RandNLA and its Applications in Second-order Optimization and Deep Learning

Zhewei Yao

University of California, Berkeley

July 2019

Outline

- **Randomized Numerical Linear Algebra**
- RandNLA in Second-order Optimization
- RandNLA in Deep Learning
- Conclusions

Matrices provide a natural structure with which to **model data.**

- $A \in R^{n \times m}$ can encode information about m objects, each of which is described by n features; etc.
- A positive definite $A \in R^{n \times n}$ can encode the correlations/similarities between all pairs of n objects; etc.

Motivated by data problems, recent years have witnessed **many exciting developments**

- Particularly remarkable is the use of randomization.
  - Massive data
  - Computationally expensive or NP-hard

RandNLA in l2-norm regression

A linear measurement model:

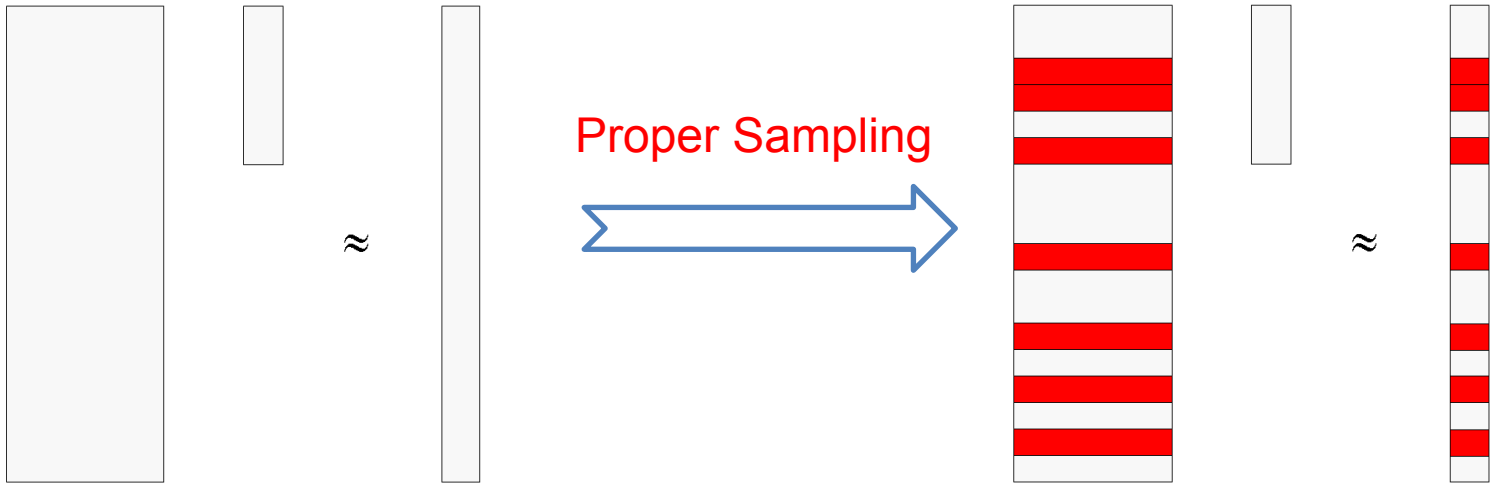
$$y = Ax + \epsilon$$

- y is the measurement
- x is unknown
- ϵ is an error process

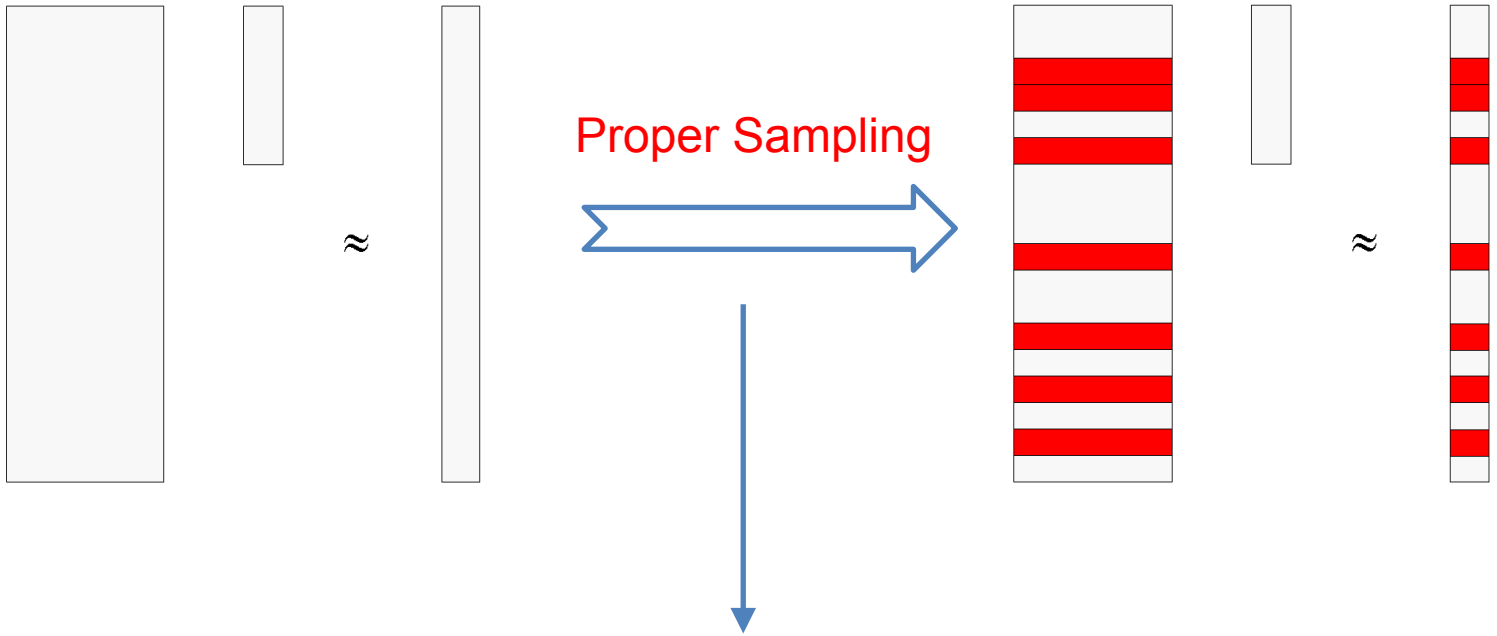
In order to estimate x, solve:

$$\hat{x} = \arg\min \|y - Ax\|$$

RandNLA in l2-norm regression



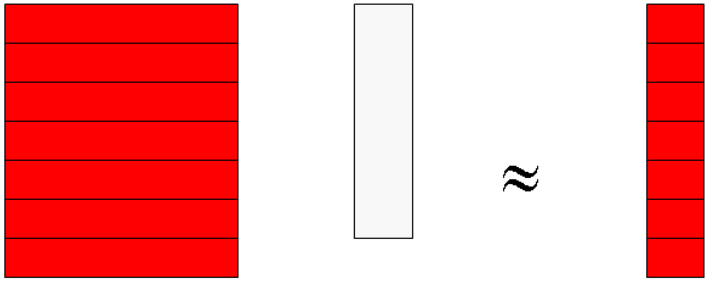
RandNLA in l2-norm regression



° Uniform Sampling
° Leverage Score Sampling

RandNLA in l2-norm regression

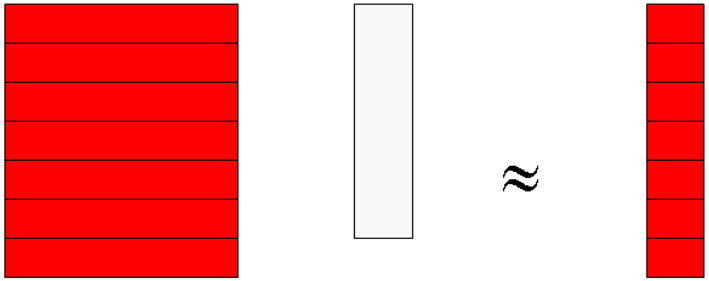
[1, 2] showed that: The run time needed to solve LS reduces from O(mn^2) to O(mnlogn).



Drineas, Mahoney, Muthu, and Sarlo´s. Faster Least Squares Approximation, 2010; [1]
Drineas, Magdon-Ismail, Mahoney, and Woodruff. Fast approximation of matrix coherence and statistical leverage, 2010 [2]

RandNLA in l2-norm regression

[1, 2] showed that: The run time needed to solve LS reduces from O(mn^2) to O(mnlogn).

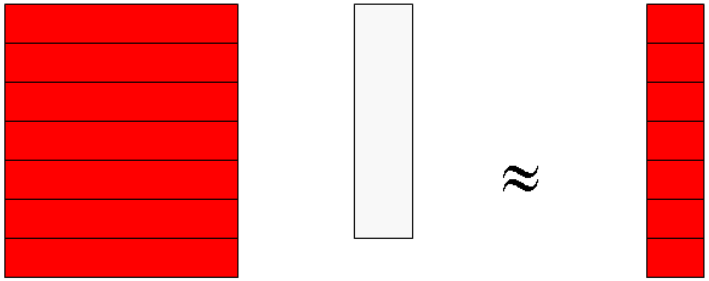


Solution: $(A^T A)^{-1} A^T y \longrightarrow (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \tilde{y}$

Drineas, Mahoney, Muthu, and Sarlo´s. Faster Least Squares Approximation, 2010; [1]
Drineas, Magdon-Ismail, Mahoney, and Woodruff. Fast approximation of matrix coherence and statistical leverage, 2010 [2]

RandNLA in l2-norm regression

[1, 2] showed that: The run time needed to solve LS reduces from O(mn^2) to O(mnlogn).



Solution: $(A^T A)^{-1} A^T y \longrightarrow (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \tilde{y}$

Iterative Second-order Method:

$$x^{(t+1)} = x^{(t)} + H^{-1} g$$

Drineas, Mahoney, Muthu, and Sarlo´s. Faster Least Squares Approximation, 2010; [1]
Drineas, Magdon-Ismail, Mahoney, and Woodruff. Fast approximation of matrix coherence and statistical leverage, 2010 [2]

Outline

° Randomized Numerical Linear Algebra

° **RandNLA in Second-order Optimization**

° RandNLA in Deep Learning

° Conclusions

Sub-sampled second-order optimization

Consider optimizing $F : R^d \to R$:

$$\min_{x \in R^d} F(x)$$

For finite-sum problems in high dimensions, where

$$F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

computing the exact gradient/Hessian requires a pass over the entire data, which can be costly when $n \gg 1$.

Sub-sampled second-order optimization

Consider optimizing $F : R^d \rightarrow R$:

$$\min_{x \in R^d} F(x)$$

For finite-sum problems in high dimensions, where

$$F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

computing the exact gradient/Hessian requires a pass over the entire data, which can be costly when n ≫ 1.

DEFINITION ((ϵ_g, ϵ_H)-OPTIMALITY). Given x, is an (ϵ_g, ϵ_H)-optimal solution if

$$\|\nabla F(x)\| \leq \epsilon_g \quad and \quad \lambda_{\min}(\nabla^2 F(x)) \geq -\epsilon_H .$$

Approximate everything one can approximate

To increase efficiency, incorporate approximations of:

- gradient information, and
- Hessian information, and
- inexact solutions of the underlying sub-problems.

Approximate everything one can approximate

To increase efficiency, incorporate approximations of:

- gradient information, and
- Hessian information, and
- inexact solutions of the underlying sub-problems.

More specifically, we consider the sub-sampled gradient and Hessian as：

$$g \triangleq \frac{1}{|S_g|}\sum_{i\in S_g}\nabla f_i(x), \quad and \quad H \triangleq \frac{1}{|S_H|}\sum_{i\in S_H}\nabla^2 f_i(x)$$

Approximate everything one can approximate

To increase efficiency, incorporate approximations of:

- gradient information, and
- Hessian information, and
- inexact solutions of the underlying sub-problems.

More specifically, we consider the sub-sampled gradient and Hessian as:

$$g \triangleq \frac{1}{|S_g|}\sum_{i\in S_g}\nabla f_i(x), \quad and \quad H \triangleq \frac{1}{|S_H|}\sum_{i\in S_H}\nabla^2 f_i(x)$$

Also consider, at step t, approximate solution of underlying sub-problem:

$$x^{(t+1)} = \min_{x\in D}\{F(x^{(t)}) + (x - x^{(t)})^T g(x^{(t)}) + \frac{1}{2\alpha_t}(x - x^{(t)})^T H(x^t)(x - x^{(t)})\}$$

Non-convex methods

Two methods we considered:

° Trust Region: Classical Method for Non-Convex Problem [Sorensen, 1982, Conn et al., 2000]

$$s^{(k)} = \arg \min_{\|s\| \leq \Delta_k} < s, \nabla F(x^{(k)}) > + \frac{1}{2} < s, \nabla^2 F(x^{(k)}) s >$$

° Cubic Regularization: More Recent Method for Non-Convex Problem [Griewank, 1982, Nesterov et al., 2006, Cartis et al., 2011a, Cartis et al., 2011b]

$$s^{(k)} = \arg \min_{s \in R^d} < s, \nabla F(x^{(k)}) > + \frac{1}{2} < s, \nabla^2 F(x^{(k)}) s > + \frac{\sigma_k}{3} \|s\|^3$$

A structural result for optimization

- To get iteration complexity, previous work required $H^{(t)}$:

$$\|(H^{(t)} - \nabla^2 F(x^{(t)}))s^{(t)}\| \leq \epsilon_H \|s^{(t)}\|^2 \qquad (1)$$

A structural result for optimization

- To get iteration complexity, previous work required $H^{(t)}$:

$$\|(H^{(t)} - \nabla^2 F(x^{(t)}))s^{(t)}\| \leq \epsilon_H \|s^{(t)}\|^2 \qquad (1)$$

- Stronger than Dennis-More:

$$\lim_{t \to \infty} \frac{\|(H^{(t)} - \nabla^2 F(x^{(t)}))s^{(t)}\|}{\|s^{(t)}\|} \to 0 \qquad (2)$$

A structural result for optimization

- To get iteration complexity, previous work required $H^{(t)}$:

$$\|(H^{(t)} - \nabla^2 F(x^{(t)}))s^{(t)}\| \leq \epsilon_H \|s^{(t)}\|^2 \tag{1}$$

- Stronger than Dennis-More:

$$\lim_{t \to \infty} \frac{\|(H^{(t)} - \nabla^2 F(x^{(t)}))s^{(t)}\|}{\|s^{(t)}\|} \to 0 \tag{2}$$

We can relax (1) to

$$\|(H^{(t)} - \nabla^2 F(x^{(t)}))s^{(t)}\| \leq \epsilon_H \|s^{(t)}\| \tag{3}$$

Allowing a large body of RandNLA sketching results.

Yao, Xu, Roosta-Khorasani, and Mahoney. Inexact Non-Convex Newton-Type Methods, 2018.

Key result qua RandNLA

Approximate gradient and inexact Hessian, at each step t, must satisfy:

Condition (Gradient and Hessian Approximation Error)

For some $0 < \delta_g,\ \delta_H < 1$, the approximations of gradient and Hessian t^{th} iteration satisfy,

$$\|g_t - \nabla F(x_t)\| \leq \delta_g \approx \mathcal{O}(\epsilon_g),$$

$$\|H_t - \nabla^2 F(x_t)\| \leq \delta_H \approx \mathcal{O}(\epsilon_H),$$

Lemma (Sampling Complexity)

In order to satisfy the above condition, the sampling sizes are:

$$|S_g| \geq \mathcal{O}(\frac{1}{\delta_g^2}) \quad and \quad |S_H| \geq \mathcal{O}(\frac{1}{\delta_H^2})$$

Theoretical Results

Optimal complexity of TR:
Trust region algorithm terminates after at most

$$T \in \mathcal{O}(\max\{\epsilon_g^{-2}\epsilon_H^{-1}, \epsilon_H^{-3}\}),$$

iterations.

Optimal complexity of ARC:
Cubic regularization algorithm terminates after at most

$$T \in \mathcal{O}(\max\{\epsilon_g^{-2}, \epsilon_H^{-3}\}),$$

Iterations.

The complexity of our methods is the same as the original proposed method!

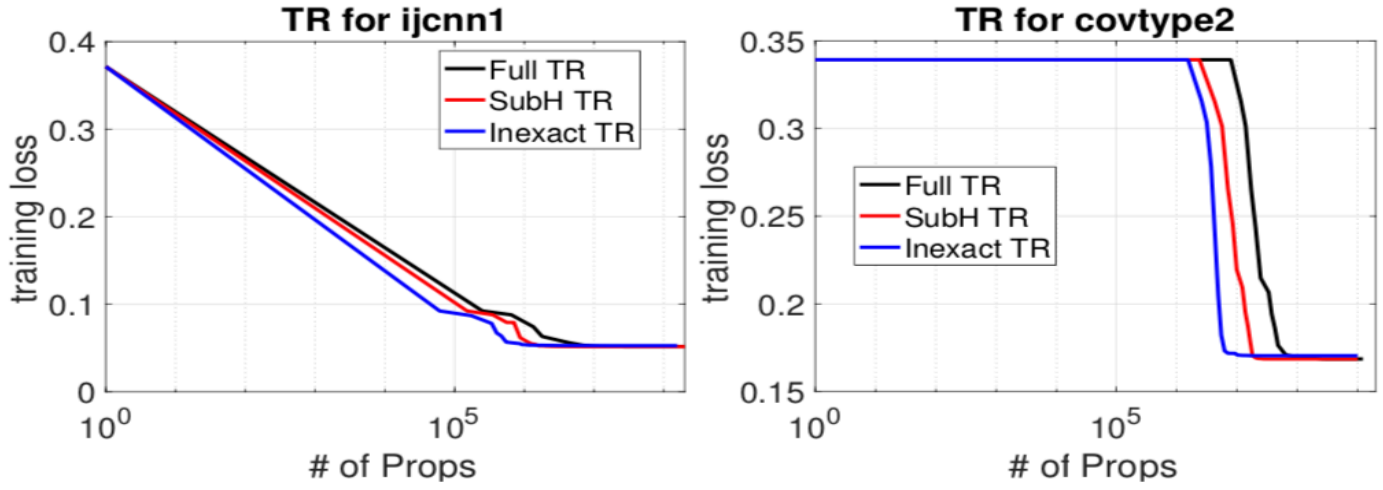
Numerical Results

We evaluate our methods in the context of simple, yet illustrative, nonlinear least squares arising from the task of binary classification with squared loss.

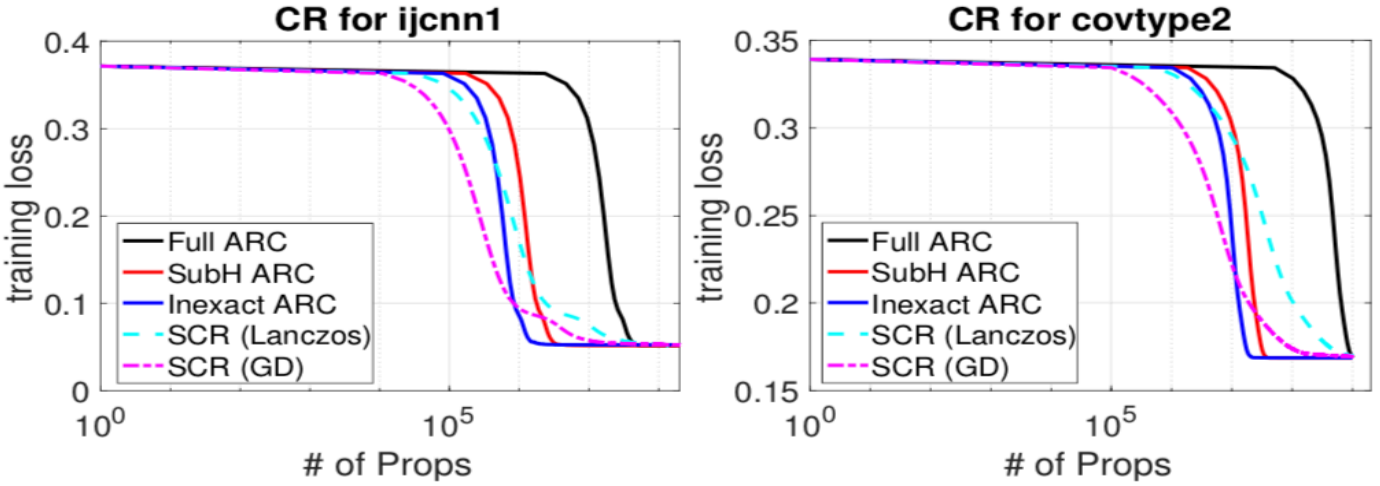
Table Datasets for Binary Classification.

DATA	Training Size (n)	# Features (d)
`covertype`	$464,810$	54
`ijcnn1`	$49,990$	22

Numerical Results



(a) Comparison between variants of TR algorithms



(b) Comparison between variants of CR algorithms

Figure 1 **Performance of various methods on `ijcnn1` and `covertype` for binary linear classification. The x-axis is drawn on the logarithmic scale.**

Outline

° Randomized Numerical Linear Algebra

° RandNLA in Second-order Optimization

° **RandNLA in Deep Learning**

° Conclusions

High Level Outline

° DNN design requires training on large datasets

- Time consuming
- Need fast training -> parallelization -> large batch

° Large batch training does not work:

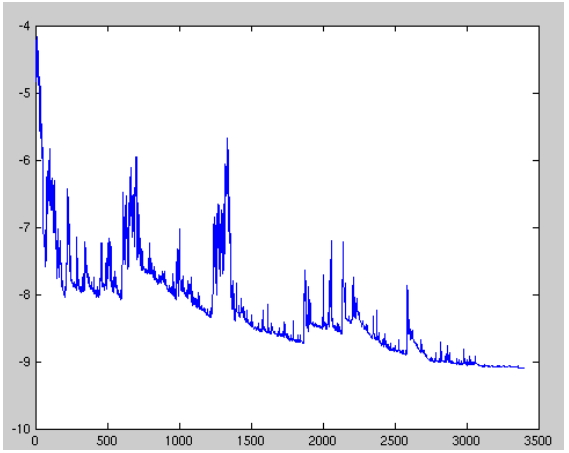
- Degrades accuracy
- Poor robustness to adversarial inputs
- Existing solutions requires extensive hyper-parameter tuning

Stochastic Gradient Descent (SGD)

$$\text{Assume } f(W^t, x) = \frac{1}{n}\sum_{i=1}^{n} f_i(W^t, x)$$

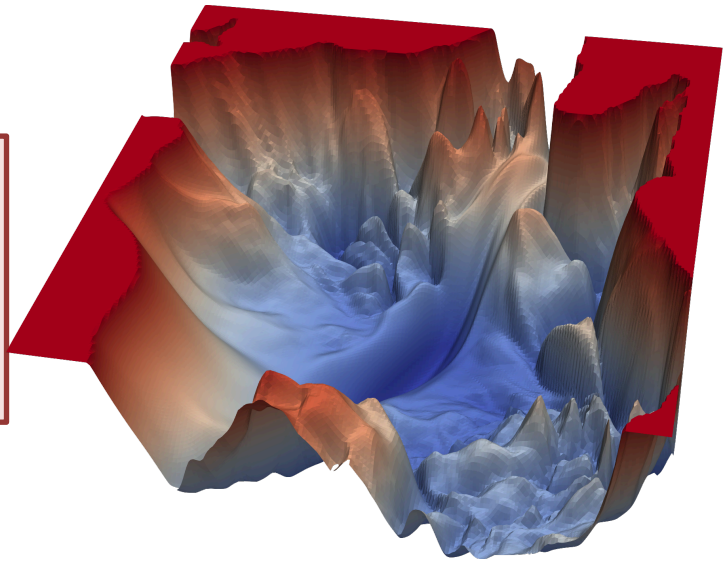
$$W^{t+1} \leftarrow W^t - \alpha \cdot \nabla_W f_i(W^t, x)$$

Pure SGD: compute gradient using 1 sample



$$W^{t+1} \leftarrow W^t - \alpha \cdot \frac{1}{b}\sum_{i=k+1}^{k+b} \nabla_W f_i(W^t, x)$$

Mini-batch: compute gradient using b samples



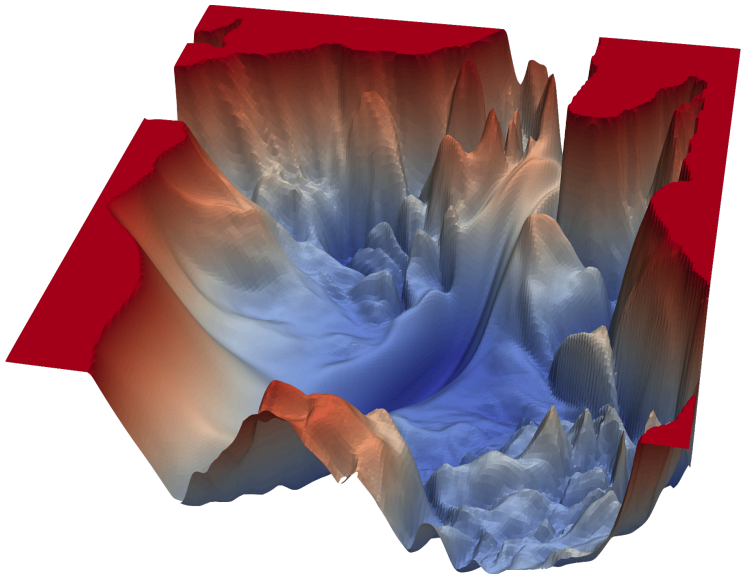
- Actually the name is a misnomer, *this is not a "descent" method*

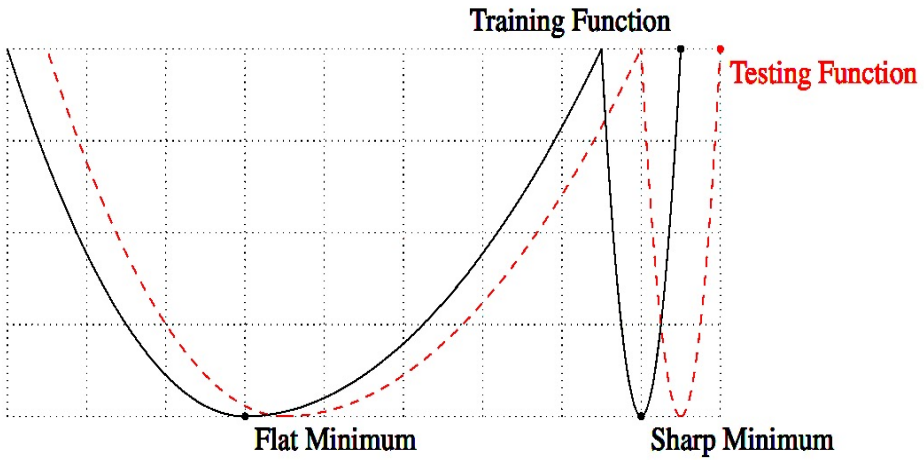
Image from https://www.cs.umd.edu/~tomg/projects/landscapes/

Poor Generalization

° Why large batch suffers from poor generalization performance?

- A common belief is that large batch training gets attracted to "sharp minimas"
- Another theory is that large batch may get stuck in saddle points





Loss landscape from https://www.cs.umd.edu/~tomg/projects/landscapes/
Keskar, Nitish Shirish, et al. "On large-batch training for deep learning: Generalization gap and sharp minima." ICLR'16 (arXiv:1609.04836)

Hessian Based Adaptive Batch Size with Adversarials

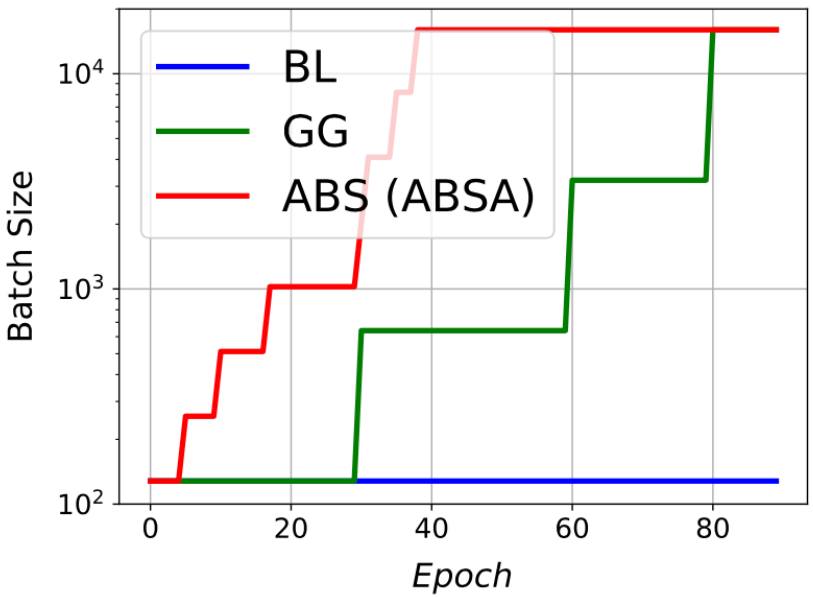
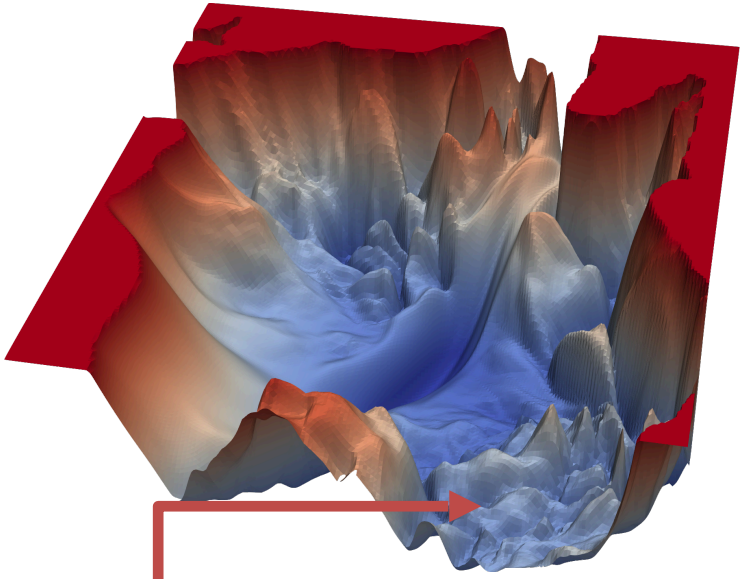
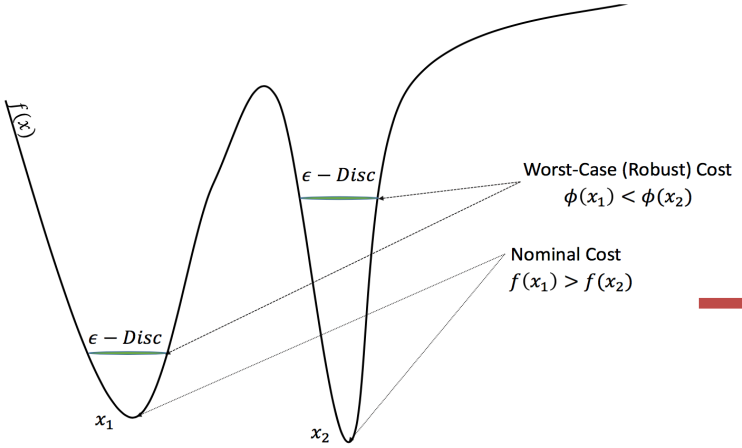


Illustration of batch size schedules of adaptive batch size as a function of training epochs.





Adversarials (robust training) can smooth out sharp "local minimas".

Yao, Gholami, Keutzer, and Mahoney. Large Batch Size Training of Neural Networks with Adversarial Training and Second-Order NeurIPS, 2018.

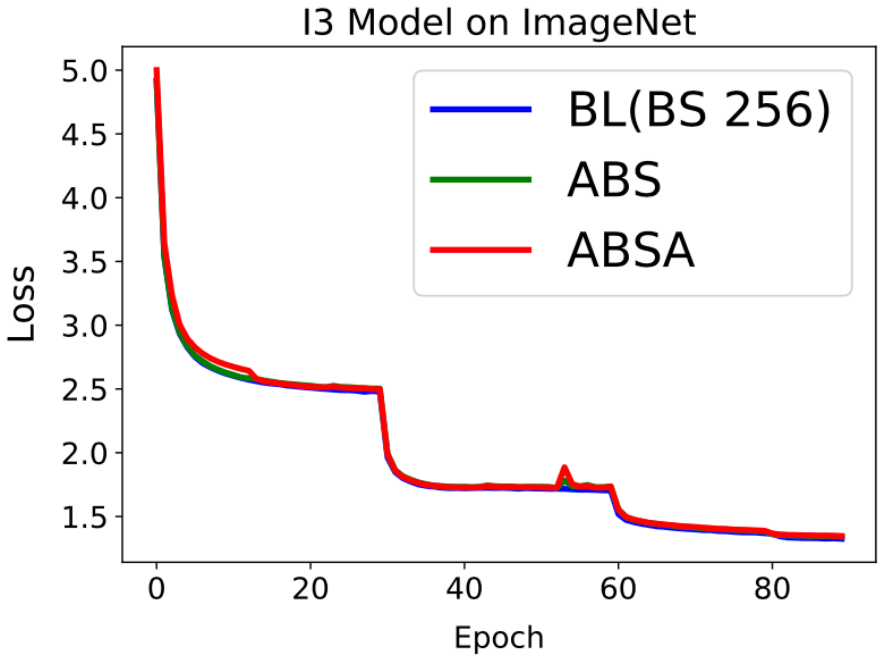
Results – ImageNet – ResNet18

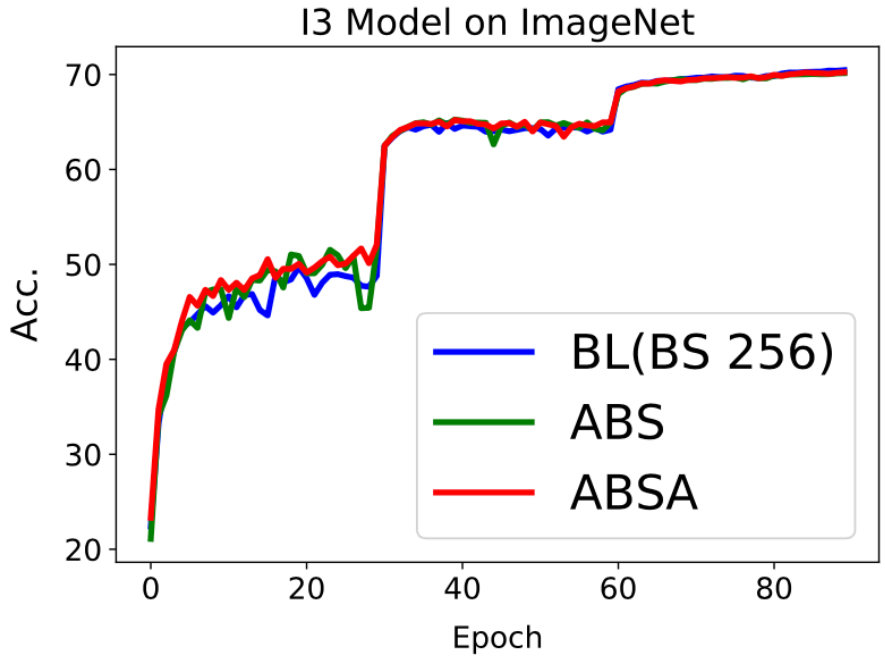
° Baseline:

- 450k SGD iterations, 70.4% validation accuracy

° ABSA:

- 66k SGD iterations, 70.2% validation accuracy





Outline

° Randomized Numerical Linear Algebra

° RandNLA in Second-order Optimization

° RandNLA in Deep Learning

° **Conclusions**

Conclusions

- RandNLA—combining linear algebra and probability—is at the center of the foundations of data.
- Randomness can be in the data and/or in the algorithm, and there can be interesting/fruitful interactions between the two:
  - Improvements in first-order/second-order convex/non-convex optimization theory/ practice.
  - Useful information in deep learning, providing more intuitive algorithm.

Thank You

Berkeley
UNIVERSITY OF CALIFORNIA