NAIS Centre for Numerical Algorithms and Intelligent Software

Mathematics for Vast Digital Resources

Simons Institute for the Theory of Computing

The University of Edinburgh

Semi-Stochastic Gradient Descent

Peter Richtárik (joint work with Jakub Konečný)

Introduction to Research in Data Science
Edinburgh - October 27, 2014

The Problem

Minimizing Average Loss

- Problems are often structured

Structure – sum of functions

n is BIG

$$\text{find } x_* = \arg\min_{x \in \mathbb{R}^d} f(x) \quad \left[= \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right]$$

$f_i(x)$ represents loss incurred on i^{th} training example

- Frequently arising in machine learning

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 continuity of the gradient of $f_i(\cdot)$

Lipschitz parameter – L

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

- Strong convexity of $f(\cdot)$

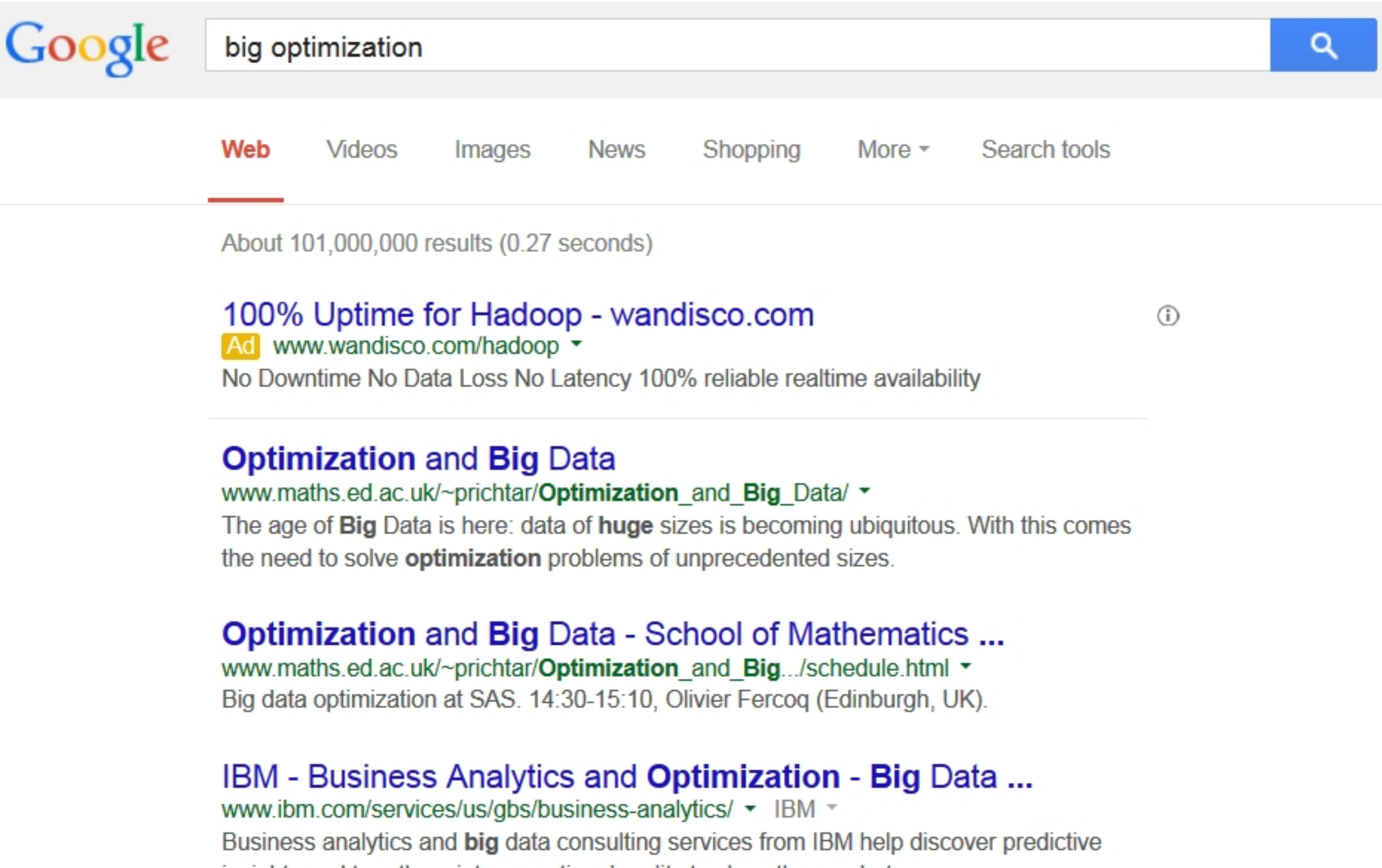
$$f(z) \geq f(x) + \langle \nabla f(x), z - x \rangle + \frac{\mu}{2} \|z - x\|^2$$

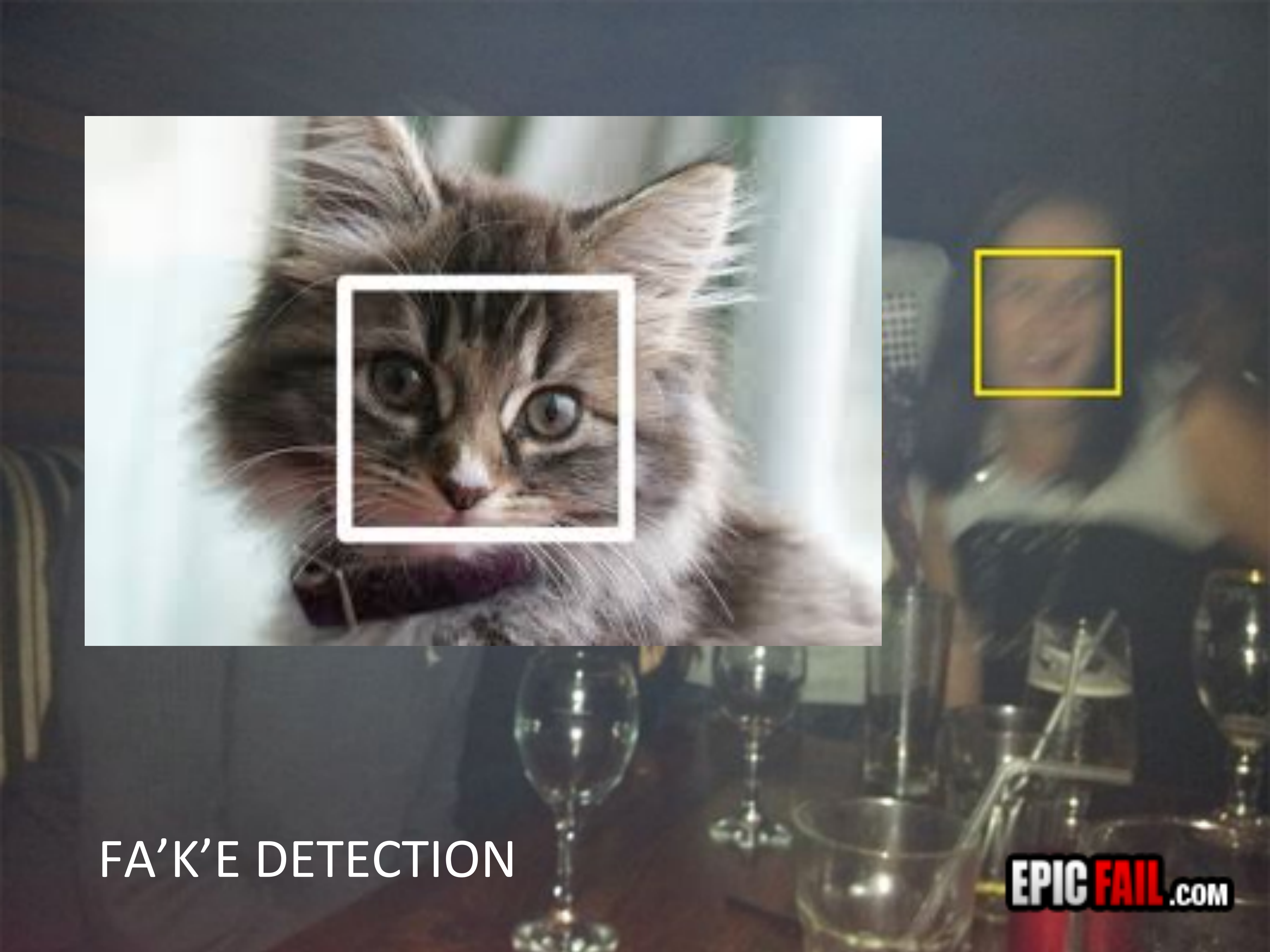
μ – modulus of strong convexity

Applications

SPAM DETECTION

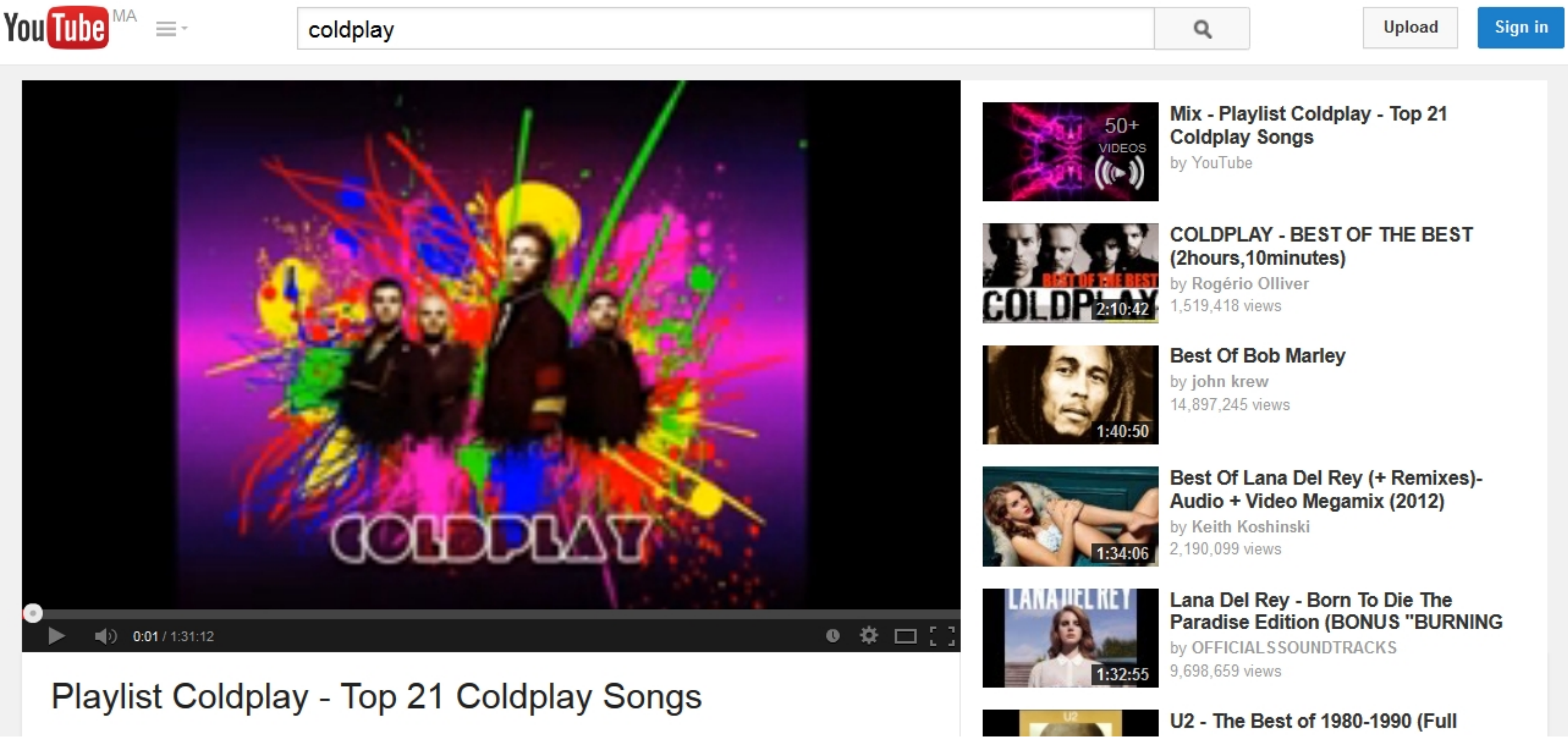
PAGE RANKING





FA’K’E DETECTION

RECOMMENDER SYSTEMS



GEOTAGGING

Cornell University Library

arXiv.org > cs > arXiv:1404.7152

Search or Ar

Computer Science > Social and Information Networks

Geotagging One Hundred Million Twitter Accounts with Total Variation Minimization

Ryan Compton, David Jurgens, David Allen

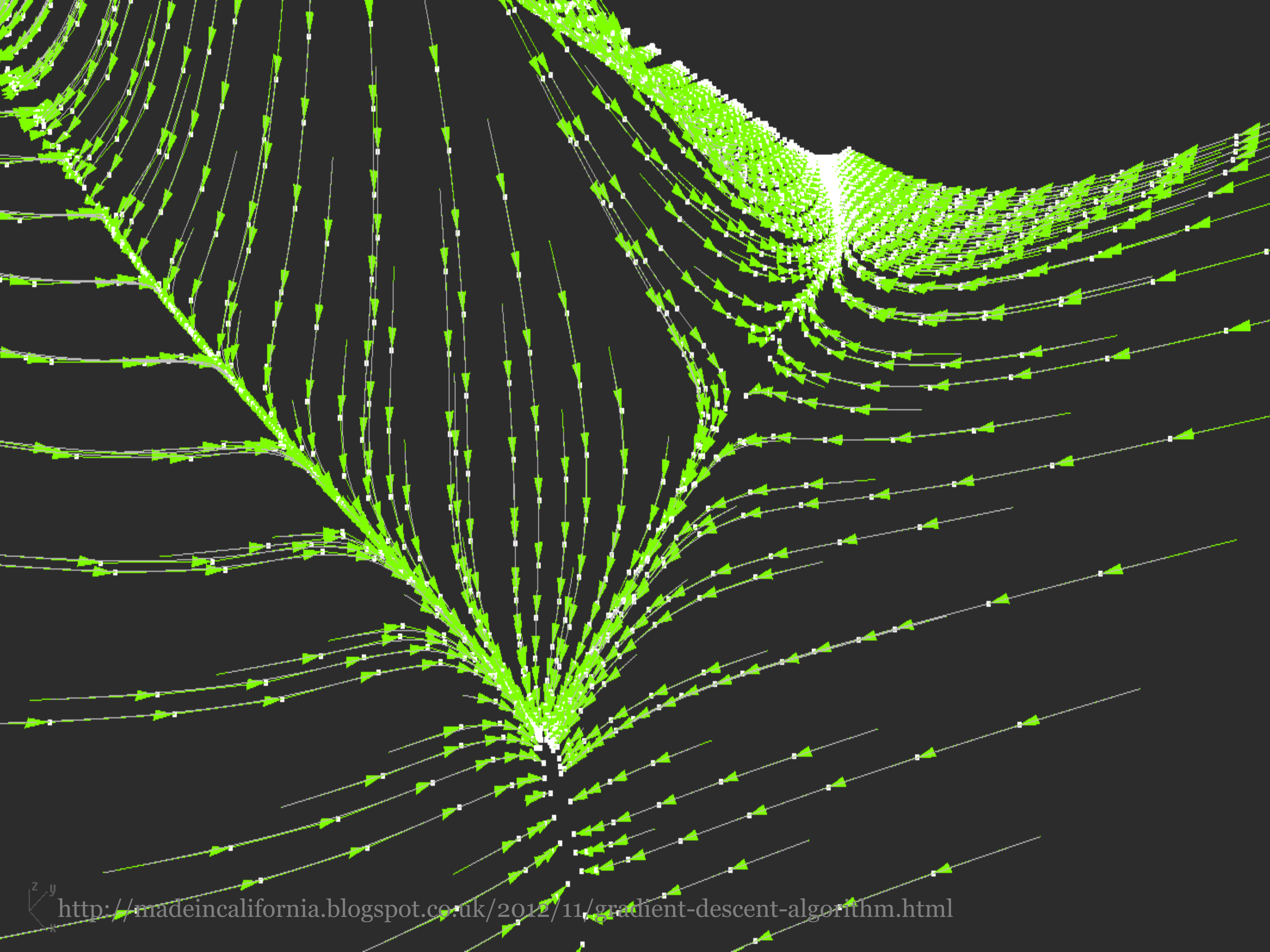
(Submitted on 28 Apr 2014)

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 control the overall error of our approach.

Leave-many-out evaluation shows that our method is able to infer location for 101,846,236 Twitter users at a median error of 6.33 km, allowing us to geotag roughly 89\% of public tweets.

Gradient Descent
vs
Stochastic Gradient Descent


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

Gradient Descent (GD)

- Update rule

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

- Fast convergence rate

$$f(x_k) - f(x_*) \leq \mathcal{O}\left((1 - \tfrac{\mu}{L})^k\right)$$

  - Alternatively, for ϵ accuracy we need

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

- Complexity of single iteration: n

(measured in gradient evaluations)

Stochastic Gradient Descent (SGD)

- Update rule

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

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

a step-size parameter

- Why it works

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

- Slow convergence

$$f(x_k) - f(x_*) \leq \mathcal{O}(1/k), \quad \text{if } h_k = \mathcal{O}(1/k)$$

- Complexity of single iteration – 1 (measured in gradient evaluations)

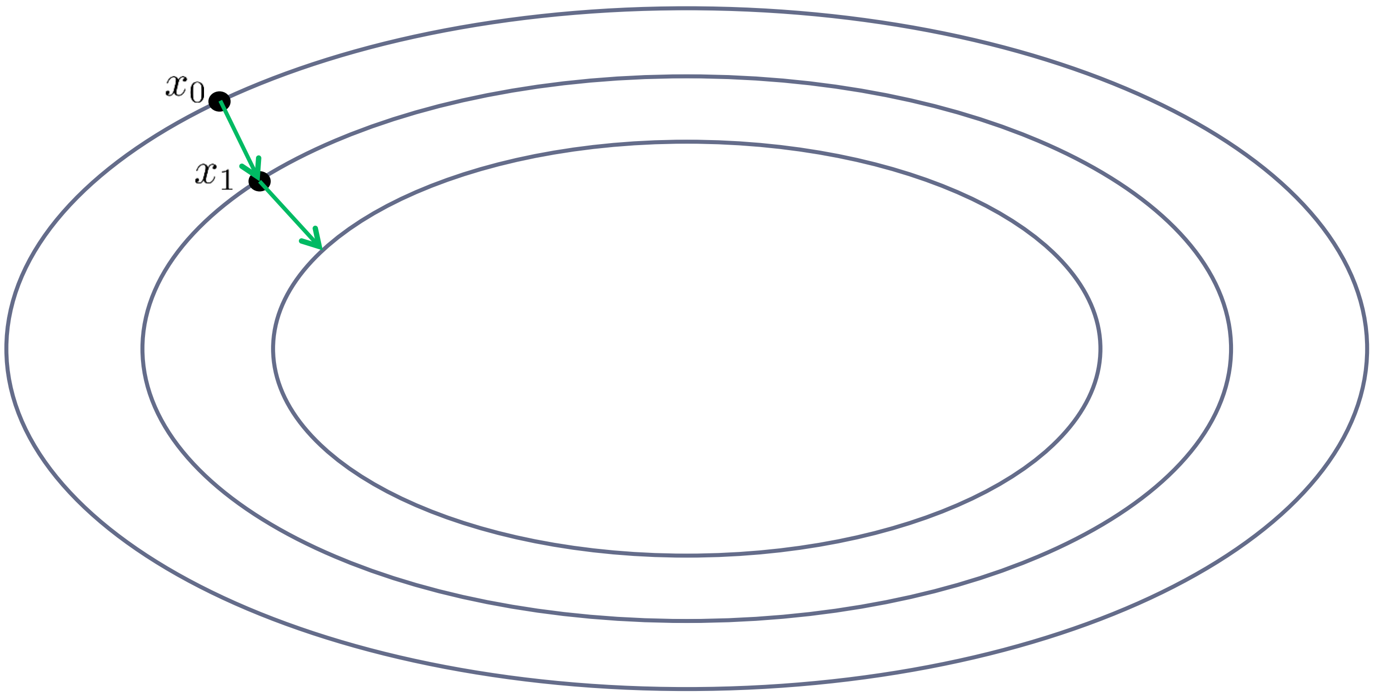
Dream...

GD	**SGD**
Fast convergence	~~Slow convergence~~
~~n gradient evaluations in each iteration~~	Complexity of iteration independent of n

Combine in a single algorithm

S2GD:
Semi-Stochastic Gradient Descent

Why dream may come true…



- The gradient does not change drastically
- We could reuse old information

Modifying "old" gradient

- Imagine someone gives us a "good" point $\tilde{x}$ and $\nabla f(\tilde{x})$

- Gradient at point x, near $\tilde{x}$, can be expressed as

$$\nabla f(x) = \underbrace{\boxed{\nabla f(x) - \nabla f(\tilde{x})}}_{\substack{\text{Gradient change} \\ \text{We can try to estimate}}} + \underbrace{\nabla f(\tilde{x})}_{\text{Already computed gradient}}$$

- Approximation of the gradient

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

The S2GD Algorithm

Simplification; size of the inner loop is random, following a geometric rule

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

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

Theorem

Let the assumptions on f, f_i (L-smoothness, μ-strong convexity) be satisfied. Consider the S2GD algorithm applied to minimization of f. Choose $0 < h < 1/2L$, and m sufficiently large so that

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

Then we have the following convergence in expectation:

$$\mathbb{E}[f(\tilde{x}_j) - f(x_*)] \leq c^j [f(\tilde{x}_0) - f(x_*)]$$

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

- How to set the parameters j, h, m?

Setting the Parameters

$$\frac{\mathbb{E}[f(\tilde{x}_k)-f(x_*)]}{f(\tilde{x}_0)-f(x_*)} \leq \epsilon$$

Fix target accuracy

- The accuracy is achieved by setting

# of epochs $\quad j = \lceil \log(1/\epsilon) \rceil$

stepsize $\quad h = \frac{1}{(2+4e)L}$

# of iterations $\quad m = 43\kappa$

- Total complexity (in gradient evaluations)

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

# of epochs

full gradient evaluation

m cheap iterations

Complexity

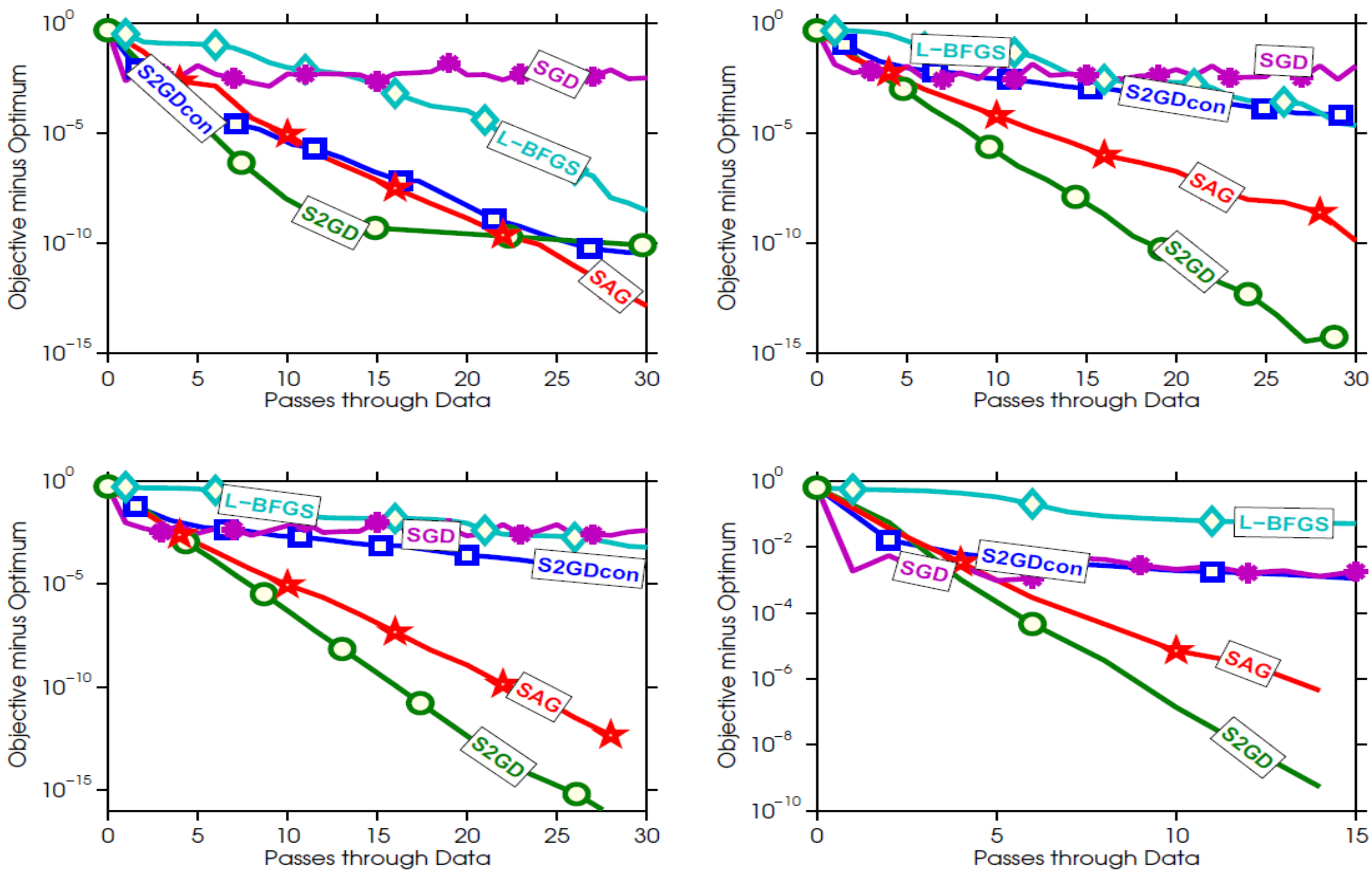
- S2GD complexity

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

- GD complexity
  - $\mathcal{O}\left[\kappa\log(1/\epsilon)\right]$ iterations
  - $\mathcal{O}(n)$ complexity of a single iteration
  - Total

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

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

- Efficient handling of sparse data
- Pre-processing with SGD
- Inexact computation of gradients
- Non-strongly convex losses
- High-probability result
- Mini-batching: mS2GD
  - Konecny, Liu, Richtarik and Takac. mS2GD: Minibatch Semi-Stochastic Coordinate Descent in the Proximal Setting, October 2014
- Coordinate variant: S2CD
  - Konecny, Qu and Richtarik. S2CD: Semi-Stochastic Coordinate Descent, October 2014
- Many more ideas!!! (PhD project)

Sparse Data

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

$$\underbrace{\nabla f_i(x) - \nabla f_i(\tilde{x})}_{\text{SPARSE}} + \underbrace{\nabla f(\tilde{x})}_{\text{DENSE}}$$

- Can we do something about it?

Sparse Data (Continued)

- Yes we can!
- To compute $\nabla f_i(\tilde{x})$, we only need coordinates of x corresponding to nonzero elements of a_i

- For each coordinate j, remember when was it updated last time – χ_j
  - Before computing $\nabla f_i(\tilde{x})$ in inner iteration number k, update required coordinates
  - Step being $(\tilde{x})_j \leftarrow (\tilde{x})_j - h\underbrace{(k - \chi_j)}(\nabla f(x))_j$
  - Compute direction and make a single update

The "old gradient"

Number of iterations when the coordinate was not updated

S2GD: Implementation for Sparse Data

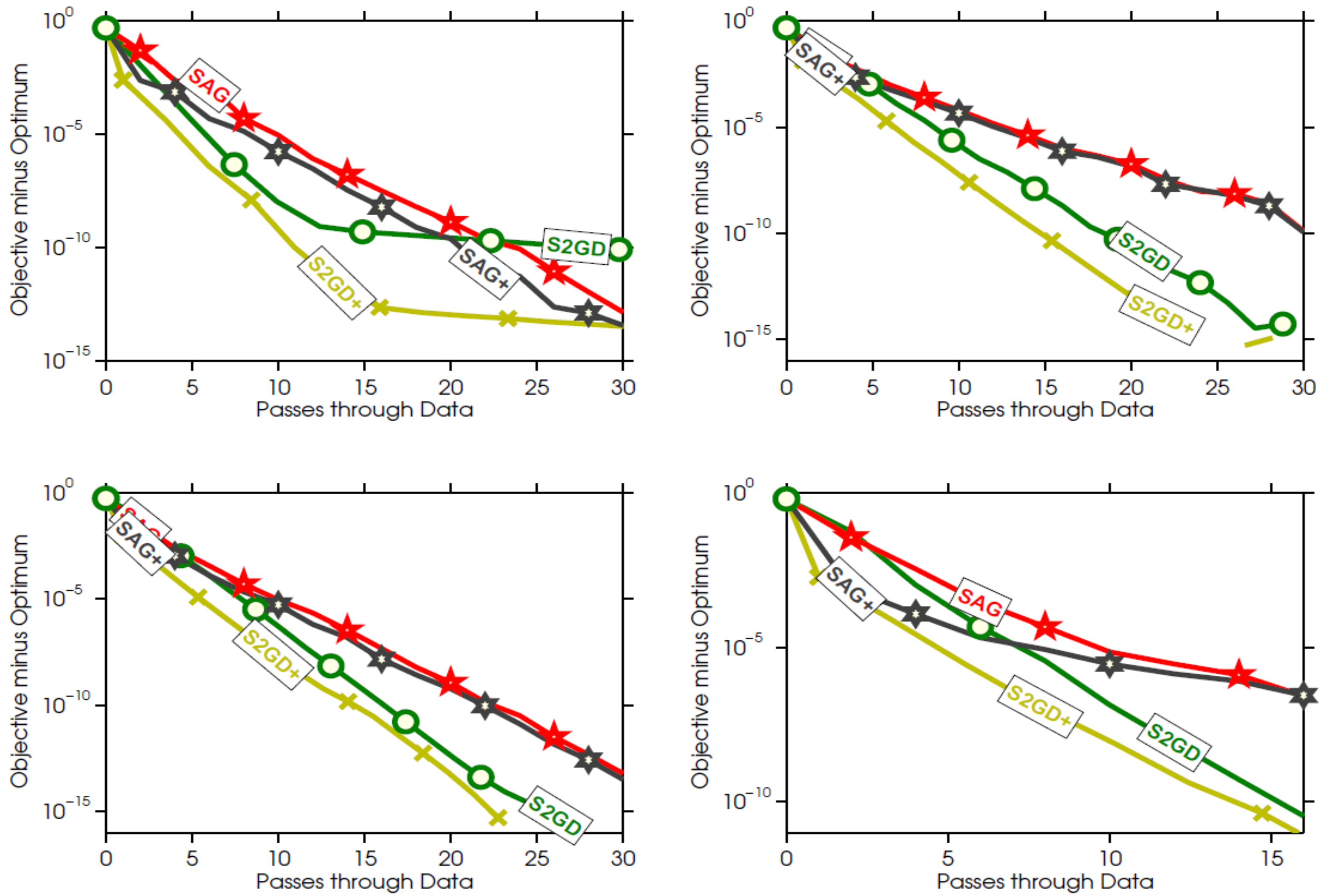
parameters: m = max # of stochastic steps per epoch, h = stepsize,
ν = lower bound on μ
for $j = 0, 1, 2, \dots$ **do**
  $g_j \leftarrow \frac{1}{n}\sum_{i=1}^{n} f_i'(x_j)$
  $y_{j,0} \leftarrow x_j$
  $\chi_i \leftarrow 0$ for $i = 1, 2, \dots, n$ ▷ Store when a coordinate was updated last time
  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 first full gradient (in case we are starting from arbitrary point),
we can formulate the following algorithm (S2GD+)

parameters: $\alpha \geq 1$ (e.g., $\alpha = 1$)
1. Run SGD for a single pass over the data (i.e., n iterations);
output x
2. Starting from $x_0 = x$,
run a version of S2GD in which $t_j = \alpha n$ for all j

S2GD+ Experiment



High Probability Result

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

Paying just logarithm of probability
Independent from other parameters

For any

$$0 < \rho < 1, \quad 0 < \epsilon < 1, \quad j \geq \frac{\log(1/(\epsilon\rho))}{\log(1/c)}$$

we have:

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

Inexact Case

- Question: What if we have access to inexact oracle?
  - Assume we can get the same update direction with error δ:

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

- S2GD algorithm in this setting gives

$$\mathbb{E}\left(f(x_j) - f(x_*)\right) \leq c^j \left(f(x_0) - f(x_*) - \frac{b}{1-c}\right) + \frac{b}{1-c}$$

with

$$c = \frac{(1-\mu h)^m}{(1-(1-\mu h)^m)(1-4Lh)} + \frac{2(L-\mu)h}{1-4Lh}, \quad b = \frac{2\mathbb{E}\|\delta\|^2}{(1-4Lh)h}$$

Code

- Efficient implementation for logistic regression - available at MLOSS

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