Stochastic Optimization Methods for Machine Learning

Jorge Nocedal

Northwestern University

SIAM CSE, March 2017

Collaborators

Richard Byrd
University of Colorado

R. Bollagragada
Northwestern

N. Keskar
Northwestern

D. Mudigere

P. Tang
INTEL

M. Smelyanski

Roger Fletcher 1939-2016

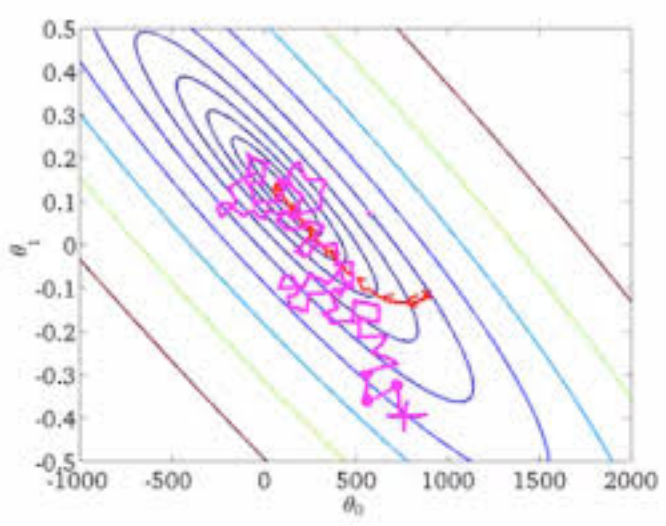
His imagination, originality and humility will be an example for future generations.

Initial Remarks

1. Continuous optimization in applied math *B. Stoufflet*
2. Central role also in Statistics
3. I will talk about optimization algorithms that are good learning algorithms - that generalize well
4. Illustrate with concrete example: training Deep Neural Networks
5. Contrast classical gradient-based methods and with the stochastic gradient method
6. For decades nonlinear optimization research focused on descent methods (line search or trust region). How else can one obtain (deterministic) convergence guarantees?
7. In large-scale machine learning applications, it is best to require only descent in expectation

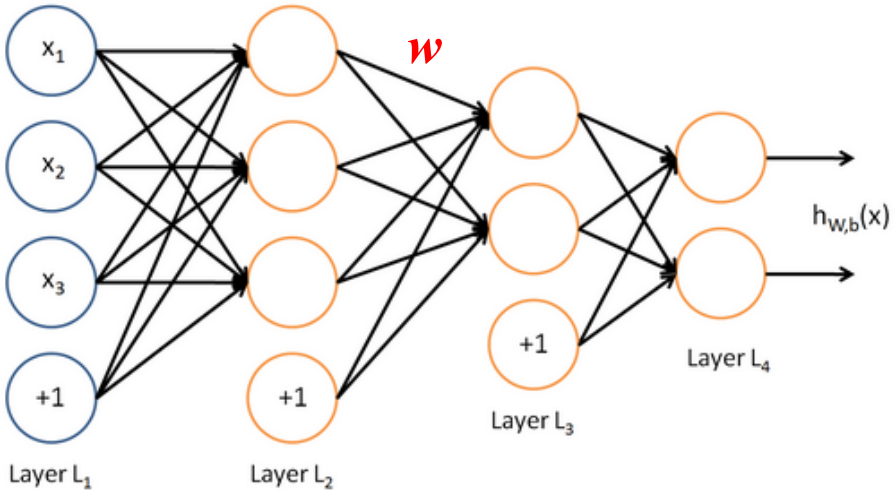
Initial remarks

Algorithms whose iterates are random variables and that are allowed to wonder around



1. Perform a more effective exploration of the data
2. Markov process has shown to be particularly effective: there is randomness at each iteration, but independent of previous decisions
3. Such behavior allows the optimization algorithm to produce solutions (prediction functions) that generalize well
4. Different from simulated annealing/genetic methods

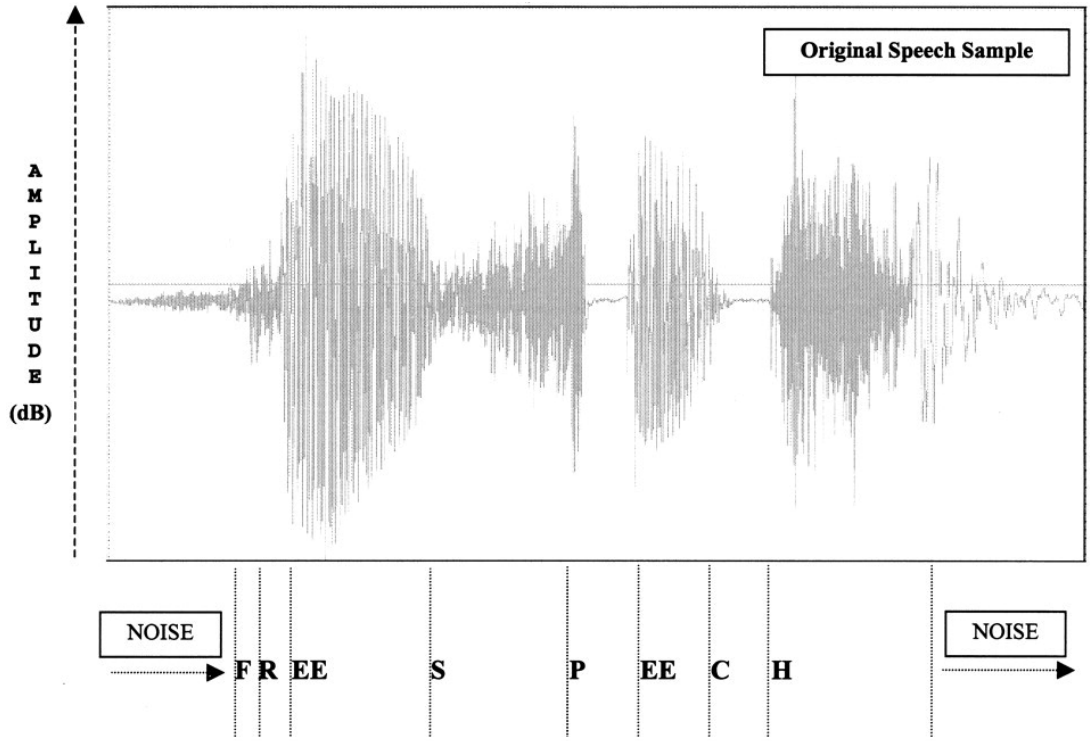
Deep neural networks



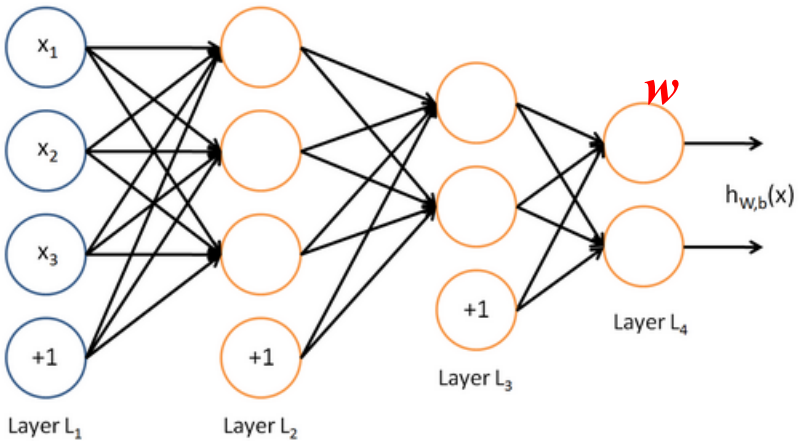
- Have produced quite a stir since 2011
- How? It is still not well understood,
- *Zhang, Bengio, Hardt, Recht, Vinyals (2017)*
- A highly nonlinear and non-convex predictor
- Input: images, acoustic frames, text
- Output: image classification, speech recognition, translation

Example: Speech recognition

Observe features X in acoustic frames
Predict word or sentence "FREE SPEECH"



Capacity of Deep neural networks



Can be viewed as a function with great expressive capacity: can reproduce large classes of functions

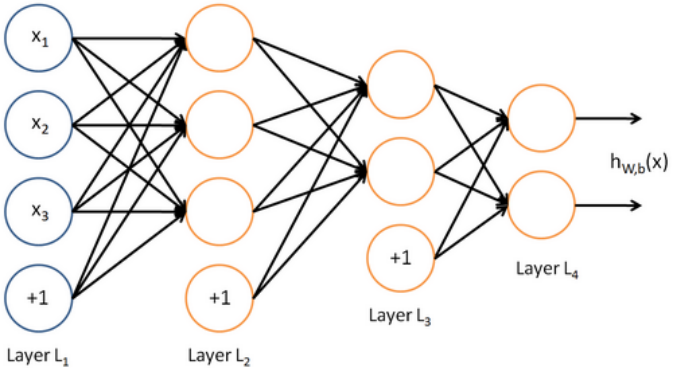
Piecewise polynomial in w of degree 7 in 10 million variables

Or can be seen as a composition of logistic regression units

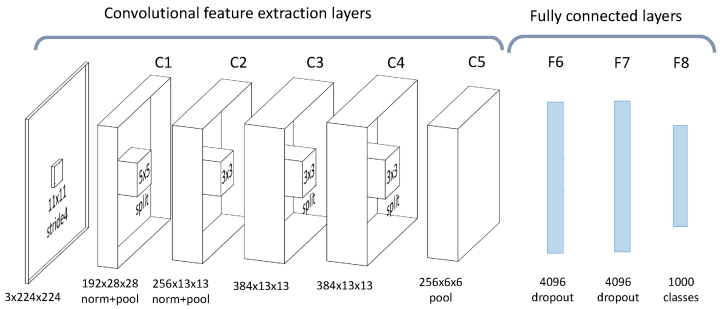
When training the DNN: Many minimizers, degenerate due to overcapacity

Our Observations Apply to Dominant Architectures

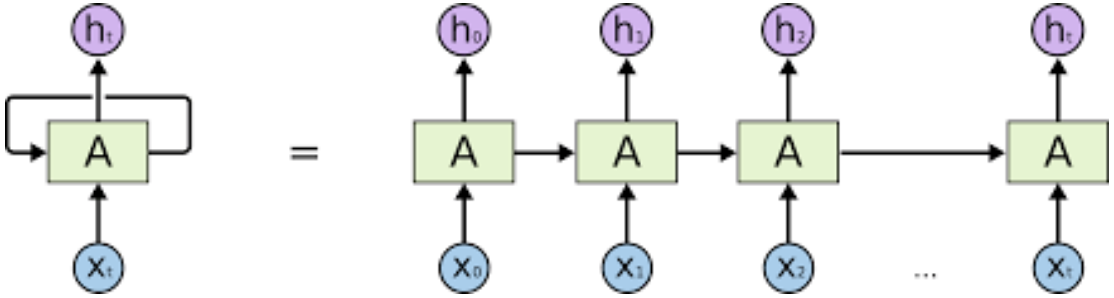
Feed forward



Convolutional Neural Network



Recurrent Neural Network



Next

- Illustrate how 2 optimization algorithms that give equally good solutions on the training problem produce solutions with different generalization properties
- Discuss the notable properties of the stochastic gradient method and how it dominates the classical gradient method
- Bottleneck: parallelism
- Search for new optimization algorithms suited for machine learning
- Sub-sampled Newton methods

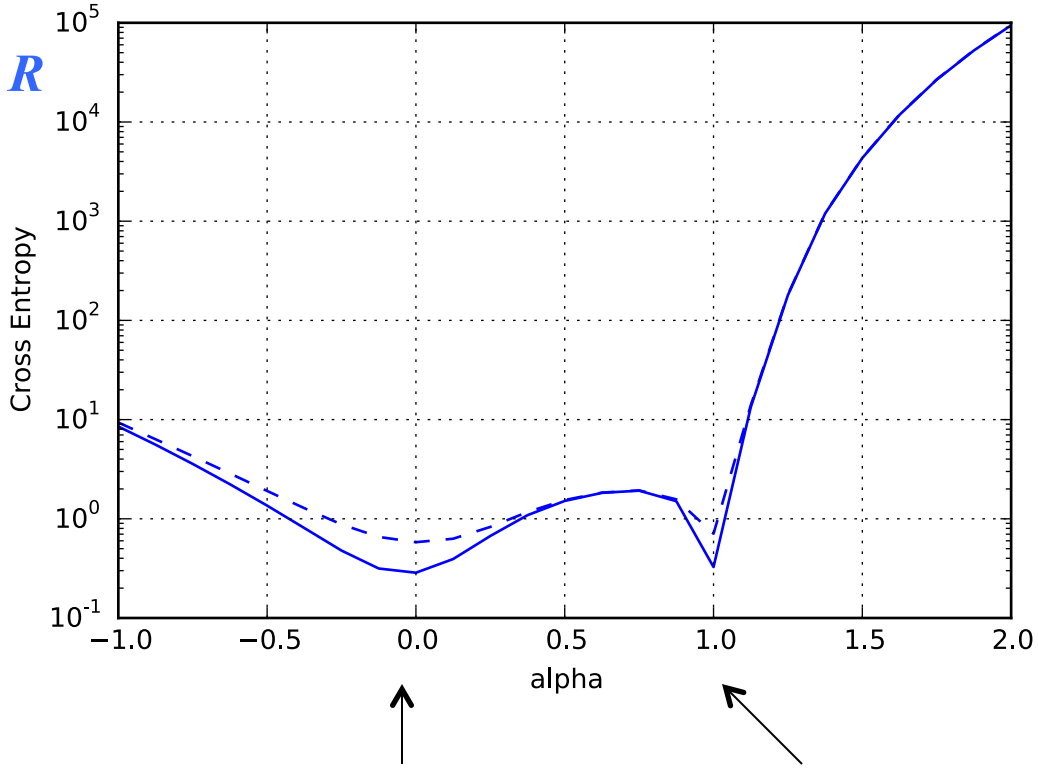
Some pictures

Training deep neural networks with:

- stochastic gradient method
- gradient based method (L-BFGS), batch method

Sharp and wide minima

Keskar et al. (2016)



Observing *training error* and *testing error* along line from SG solution to batch solution

Deep convolutional neural net CIFAR-10

Stochastic gradient solution

full gradient methodsolution

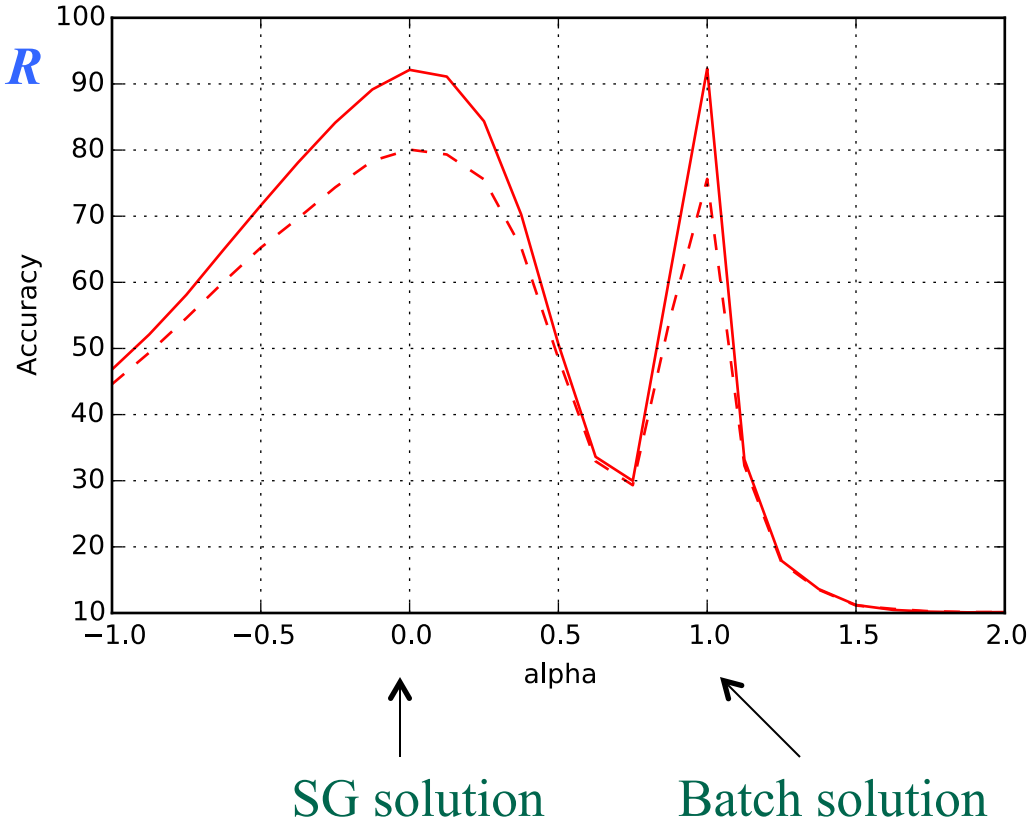
SG: mini-batch of size 256

ADAM optimizer

Batch: 10% of training set

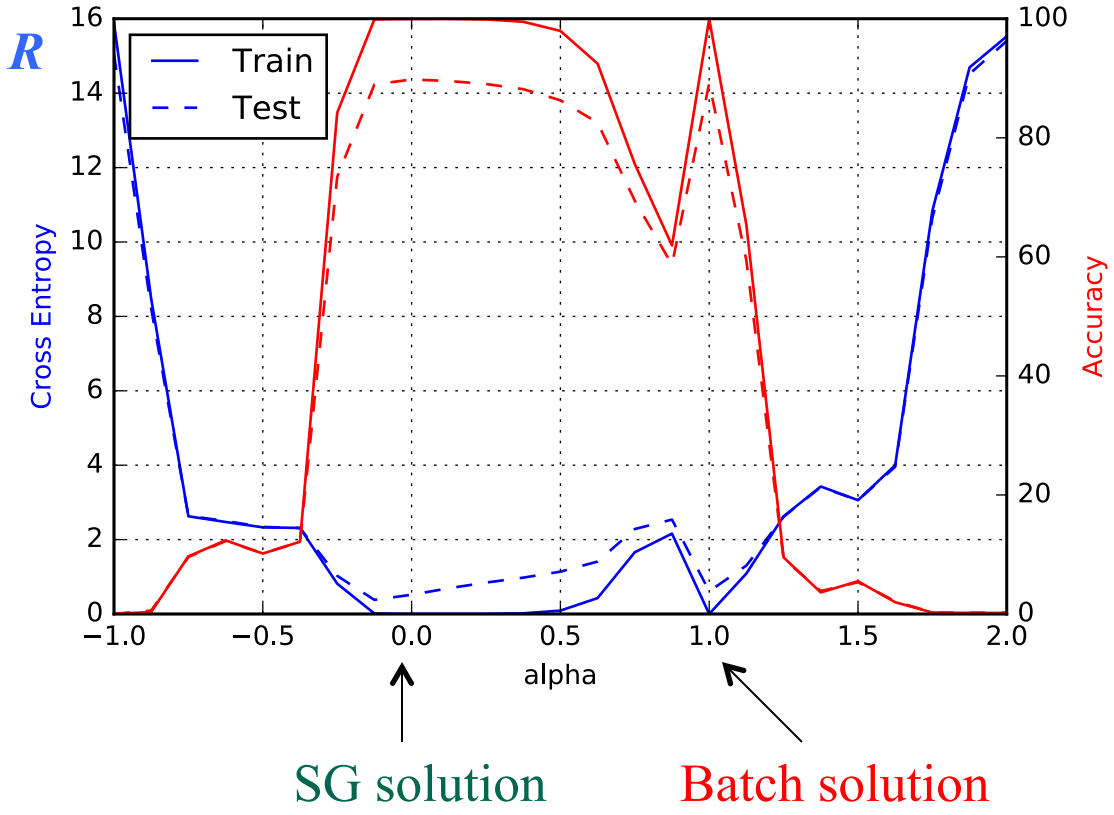
Accuracy: correct classification

Keskar et al. (2016)



Another example

Keskar et al. (2016)



What is going on?

Gradient method "over-fits"

We need to back-up:

- Define setting of supervised training
- Describe of optimization methods and their properties

Supervised Learning

Given a sizable training set of size n; each example i consists of

x_i : feature information $\quad$ y_i : correct label

Supervised Learning

Given a sizable training set of size *n;* each example *i* consists of

x_i : feature information $\quad$ y_i : correct label

Define prediction function ***h*** that depends on unknown parameter *w*,

$$h(w;x) = w^T x \quad \text{or} \quad h(w;x) = \text{nonlinear}$$

that makes good predictions on unseen data $(\hat{x}, \hat{y})$

$$\overline{y} = h(w;\hat{x}) \quad \text{with} \quad \overline{y} \approx \hat{y}$$

Choose a loss function $\ell(\overline{y}, \hat{y})$ and solve optimization problem

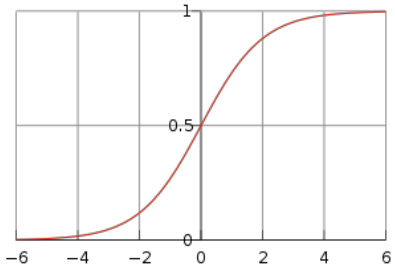
$$\min_w \; F(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(w;x_i), y_i)$$

Loss Functions

Logistic regression: $\log(1+\exp(-y(w^T x))$

For multi-class classification, C= set of classes

$$F(w) = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(w_{y_j}^T x_i)}{\sum_{j\in C}\exp(w_j^T x_i)}$$



Training error vs Testing Error --- Learning Algorithms

$(x,y) \in Z$ denotes all input-output pairs with distribution $P(x,y)$

Define $f(w,x_i,y_i) = \ell(h(w;x_i),y_i)$

Expected Risk: $F(w) = \int f(w;x,y)\,dP(x,y)$

Empirical Risk: $R(w) = \frac{1}{n}\sum_{i=1}^{n} f(w;x_i,y_i)$

$$R(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

Finite Sum Problem

$f_i \equiv f(w;x_i,y_i)$ denotes the loss associated with the i-th data point

Optimization Problem

A. Kingsbury

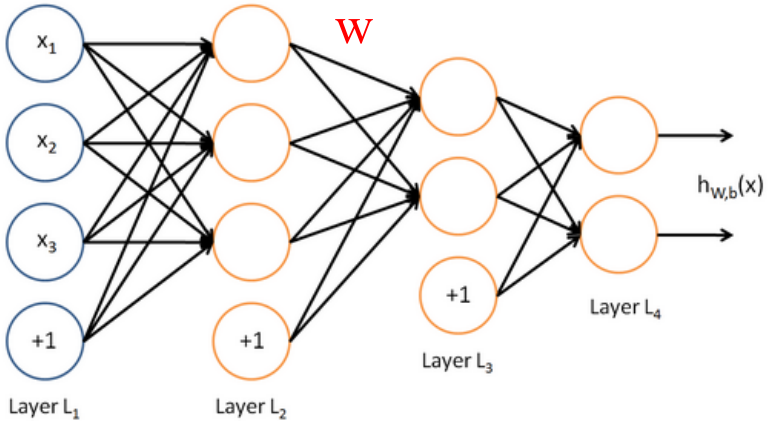
7 hidden layers

2000 units per layer

9000 label classes

6–44 million total parameters w

3.6–360 million examples



Therefore the problem: $\min_w \; F(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(w;x_i),y_i)$

Is a formidable optimization problem.

Training time ~ 2 days to 1 week

.. and deserves the respect of the CSE audience

Stochastic Gradient Method

For empirical risk minimization:

$$R_n(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

$$w_{k+1} = w_k - \alpha_k \nabla f_i(w_k)$$

$i \in \{1,\ldots,n\}$ choose at random

- Very cheap, noisy iteration; gradient w.r.t. just 1 data point
- Not a gradient descent method
- Stochastic process dependent on the choice of i

$$w_{k+1} = w_k - \alpha_k \nabla R_n(w_k)$$

gradient method

$$w_{k+1} = w_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla f_i(w_k)$$

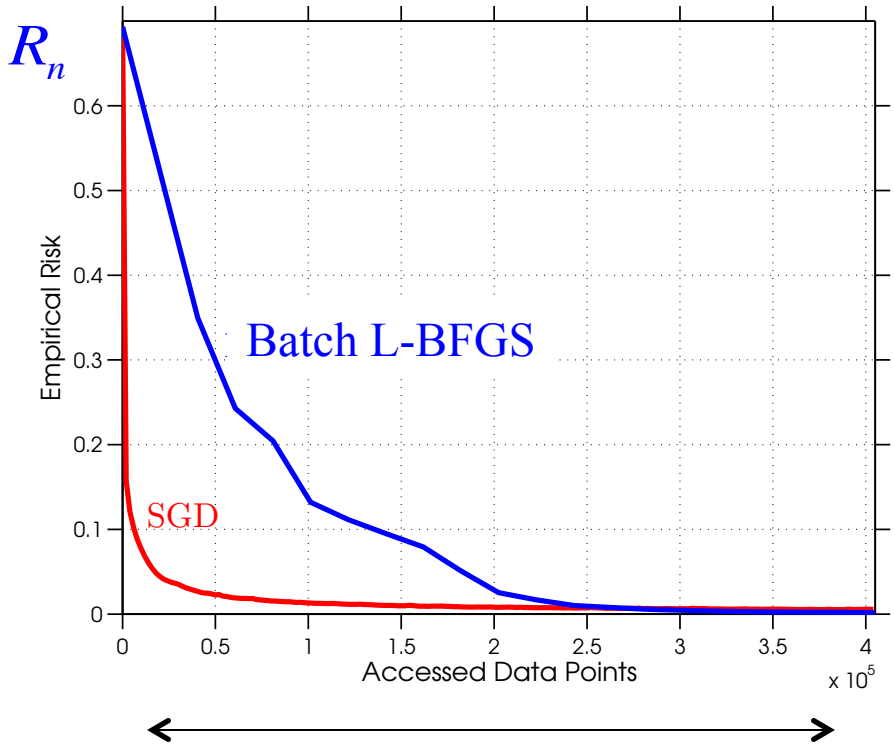
- More expensive, accurate step
- Can choose among a wide range of optimization algorithms
- Opportunities for parallelism

Why has SG emerged as the preeminent method?

Computational trade-offs between stochastic and batch methods

Ability to generalize: minimize F

Efficiency



Logistic regression;
speech data

Fast initial progress
of SG followed by drastic
slowdown

Can we explain this?

Intuition

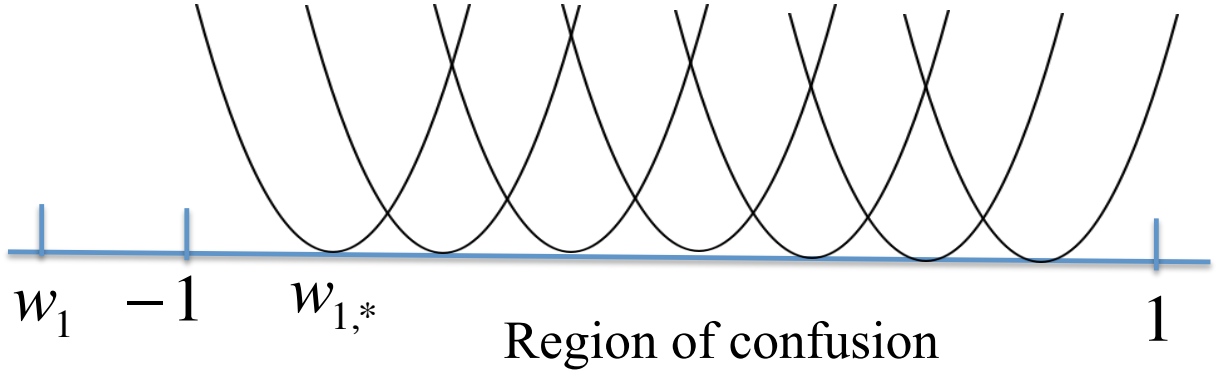
SG employs information more efficiently than batch methods

Argument 1:

Suppose data consists of 10 copies of a set S
Iteration of batch method 10 times more expensive
SG performs same computations

Example by Bertsekas

$$R_n(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$



Note that this is a geographical argument

Analysis: given w_k what is the expected decrease in the objective function R_n as we choose one of the quadratics randomly?

Computational complexity

Total work to obtain $R_n(w_k) \leq R_n(w^*) + \epsilon$

Batch gradient method:	$nd\kappa \log(1/\epsilon)$
Stochastic gradient method:	$d\nu\kappa^2 / \epsilon$

Think of $\epsilon = 10^{-3}$

n : # of training points

d : # of variables

κ : condition number

Srebro et al. Bottou et al.

A fundamental inequality

$$\mathbb{E}_k[R_n(w_{k+1}) - R_n(w_k)] \leq -\alpha_k \|\nabla R_n(w_k)\|_2^2 + \alpha_k^2 \mathbb{E}_k \|\nabla f_{i_k}(w_k)\|^2$$

Initially, gradient decrease dominates; then variance in gradient hinders progress

To ensure convergence: $\alpha_k \to 0$ in SG method to control noise

Variance reduction methods directly control the noise given in the last term

The variant when $\alpha_k = \alpha$ is constant has also been thorough studied and yields convergence to a neighborhood of the solution at linear rate

Bottou, Curtis, Nocedal (2016) prepared for SIAM Review

Learning algorithm

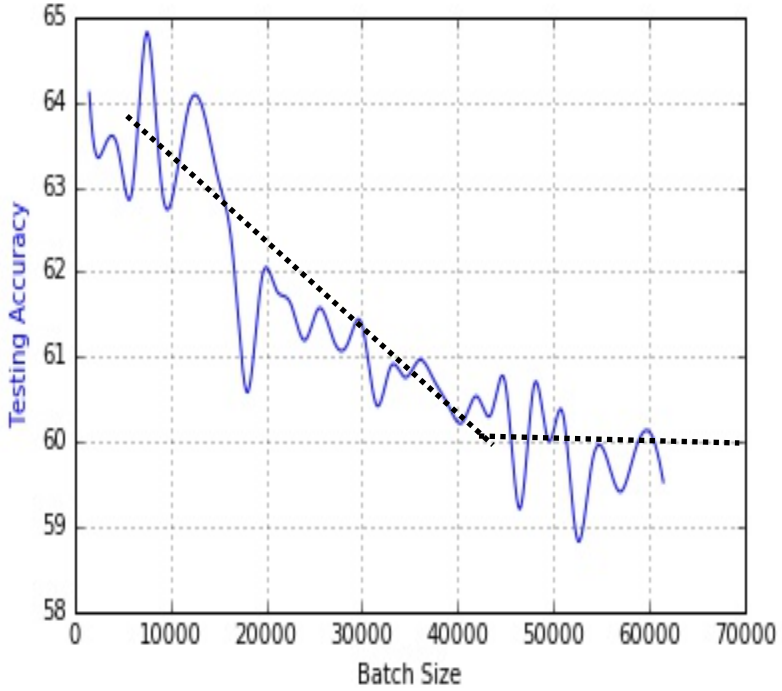
Let us look more closely at the training of deep Neural networks

It has been known for a long time that batch methods are inferior

- Accuracy is lost with increase in batch size
- ADAM optimizer: 256 (small batch) v/s 10% (large batch)

Testing Accuracy*: R*

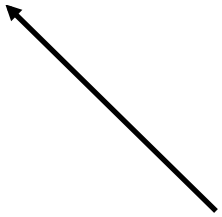
Studied 6 network configurations



Training and Testing Accuracy

SB: small batch LB: large batch

Network Name	Training Accuracy		Testing Accuracy	
	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99 \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$



No Problems in Training!

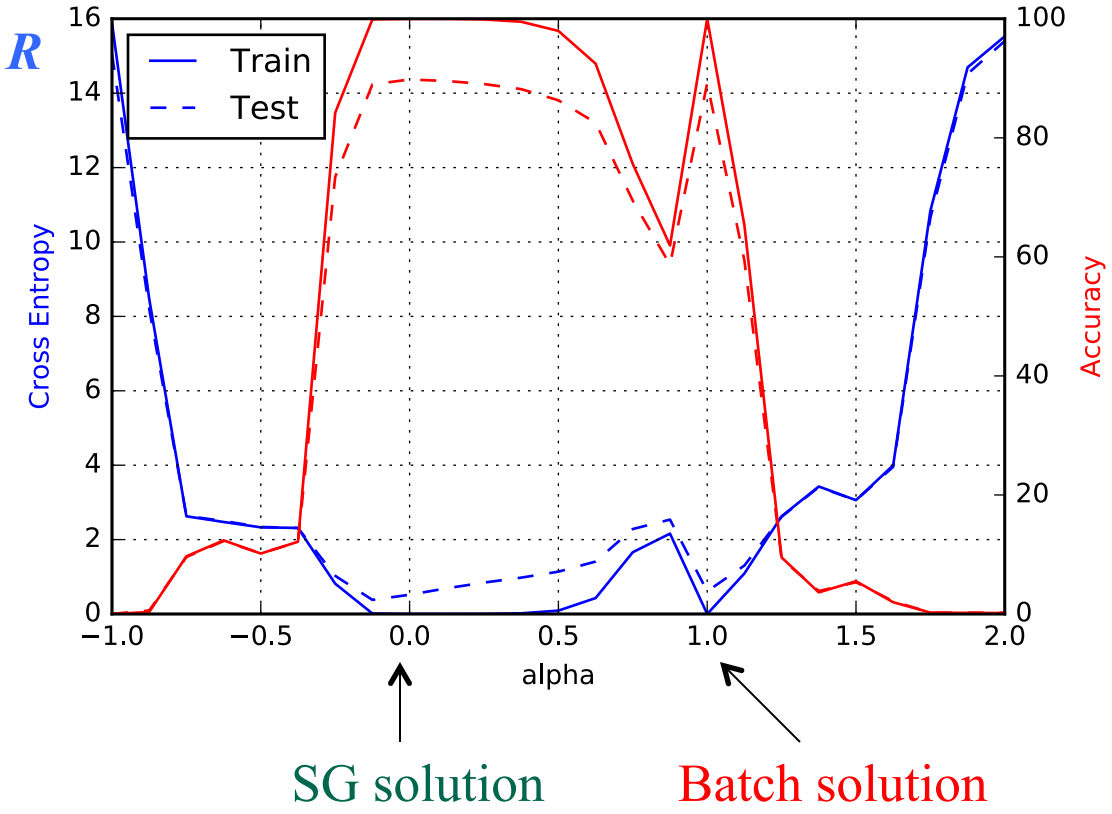
Network configuarions

Table 1: Network Configurations

Name	Network Type	Architecture	Data set
F_1	Fully Connected	Section B.1	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	Section B.2	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	Section B.3	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	Section B.4	CIFAR-10
C_3	(Shallow) Convolutional	Section B.3	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	Section B.4	CIFAR-100

Sharp and flat minima

Keskar et al. (2016)



Observing R along line From SG solution to batch solution

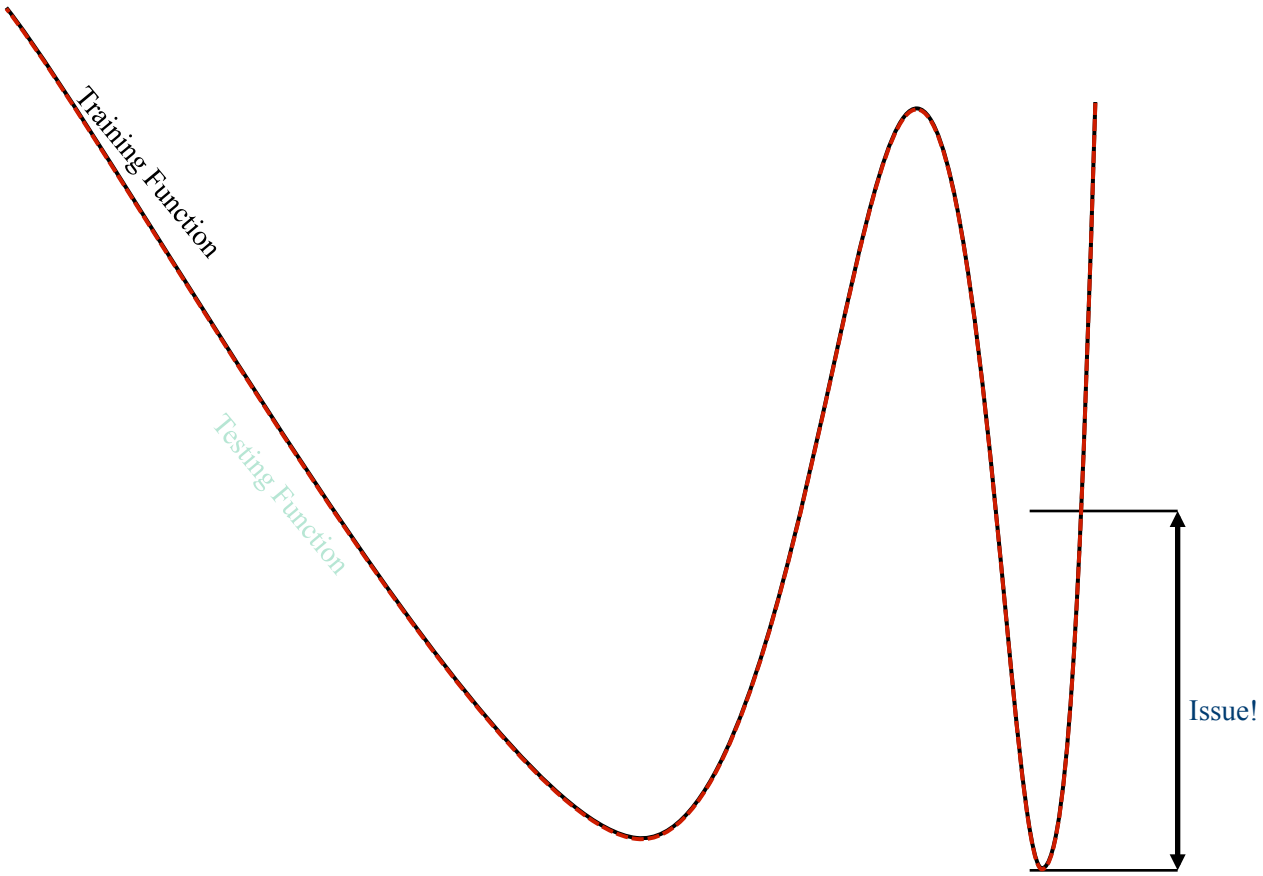
Goodfellow et al

Deep convolutional Neural net CIFAR-10

SG: mini-batch of size 256 Batch: 10% of training set

ADAM optimizer

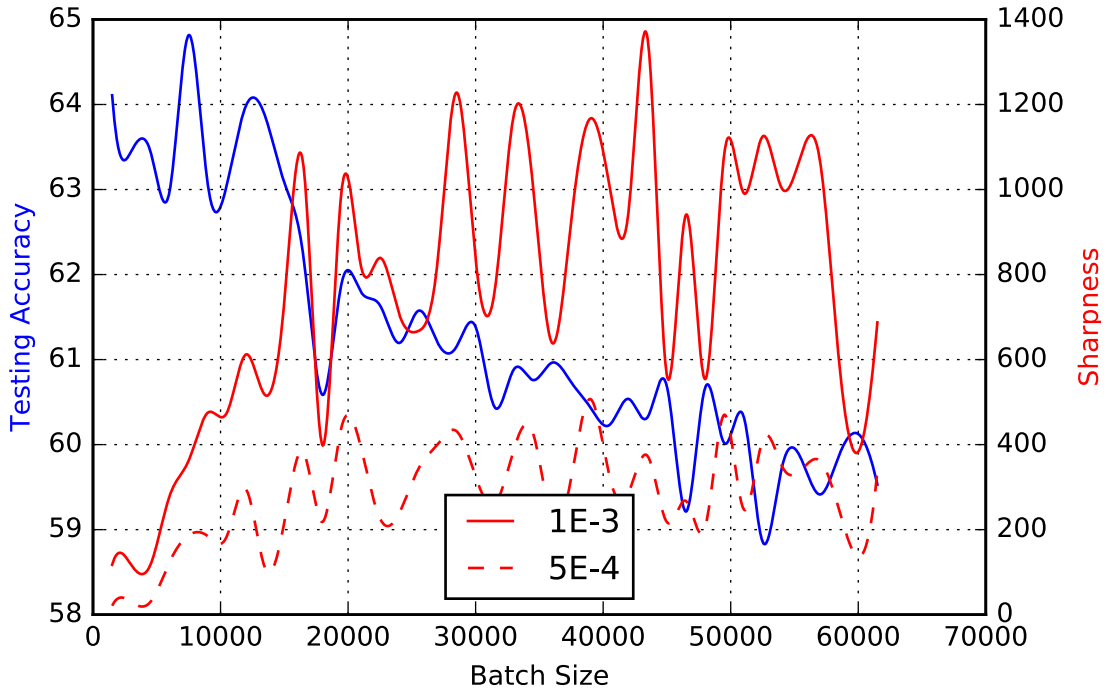
Sharp minima



Small batch solution large batch solution

Testing accuracy and sharpness

Keskar (2016)

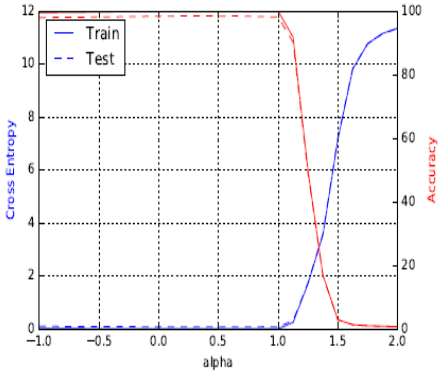


Testing accuracy vs batch size

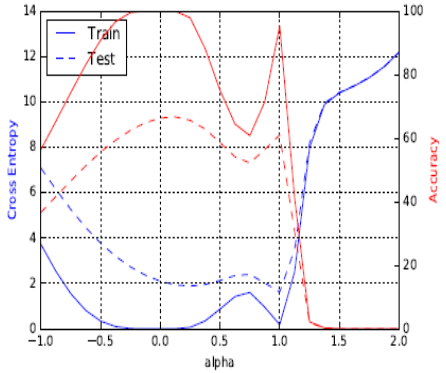
Sharpness of minimizer vs batch size

Sharpness:

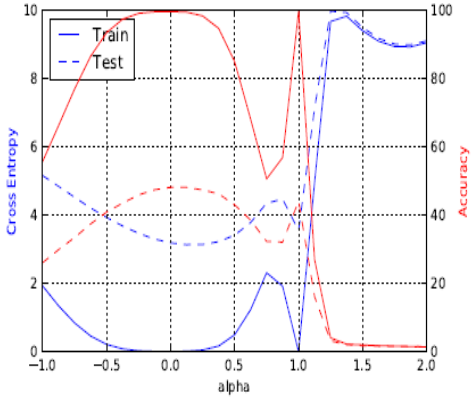
Max R in a small box around minimizer



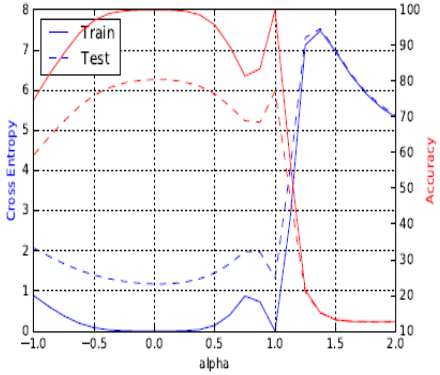
(a) F_1



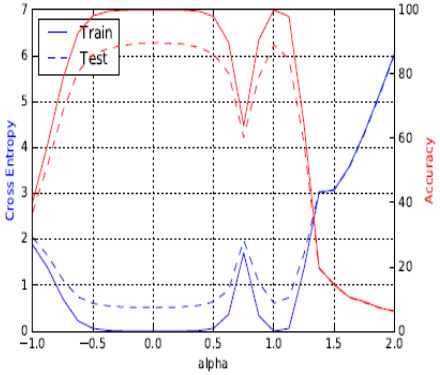
(b) F_2



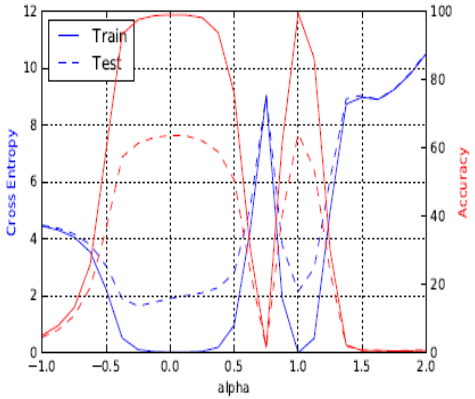
(e) C_3



(c) C_1



(d) C_2

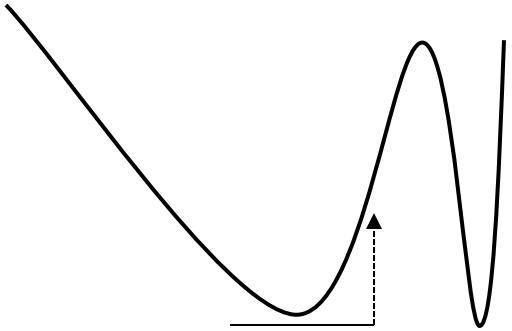


(f) C_4

Sharpness Metric

Given a minimizer w^* and a box B of width ϵ centered at w^*, we define the sharpness of w^* as

$$max_{w \in B} \frac{f(w^* + w) - f(w^*)}{1 + f(w^*)}$$

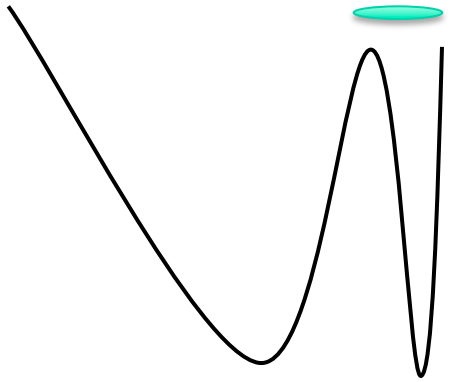


Box B

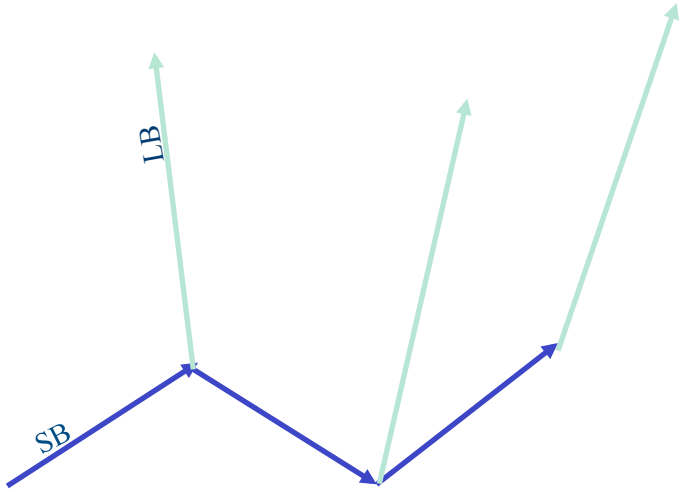
	$\epsilon = 10^{-3}$		$\epsilon = 5 \cdot 10^{-4}$	
	SB	LB	SB	LB
F_1	1.23 ± 0.83	205.14 ± 69.52	0.61 ± 0.27	42.90 ± 17.14
F_2	1.39 ± 0.02	310.64 ± 38.46	0.90 ± 0.05	93.15 ± 6.81
C_1	28.58 ± 3.13	707.23 ± 43.04	7.08 ± 0.88	227.31 ± 23.23
C_2	8.68 ± 1.32	925.32 ± 38.29	2.07 ± 0.86	175.31 ± 18.28
C_3	29.85 ± 5.98	258.75 ± 8.96	8.56 ± 0.99	105.11 ± 13.22
C_4	12.83 ± 3.84	421.84 ± 36.97	4.07 ± 0.87	109.35 ± 16.57

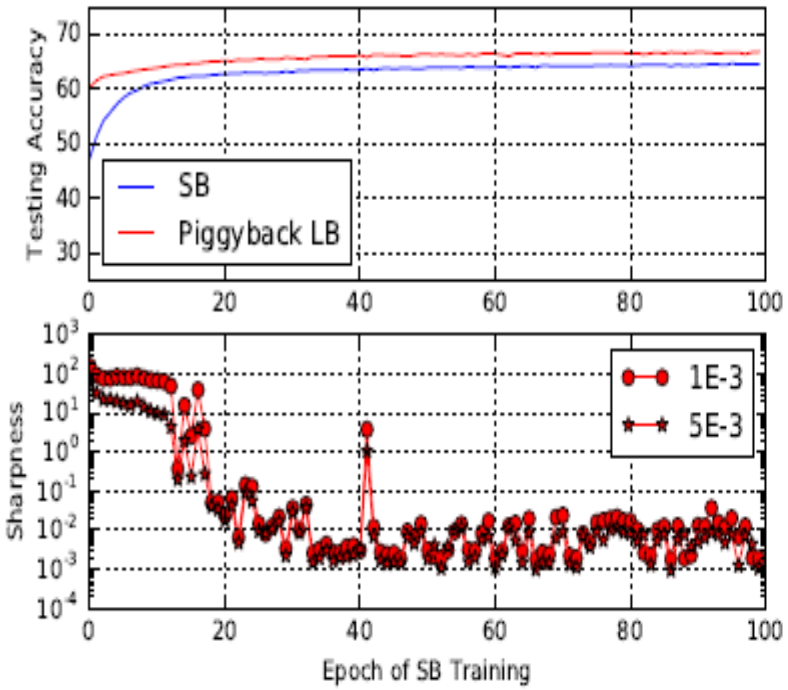
Robust Optimization View

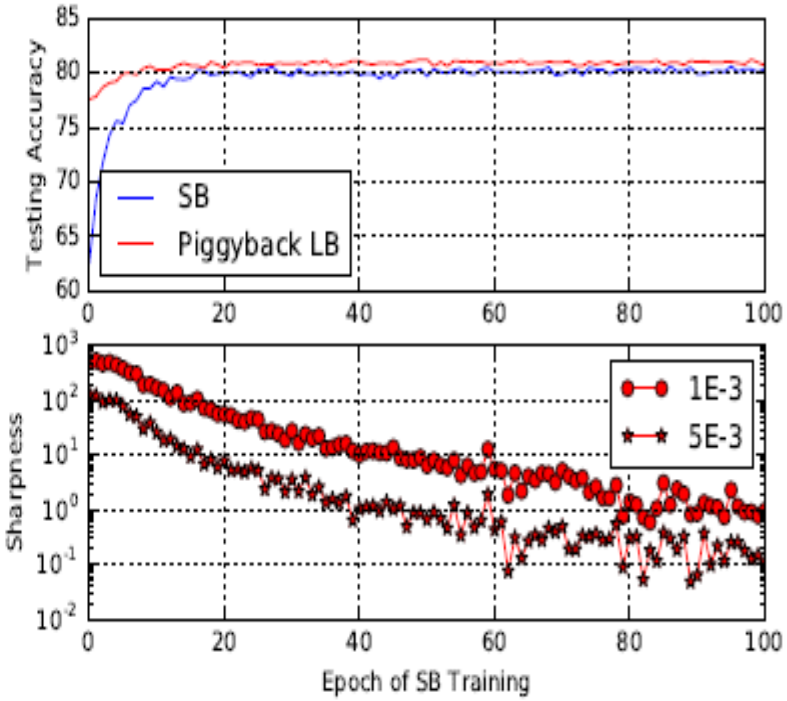
$$\min_{x} \quad \phi(x) := \max_{\|\Delta x\| \leq \epsilon} f(x + \Delta x)$$



Hot starts


LB
SB


Testing Accuracy
SB
Piggyback LB
Sharpness
1E-3
5E-3
Epoch of SB Training


Testing Accuracy
SB
Piggyback LB
Sharpness
1E-3
5E-3
Epoch of SB Training

- Can one understand reasons for convergence to *sharp minima*?
- What is the relative frequency of flat and sharp minima?
- Is there anything inherent to the architecture of the network that causes this behavior?

- Can one **steer** the training method away from sharp minima?
- Is a robust formulation feasible?

Drawback of SG method: distributed computing

SG is notoriously hard to parallelize

- Because it updates the parameters w with high frequency
- Because it slows down with delayed updates.

SG still works with relaxed synchronization

- Because this is just a little bit more noise.

Communication overhead give room for new opportunities

- There is ample time to compute things while communication takes place.
- Opportunity for optimization algorithms with higher per-iteration costs

➔ SG may not be the best algorithm for distributed training.

Beyond the stochastic gradient method

Let's consider an algorithm that is more general

Subsampled Newton Methods

$$R_n(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

Choose $S \subset \{1,\ldots,n\}$, $X \in \{1,\ldots,n\}$ uniformly and independently

$$\nabla^2 F_S(w_k)p = -\nabla F_X(w_k) \qquad w_{k+1} = w_k + \alpha_k p$$

Sub-sampled gradient and Hessian

$$\nabla F_X(w_k) = \frac{1}{|X|}\sum_{i\in X} \nabla f_i(w_k) \qquad \nabla^2 F_S(w_k) = \frac{1}{|S|}\sum_{i\in S} \nabla^2 f_i(w_k)$$

Focus on true objective: expected risk F

The stochastic nature of the objective creates opportunities:

Coordinate Hessian sample ***S*** and gradient sample ***X*** for optimal complexity

Active research area

- Friedlander and Schmidt (2011)
- Byrd, Chin, Neveitt, N. (2011)
- Erdogdu and Montanari (2015)
- Roosta-Khorasani and Mahoney (2016)
- Agarwal, Bullins and Hazan (2016)
- Pilanci and Wainwright (2015)
- Pasupathy, Glynn, Ghosh, Hashemi (2015)
- Xu, Yang, Roosta-Khorasani, Re', Mahoney (2016)
- Byrd, Bollapragada, N. (2016)

Linear convergence

Focus on Expected Risk F

$$\nabla^2 F_{S_k}(w_k)p = -\nabla F_{X_k}(w_k) \qquad w_{k+1} = w_k + \alpha p$$

The following result is well known for strongly convex objective:

Theorem: Under standard assumptions. If

a) $\alpha = \mu / L$

b) $|S_k| =$ constant

c) $|X_k| = \eta^k \quad \eta > 1$ (geometric growth)

Then, $\mathbb{E}[\| w_k - w^* \|] \to 0$ at a linear rate and

work complexity matches that of stochastic gradient method

Byrd, Chin, N. Wu, 2012

$\mu =$ smallest eigenvalue of any subsampled Hessian

$L =$ largest eigenvalue of Hessian of F

Local superlinear convergence

We can show the linear-quadratic result

$$\mathbb{E}_k[\| w_{k+1} - w^* \|] \leq C_1 \| w_k - w^* \|^2 + \frac{\sigma \| w_k - w^* \|}{\mu\sqrt{|S_k|}} + \frac{v}{\mu\sqrt{|X_k|}}$$

To obtain superlinear convergence:

i) $|S_k| \to \infty$

ii) $|X_k|$ must increase faster than geometrically

Formal superlinear convergence

Theorem: under the conditions just stated, there is a neighborhood of w^* such that for the sub-sampled Newton method with $\alpha_k = 1$

$$\mathbb{E}[\| w_k - w^* \|] \rightarrow 0 \quad \text{superlinearly}$$

Inexact Methods- what iterative solver to use?

$$\nabla^2 F_S(w_k)p = -\nabla F_X(w_k) \qquad w_{k+1} = w_k + \alpha_k p$$

1. Inexact method
   - Conjugate gradient
   - Stochastic gradient
2. Both require only Hessian-vector products

Newton-CG chooses a fixed sample S, applies CG to

$$q_k(p) = F(w_k) + \nabla F(w_k)^T p + \frac{1}{2} p^T \nabla^2 F_S(w_k) p$$

Newton-SGI (stochastic gradient iteration)

If we apply the standard gradient method to

$$q_k(p) = F(w_k) + \nabla F(w_k)^T p + \frac{1}{2} p^T \nabla^2 F(w_k) p$$

we obtain the iteration

$$p_k^{i+1} = p_k^i - \nabla q_k(p_k^i) \quad = (I - \nabla^2 F(w_k)) p_k^i - \nabla F(w_k)$$

Consider instead the semi-stochastic gradient iteration:

1. Choose and index j at random;
2. $p_k^{i+1} = (I - \nabla^2 F_j(w_k)) p_k^i - \nabla F(w_k)$

Change sample Hessian at each inner iteration

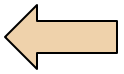
This method is implicit in Agarwal, Bullins, Hazan 2016

Comparing Newton-CG and Newton-GD

Number of Hessian-vector products to achieve

$$\| w_{k+1} - w^* \| \leq \frac{1}{2} \| w_k - w^* \| \qquad (*)$$

$O\left((\hat{\kappa}_l^{\max})^2 \hat{\kappa}_l \log(\hat{\kappa}_l) \log(d)\right)$	Newton-SGI
$O\left((\hat{\kappa}_l^{\max})^2 \sqrt{\hat{\kappa}_l^{\max}} \log(\hat{\kappa}_l^{\max})\right)$	Newton-CG



Agarwal, Bullins and Hazan (2016) and Xu, Yang, Re, Roosta-Khorasani, Mahoney (2016) Byrd, Bollapragada, N, (2016)

Decrease (*) obtained at each step with probability 1-p

Our results give convergence of the whole sequence in expectation

Complexity bounds are very pessimistic, particularly for CG

Final Remarks

- The search for effective optimization algorithms for machine learning is ongoing
- In spite of the total dominance of the SG method at present on very large scale applications

- SG does not parallelize well
- SG is a first order method affected by ill conditioning
- SG too noisy when high accuracy is desired

- A method that is noisy at the start and gradually becomes more accurate seems attractive
- Generalization properties are vital