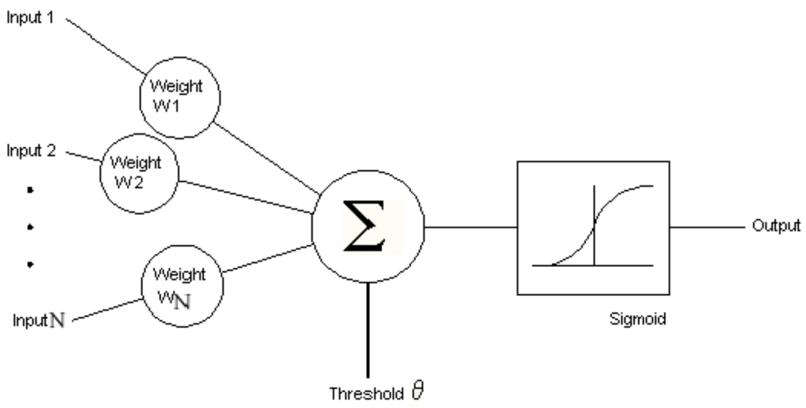
Linear Time Second Order Stochastic Optimization for Machine Learning

Naman Agarwal Brian Bullins Elad Hazan

PRINCETON UNIVERSITY

ML optimization

- Single Neuron:



$$argmin_{w \in R^d} \sum_{i=1 \; to \; m} l(w^T x_i, y_i) + |w|^2$$

- Logistic / Ridge regression, SVMs, Deep Neural Nets

- training set size (m) & dimension of data (d) are **very large**
- Training time measured in **weeks** (speech recognition, images…)

First Order Methods

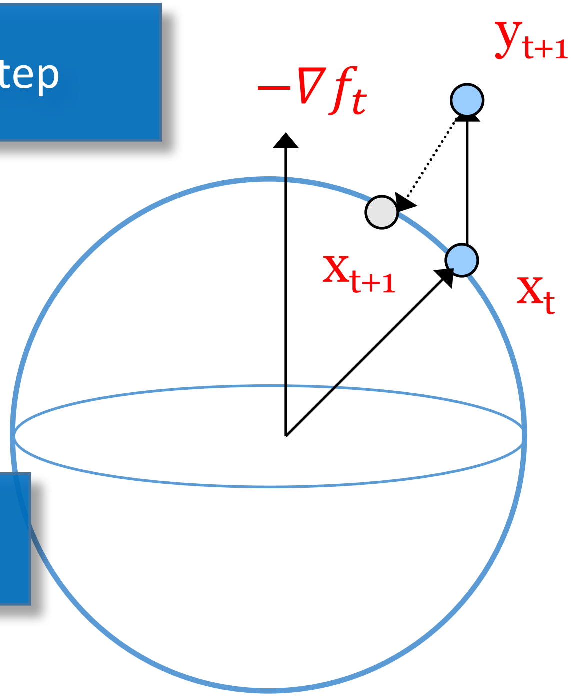
- Gradient Descent

$O(md)$ per step

$$y_t = x_t - \eta \nabla f(x_t)$$

Project

$O(d)$ per step



- Stochastic Gradient Descent
  - Sample f_i randomly and take a step according to $-\nabla f_i$

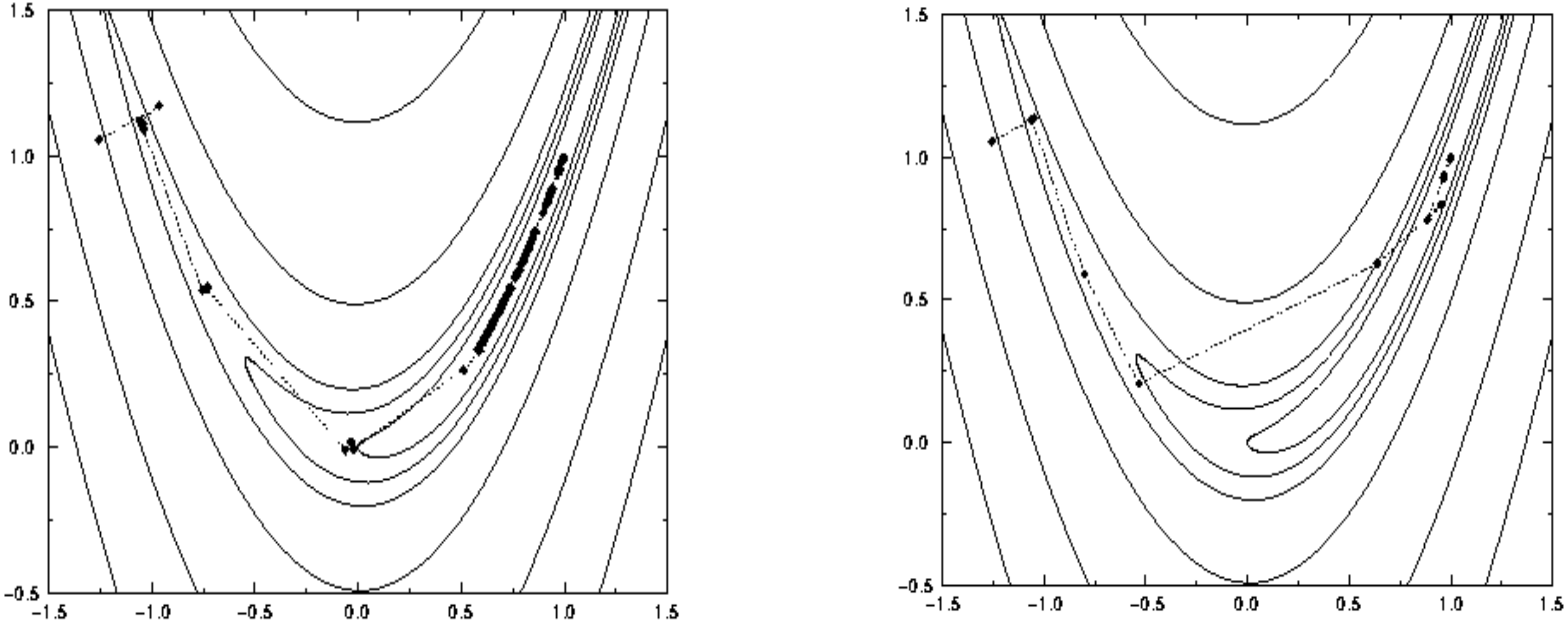
Gradient Descent ++

- Accelerated Gradient Descent (Nesterov)
  - Use momentum from previous steps to take bigger steps
- Adaptive Regularization
  - Adagrad (Duchi, Hazan, Singer), etc…
- Variance Reduction
  - SVRG (Johnson and Zhang; Mahdavi, Zhang,Jin) / SAG,SAGA (Schmidt, LeRoux, Bach)
  - Dual Coordinate Ascent (Shalev-Shwartz & Zhang)

- Are we at the limit ?
- Beyond first order necessary?

Higher Order Optimization

- Gradient Descent – Direction of Steepest Descent
- Second Order Methods – Use Local Curvature

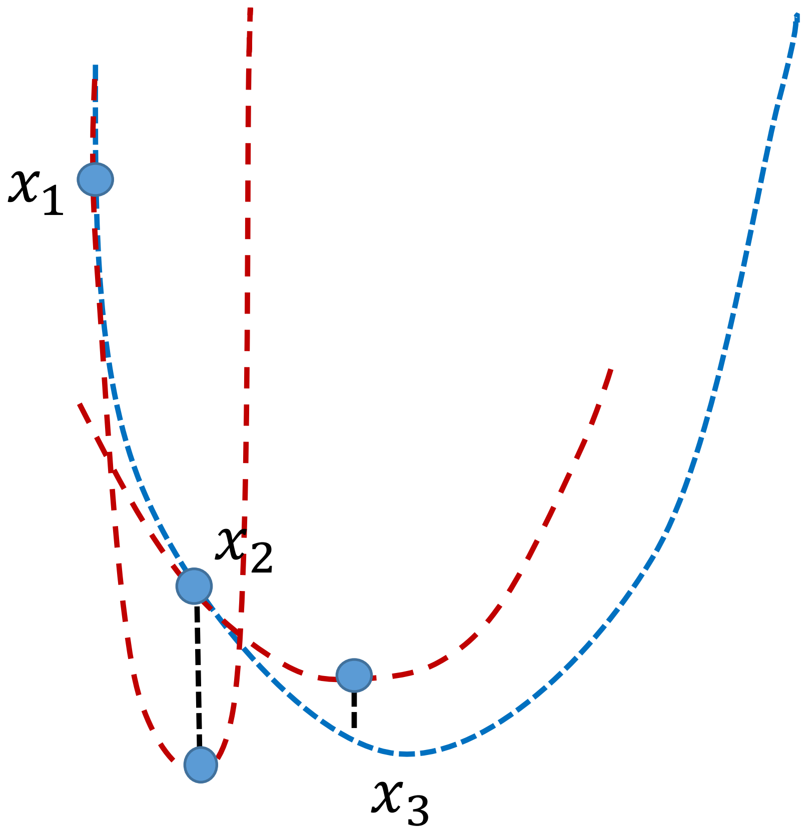


Higher Order Optimization

$$x_{t+1} = x_t - \eta\,[\nabla^2 f(x)]^{-1}\,\nabla f(x)$$

d^3 time per iteration!
Infeasible for ML!!

Till now ☺



Can the matrix inversion be sped-up??

- Spielman-Teng ‘04: diagonally dominant systems of equations in linear time!
  - 2015 Godel prize
  - Used by Daitch-Speilman for faster flow algorithms

- Erdogu-Montanari ‘15: low rank approximation & inversion by Sherman-Morisson
  - Allow stochastic information
  - Still prohibitive: rank * d^2

- Concurrently: Agarwal, Langford, Luo: faster, linear-time low-rank sketching (online setting)

Our results

- **LiSSA**
  - **Natural** Stochastic Newton Method
  - Every iteration in **O(d)** time. Linear in **Input Sparsity**
  - $\mathbf{log}\left(\frac{\mathbf{1}}{\boldsymbol{\epsilon}}\right)$ iterations (Linear Convergence)
  - Better dependence on the **condition number**
  - better local convergence than known FO methods empirically
- **LiSSA ++**
  - **better condition number**
  - Couple with Matrix Sampling/ Sketching techniques - **Best known running time** for $\boldsymbol{m \gg d}$

Overview of Running Times

Algorithm	**Running Time**
Gradient Descent	$md\kappa \log\frac{1}{\epsilon}$
Accelerated Gradient Descent	$md\sqrt{\kappa}\log\frac{1}{\epsilon}$
SVRG/SAGA/SDCA	$\big(m + O(\kappa)\big)d \log\frac{1}{\epsilon}$
LiSSA	$(m + O(\kappa')V)\, d \log\frac{1}{\epsilon}$
Accelerated SDCA / Catalyst	$\left(m + O(\sqrt{\kappa m}\,)\right) d \log\frac{1}{\epsilon}$
LiSSA ++	$O\left(m + O\left(\sqrt{\kappa d}\right)\right) d \log^2\frac{1}{\epsilon}$

Making Second Order Stochastic

- Easy to get an unbiased estimator of the Hessian

$$\widetilde{\nabla^2} f = \nabla^2 f_i \qquad\qquad i \sim uniform\ [1, m]$$

$$E\left[\widetilde{\nabla^2} f\right] = \nabla^2 f$$

- But

$$E\left[\widetilde{\nabla^2} f^{-1}\right] \neq \nabla^2 f^{-1}$$

Nevertheless, used recently in **NewSamp**
(Erdogdu & Montanari)

Our Approach – circumvent inversion!

- The Neumann Series/ Taylor Expansion of the Inverse
- For $M \succcurlyeq 0, || M || \leq 1$

$$M^{-1} = \sum_{i=0 \ to \ \infty} (I \ - M)^i$$

- Thus:

$$E\left[\widetilde{\nabla^2 f}^{-1} \nabla f\right] = \sum_i (I - \nabla^2)^i \nabla \ = \ E_{k \sim N} \prod_{k=1 \ to \ q} \left(I - \widetilde{\nabla_k^2}\right) \nabla$$

For any distribution on naturals $k \sim N$

Vector-vector products only

Improved Estimator

- Previously, Estimate a single term in one estimate
- Recursive Reformulation of the series

$$\boldsymbol{M^{-1} = I + (I - M)(\underbrace{I}_{Ind.\,Sample} + (I - M)(I + \underbrace{\dots\dots}_{to\ S_2})))}$$

$$\widetilde{M_{S_2}^{-1}} = I + (I - M)\underbrace{\widetilde{M_{S_2-1}^{-1}}}_{Recursive\ estimate\ \ \widetilde{M_{S_2-1}^{-1}}}$$

- Truncate after S_2 steps. Typically $S_2 \sim \kappa$
- $E\left[\widetilde{M_{S_2}^{-1}}\right] \to M^{-1}$ as $S_2 \to \infty$
- Repeat and average to reduce the variance

Linear time step

- Need to compute the Newton direction

$$\widetilde{M^{-1}}\,\nabla f = \Big(I + \big(I - \tilde{M}\big)(\,\dots\,)\Big)\nabla \mathrm{f}$$

- Reduces to computing $\tilde{M}\nabla f$ quickly
- O(d) if loss is of the form $l(w^T x, y)$

- Hessian is of the form $-\,\mathrm{g(x, y)}\; x x^T$
  matrix-vector product → vector-vector product
  input sparsity time !!

LiSSA –
Linear-time Second-order Stochastic Algorithm

- Use the estimator $\widetilde{\nabla^{-2} f}$ defined previously
- Compute a full (large batch) gradient $\nabla \mathrm{f}$
- Move in the direction $\widetilde{\nabla^{-2} f}\ \nabla f$
- Start with a few Gradient Descent Steps

Main Theorem – (not ++)

Theorem 1

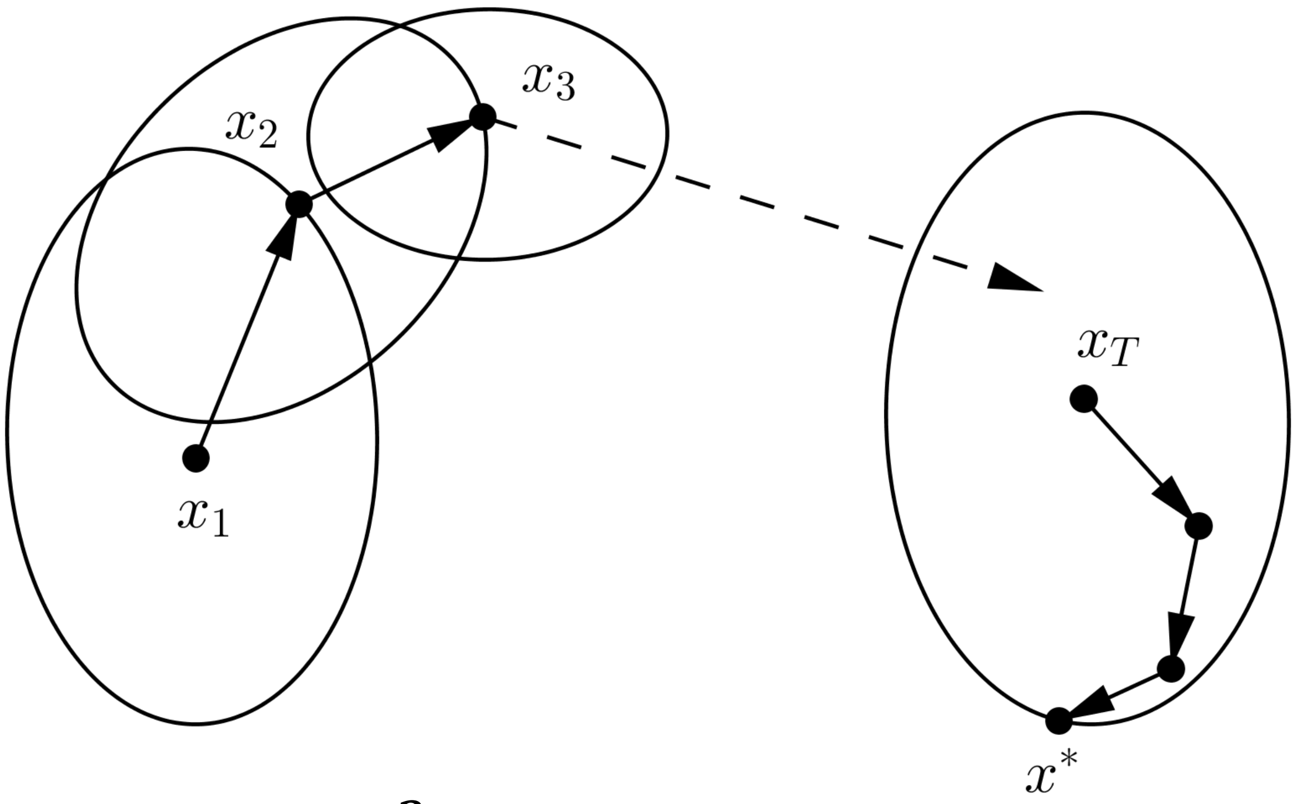
For large t, LiSSA returns a point in the parameter space w_t s.t.

$$f(w_t) \leq \; f(w^*) + \epsilon$$

In total time $\log\left(\frac{1}{\epsilon}\right) \; d \; (m + O(\kappa) \;\; V \;)$

- V is a bound on the variance of the estimator
  - In Practice - a small constant (e.g. 1)
  - In Theory - $V \leq \kappa^2$

Analysis: Newton ~ Iterative Mirrored Descent



Newton: $x_{t+1} \leftarrow x_t - \nabla^{-2} f(x_t) \nabla f(x_t)$

$\nabla R(x_{t+1}) \leftarrow \nabla R(x_t) - \nabla f(x_t)$ for $R(x) = x^T \nabla^2 f(x_t) x$

Local vs Global Condition Number

- For GD++, condition number is defined by the smoothness and strong convexity parameters

$$|| \nabla^2 f(x) || \le \beta$$
$$|| \nabla^2 f(x) || \ge \alpha$$

$$Must\ hold\ \forall x$$

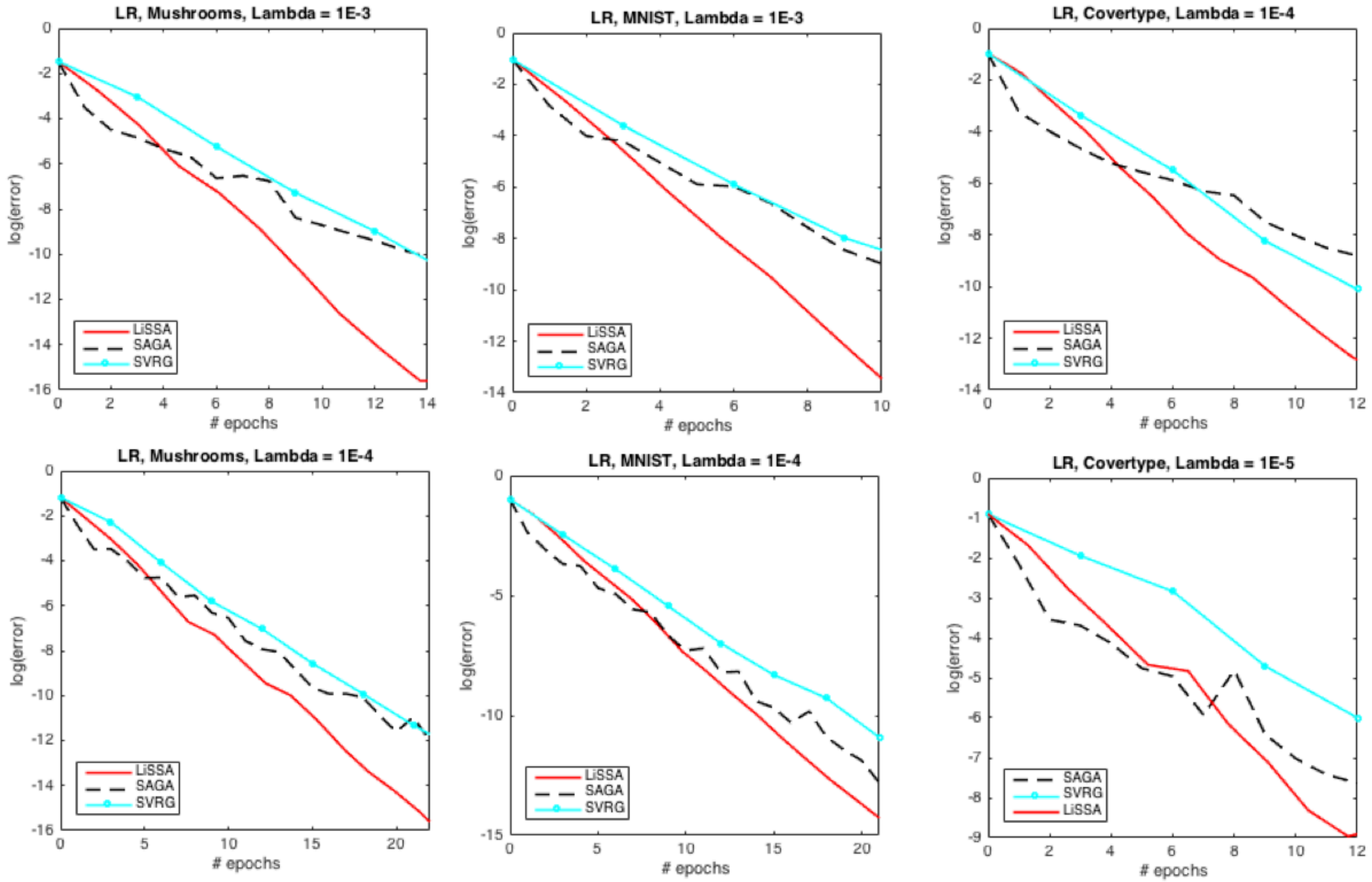
$$\kappa_{global} = \frac{\beta}{\alpha}$$

- For LiSSA, condition number of the local quadratic

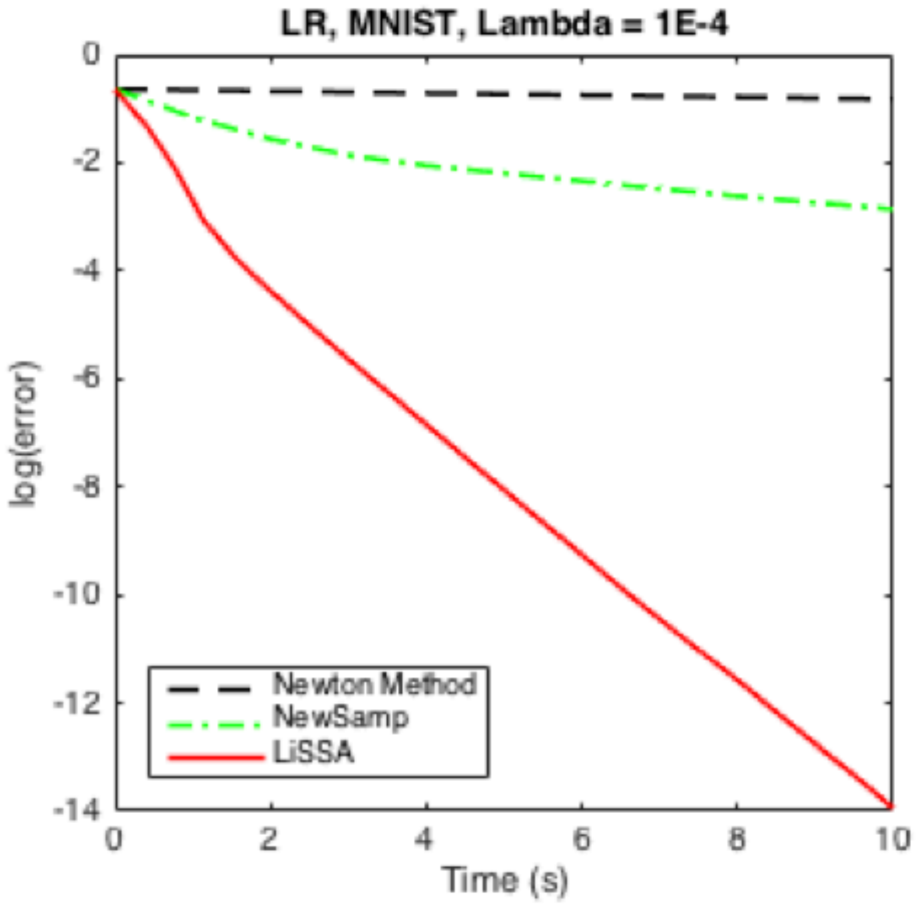
$$\kappa_{local} = \max_x \kappa(\nabla^2 f(x)) \le \kappa_{global}$$

- In practice – A difference factor of up to 10 between $\kappa_{local}\ vs\ \kappa_{global}$

Experimental Results - convex



Comparison b/w Second Order Methods



LiSSA: Alternative Interpretation

- Variance adjusted SGD on the local quadratic approximation
- Quadratic Approximation of $f(x)$ around a point

$$Q(y) = \nabla f(x)^T y + y^T \nabla^2 f(x)\, y$$

Exactly Corresponds to LiSSA

LiSSA ⟺ SGD on a Quadratic

$$\nabla Q(y) = \nabla f(x) + \underbrace{\nabla^2 f(x)}_{estimate} y$$

$$y - \nabla Q(y) = \left(I - \underbrace{\nabla^2 f(x)}_{estimate}\right) y - \underbrace{\nabla f(x)}_{Fixed}$$

Acts as **Variance Reduction**

LiSSA+

- Use FO algorithms (SVRG/SDCA) for intermediate quadratic
- Need to solve each quadratic upto error ϵ
- Total of $\log\log\frac{1}{\epsilon}$ quadratic sub problems
- Total time – $Time(FO)\log\log\frac{1}{\epsilon}$

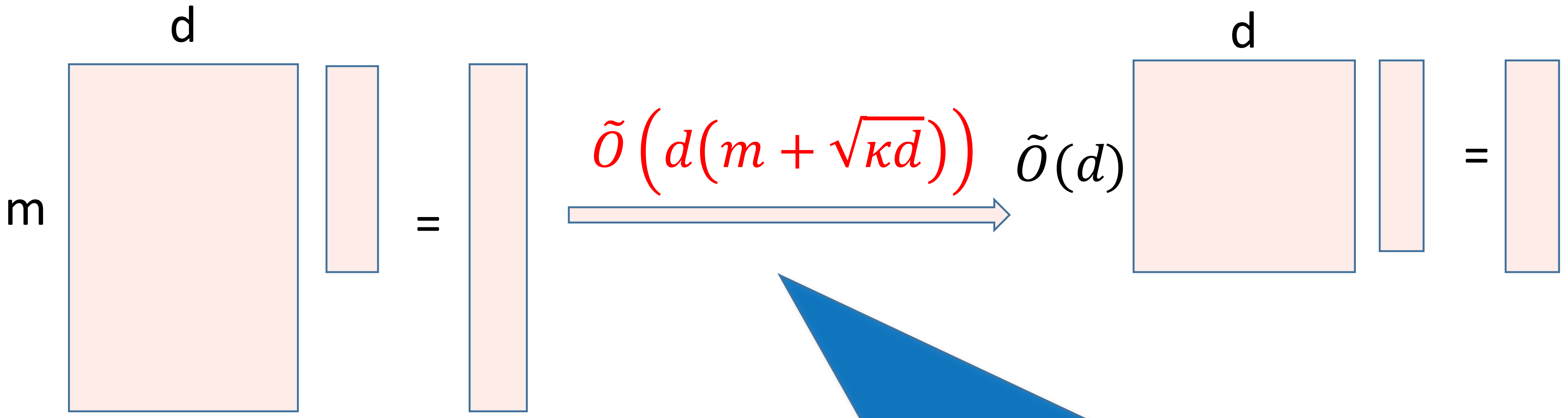
> - Better conditioned than the original
> - Faster running time in practice

LiSSA ++ (Motivation)

- Why run FO on the quadratic approximation when full $f(x)$ is available ?
- Better Conditioning
- Can use special quadratic structure of the problem
- Make use of Matrix Sampling and Sketching Techniques

LiSSA ++ Overview

$$\tilde{O}\left(d\left(d+\sqrt{\kappa d}\right)\right)\log^2\frac{1}{\epsilon}$$



$$\tilde{O}\left(d\left(m+\sqrt{\kappa d}\right)\right)\log^2\frac{1}{\epsilon}$$

- Recursive Uniform Sampling (Lee+Sidford)
- Time – Input + O(1) Linear systems in O($d\log d$)
- Can use FO on those systems

LiSSA++ Theorem

Theorem 2

For large t, LiSSA++ returns a point in the parameter space w_t s.t.

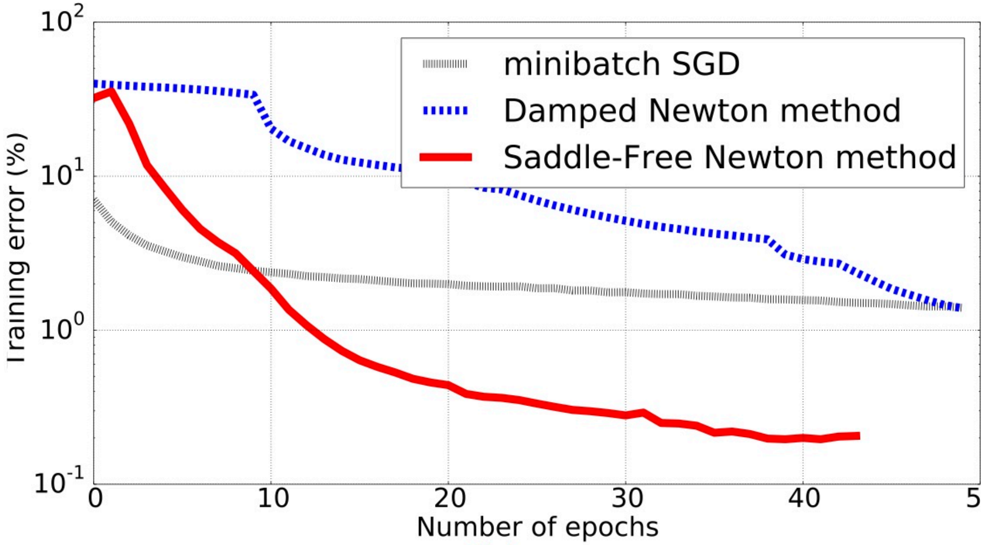
$$f(w_t) \leq f(w^*) + \epsilon$$

In total time $\tilde{O}\left(\log^2\left(\frac{1}{\epsilon}\right) d\left(m + \sqrt{\kappa d}\right)\right)$

- Fastest over all running time for when $\kappa, m \gg d$

2^{nd} order information: new phenomena?

- escape saddle points?
- "Computational lens for deep nets": experiment with 2^{nd} order information:
  - Trust region
  - Cubic regularization
  - Eigenvalue methods….



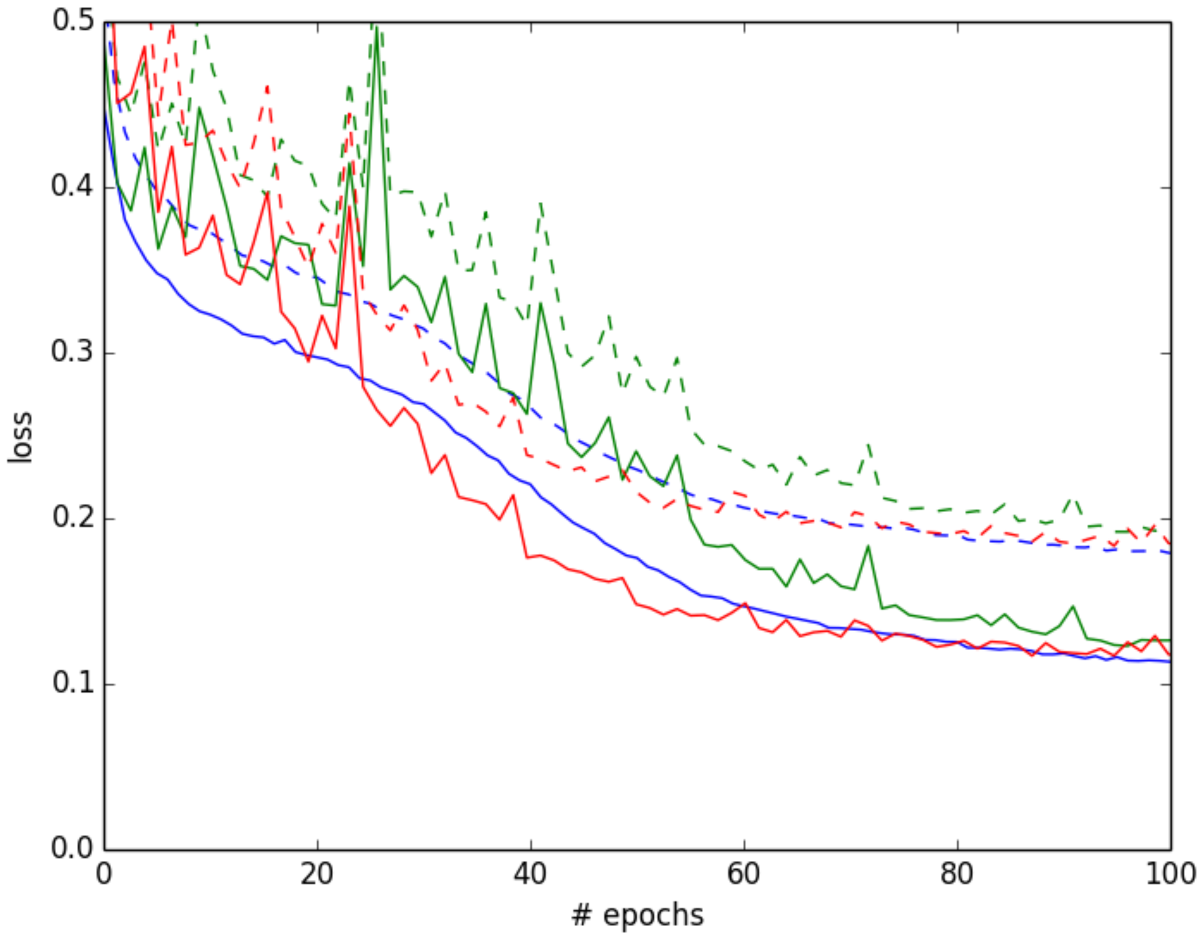
(b)

Bengio-group experiment

Follow-up work (w. Naman, Brian & Tengyu)

- Apply to train deep nets – non-convex optimization
- Can implement variants that are reasonable (trust region etc.) using LiSSA techniques
- Theory works (w. the Pearlmuter trick)
- practice - doesn't improve upon GD++ ☹

Bottou “say what doesn’t work”



Summary

- LiSSA
  - A natural, practical and efficient Second Order Stochastic Algorithm
  - Empirically faster than GD++ on real world datasets
- LiSSA++
  - Fastest running time known theoretically
  - Better dependence on the condition number
- 2nd order information doesn't seem to improve performance for deep nets (so far)

Thank you !