) + \nabla F(\tilde{x}_j)\right)$
end for
$\tilde{x}_{j+1} \leftarrow x$

Simplification. Size of the inner loop (*m*) is random in theory, following a geometric rule.

$$\nabla F(x) \approx \nabla f_i(x) - \nabla f_i(\tilde{x}) + \nabla F(\tilde{x})$$

Complexity

Theorem: Convergence Rate

$$c = \underbrace{\frac{(1-\mu h)^m}{(1-(1-\mu h)^m)(1-2Lh)}} + \underbrace{\frac{2(L-\mu)h}{1-2Lh}}$$

For any fixed h, can be made arbitrarily small by increasing m

Can be made arbitrarily small, by decreasing h

$$\Rightarrow \quad \mathbb{E}\left[\frac{F(\tilde{x}_j)-F(x_*)}{F(\tilde{x}_0)-F(x_*)}\right] \leq c^j$$

How to set the parameters j, h, m ?

Setting the Parameters

$$\mathbb{E}\left[\frac{F(\tilde{x}_j)-F(x_*)}{F(\tilde{x}_0)-F(x_*)}\right] \leq \epsilon$$

Target error tolerance

This is achieved by setting the parameters as:

# of outer iterations → $j = \lceil \log(1/\epsilon) \rceil$

stepsize → $h = \frac{1}{(2+4e)L}$

# of inner iterations → $m = 43\kappa$

Total complexity (# stochastic gradient evaluations):

$$j(n + 43\kappa) = \mathcal{O}\left[(n+\kappa)\log(1/\epsilon)\right]$$

outer iters

full gradient evaluations

m inner iterations

Complexity of GD vs S2GD

- S2GD complexity

$$\mathcal{O}\left[(n+\kappa)\log(1/\epsilon)\right]$$

- GD complexity

$$\mathcal{O}\left[(n\kappa)\log(1/\epsilon)\right]$$

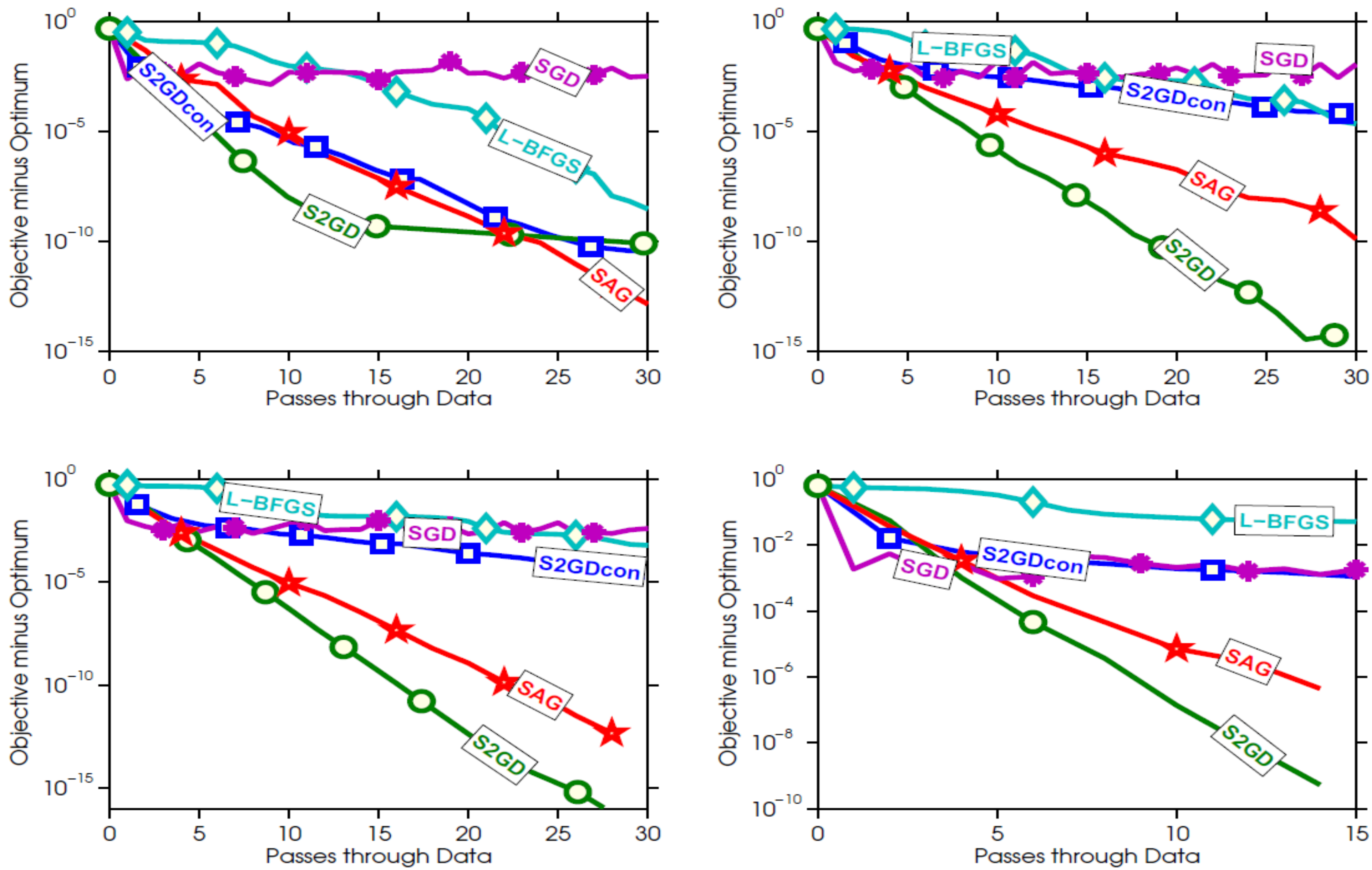
$$\mathcal{O}(n)$$

Cost of 1 iteration

$$\mathcal{O}\left[\kappa\log(1/\epsilon)\right]$$

iterations

Experiment (logistic regression on: ijcnn, rcv, real-sim, url)



Related Methods

- SAG: Stochastic Average Gradient
  (Mark Schmidt, Nicolas Le Roux, Francis Bach, 2013)
  - Refresh single stochastic gradient in each iteration
  - Need to store *n* gradients
  - Similar convergence rate
  - Cumbersome analysis
- SAGA (Aaron Defazio, Francis Bach, Simon Lacoste-Julien, 2014)
  - Refined analysis
- MISO: Minimization by Incremental Surrogate Optimization (Julien Mairal, 2014)
  - Similar to SAG, slightly worse performance
  - Elegant analysis

Related Methods

- SVRG: Stochastic Variance Reduced Gradient
  (Rie Johnson, Tong Zhang, 2013)
  - Arises as a special case in S2GD
- Prox-SVRG
  (Tong Zhang, Lin Xiao, 2014)
  - Extended to proximal setting
- EMGD: Epoch Mixed Gradient Descent
  (Lijun Zhang, Mehrdad Mahdavi , Rong Jin, 2013)
  - Handles simple constraints
  - Worse convergence rate: $\mathcal{O}\left[(n+\kappa^2)\log(1/\epsilon)\right]$

Extensions

Extensions

- Constraints [Prox-SVRG]
- Proximal setup [Prox-SVRG]
- Mini-batching [mS2GD]
- Efficient handling of sparse data [S2GD]
- Pre-processing with SGD [S2GD]
- Optimal choice of parameters [S2GD]
- Weakly convex functions [S2GD]
- High-probability result [S2GD]
- Inexact computation of gradients

S2CD: Semi-Stochastic Coordinate Descent

Algorithm 1 Semi-Stochastic Coordinate Descent (S2CD)

---

parameters: m (max # of stochastic steps per epoch); $h > 0$ (stepsize parameter); $x_0 \in \mathbb{R}^d$ (starting point)
for $k = 0, 1, 2, \ldots$ **do**
  Compute and store $\nabla f(x_k) = \frac{1}{n} \sum_i \nabla f_i(x_k)$
  Initialize the inner loop: $y_{k,0} \leftarrow x_k$
  Choose random length of the inner loop: let $t_k = T \in \{1, 2, \ldots, m\}$ with probability $(1 - \mu h)^{m-T} / \beta$
  for $t = 0$ to $t_k - 1$ **do**
    Pick coordinate $j \in \{1, 2, \ldots, d\}$,with probability p_j
    Pick function index i from the set $\{i \ : \ L_{ij} > 0\}$ with probability q_{ij}
    Update the j^{th} coordinate: $y_{k,t+1} \leftarrow y_{k,t} - hp_j^{-1} \left(\nabla_j f(x_k) + \frac{1}{nq_{ij}} (\nabla_j f_i(y_{k,t}) - \nabla_j f_i(x_k)) \right) e_j$
  end for
  Reset the starting point: $x_{k+1} \leftarrow y_{k,t_k}$
end for

Complexity: $\mathcal{O}\left(nC_{grad} + \hat{\kappa}C_{cd}\right) \log(1/\epsilon)$

S2GD: $\mathcal{O}\left(nC_{grad} + \kappa C_{grad}\right) \log(1/\epsilon)$

mS2GD: S2GD with Mini-batching

Algorithm 1 mS2GD

Input: m (max # of stochastic steps per epoch); $h > 0$ (stepsize); $x_0 \in \mathbb{R}^d$ (starting point); minibatch size $b \in [n]$
for $k = 0, 1, 2, \dots$ **do**
    Compute and store $g_k \leftarrow \nabla f(x_k) = \frac{1}{n} \sum_i \nabla f_i(x_k)$
    Initialize the inner loop: $y_{k,0} \leftarrow x_k$
    Let $t_k \leftarrow t \in \{1, 2, \dots, m\}$ with probability q_t given by (6)
    for $t = 0$ to $t_k - 1$ **do**
        Choose mini-batch $A_{kt} \subset [n]$ of size b, uniformly at random
        Compute a stoch. estimate of $\nabla f(y_{k,t})$: $v_{k,t} \leftarrow g_k + \frac{1}{b} \sum_{i \in A_{kt}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_k))$
        $y_{k,t+1} \leftarrow \text{prox}_{hR}(y_{k,t} - h v_{k,t})$
    end for
    Set $x_{k+1} \leftarrow y_{k,t_k}$
end for

Sparse Data

- For linear/logistic regression, gradient copies sparsity pattern of the example:

$$f_i(x) = \phi_i(a_i^T x)$$

$$\nabla f_i(x) = a_i^T \nabla \phi_i(u), \quad u = a_i^T x$$

- But the update direction is fully dense

$$\nabla f_i(x) - \nabla f_i(\tilde{x}) + \nabla F(\tilde{x})$$

- Can we do something about it?

S2GD: Implementation for Sparse Data

parameters: m = max # of stochastic steps per epoch, h = stepsize, ν = lower bound on μ

for $j = 0, 1, 2, \dots$ **do**

$g_j \leftarrow \frac{1}{n}\sum_{i=1}^{n} f_i'(x_j)$

$y_{j,0} \leftarrow x_j$

$\chi_i \leftarrow 0$ for $i = 1, 2, \dots, n$ ▷ Store when a coordinate was updated last time

Let $t_j \leftarrow t$ with probability $(1-\nu h)^{m-t}/\beta$ for $t = 1, 2, \dots, m$

for $t = 0$ to $t_j - 1$ **do**

Pick $i \in \{1, 2, \dots, n\}$, uniformly at random

for $s \in \mathrm{nnz}(a_i)$ **do**

$(y_{j,t})_s \leftarrow (y_{j,t})_s - (t - \chi_s)h(g_j)_s$ ▷ Update what will be needed

$\chi_s = t$

end for

$y_{j,t+1} \leftarrow y_{j,t} - h\left(f_i'(y_{j,t}) - f_i'(x_j)\right)$ ▷ A sparse update

end for

for $s = 1$ to d **do** ▷ Finish all the "lazy" updates

$(y_{j,t_j})_s \leftarrow (y_{j,t_j})_s - (t_j - \chi_s)h(g_j)_s$

end for

$x_{j+1} \leftarrow y_{j,t_j}$

end for

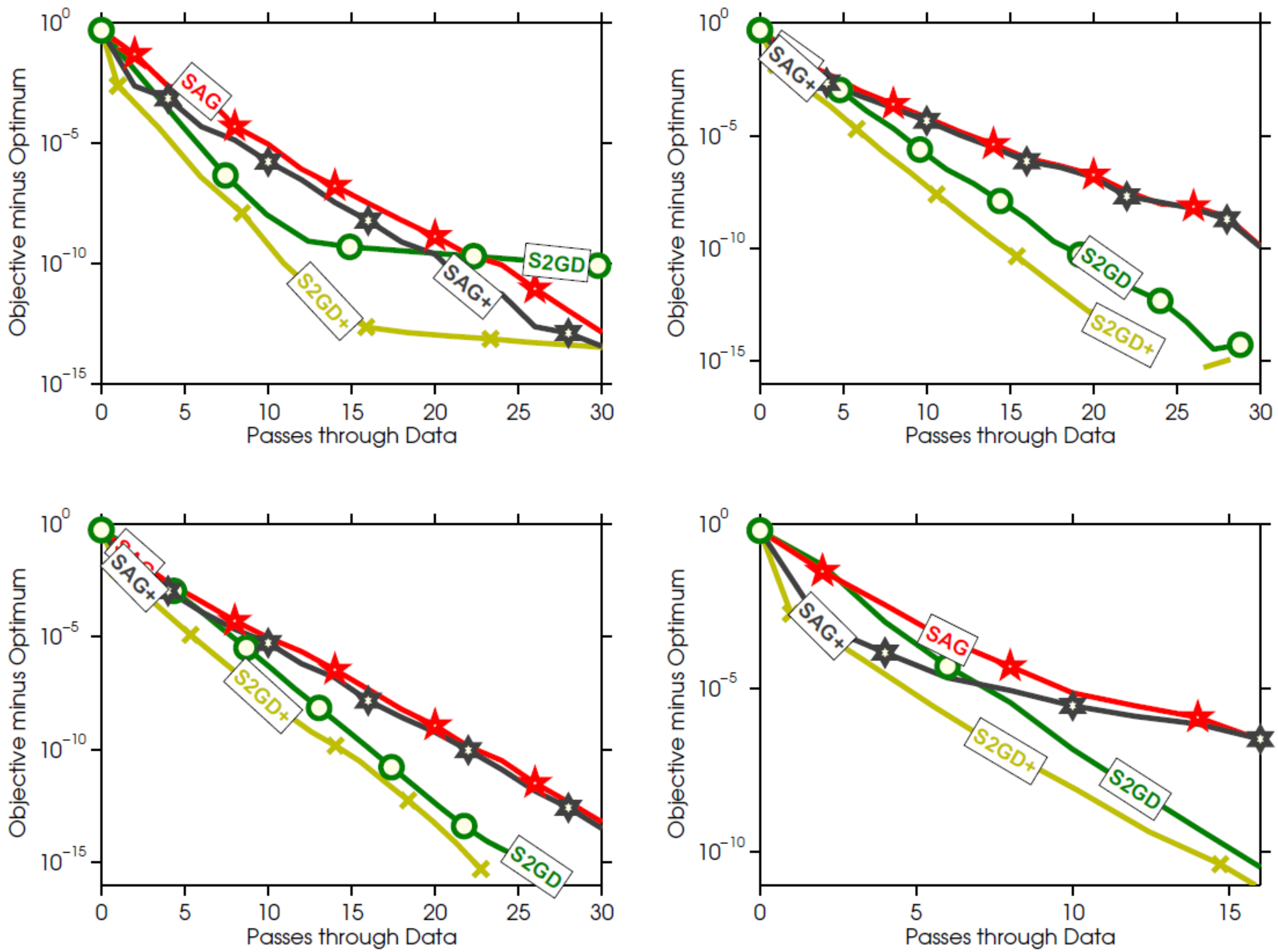
S2GD+

- Observing that SGD can make reasonable progress while S2GD computes the first full gradient, we can formulate the following algorithm:

S2GD+

- Run one pass of SGD
- Use the output as the starting point of S2GD
- Run S2GD

S2GD+ Experiment



High Probability Result

- The result holds only in expectation
- Can we say anything about the concentration of the result in practice?
- For any

Paying just a logarithmic cost

$$k \geq \frac{\log\left(\frac{1}{\epsilon\rho}\right)}{\log\left(\frac{1}{c}\right)}$$

we have:

$$\mathbb{P}\left(\frac{F(x_k) - F(x_*)}{F(x_0) - F(x_*)} \leq \epsilon\right) \geq 1 - \rho$$

Code

Efficient implementation for logistic regression available at MLOSS

http://mloss.org/software/view/556/

THE END