Nonlinear Optimization Methods for Machine Learning

Jorge Nocedal

Northwestern University

University of California, Davis, Sept 2018

Introduction

We don't really know, do we?

a) Deep neural networks are poised to impact a host new set of applications

b) They have almost run their course and will produce little beyond what has currently been achieved (which has been impressive)

c) They will be replaced by simpler prediction systems that are easier to understand and interpret

From the viewpoint of optimization

a) Training deep neural networks has been done with essentially the same optimization algorithm for the last 25 years (LeCun): the stochastic gradient method SGD. Momentum?
b) DNN are complex (nonlinear & non-convex), very high dimensional; we don’t understand why the training process actually works.
c) More questions arise every year and promising ideas are challenged
d) State-of-the-art: heuristics (batch normalization) and new architectures (residual networks) have greatly facilitated optimization

Given all of this, should we try develop new algorithms for training DNN?

i. For many the answer is NO
ii. I take a different view

Opportunities

Although training DNNs seems so difficult and is so empirical,

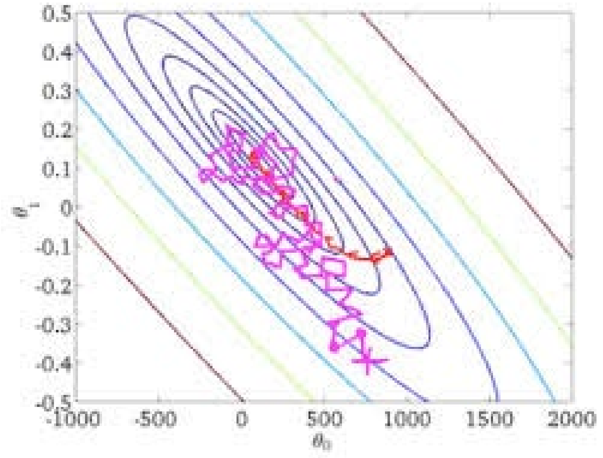
a) They provide motivation for revisiting some ideas (momentum?)
b) They suggest the development of new algorithms (variance reducing methods) that are bound to be useful in other domains
c) They pose interesting tradeoffs between computational and statistical efficiency that need to be answered --- which require fresh explanations

Specifically:

i. Is the highly noisy & Markovian nature of the stochastic gradient method SGD essential in training?
ii. If not, is there a way to achieve statistical efficiency and obtaining a high level of parallelism that dramatically reduces training time?

Current View

Algorithms whose iterates are random variables that are allowed to wonder around perform a more effective exploration of the data



and produce solutions (prediction functions) that generalize well

But a first order method suffers from ill conditioning. And the simple Markovian iteration of SGD is difficult to parallelize

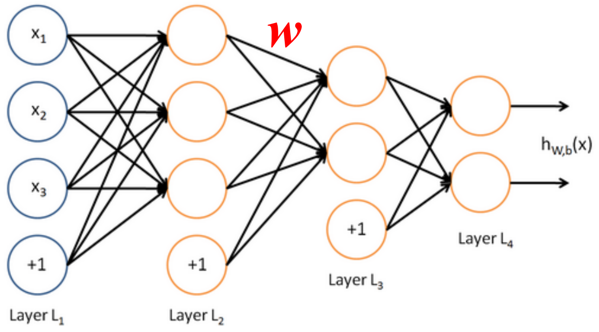
Our Proposal: A Progressive Sampling Method

- Use an increasingly accurate gradient (with a much larger batch) which enables data parallelism and the use of 2nd order information
- But, rather oddly, this could lead to generalization issues. The solutions obtained by SGD generalize better than those obtained with a large (fixed) sample
- This has been observed for many years (empirically); a dozen recent systematic studies
- The optimization method should be a good learning algorithm

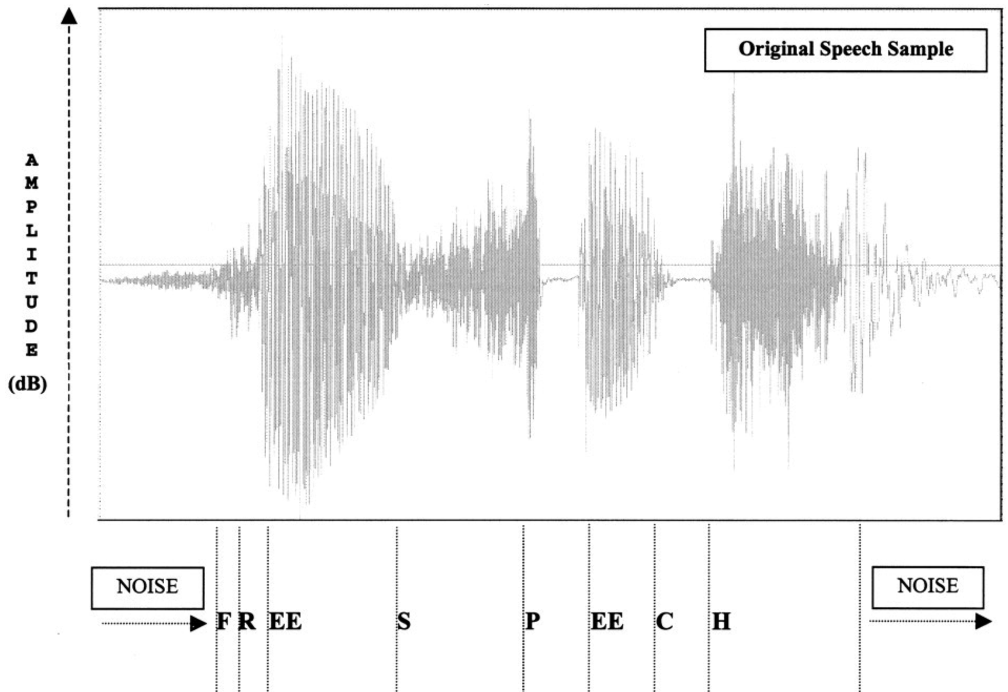
- So the story gets complicated and it is best to start at the beginning: Why neural networks? What is the optimization problem?

Deep neural networks

- Spectacularly successful for image & speech recognition, machine translation, etc
- A highly nonlinear and non-convex predictor



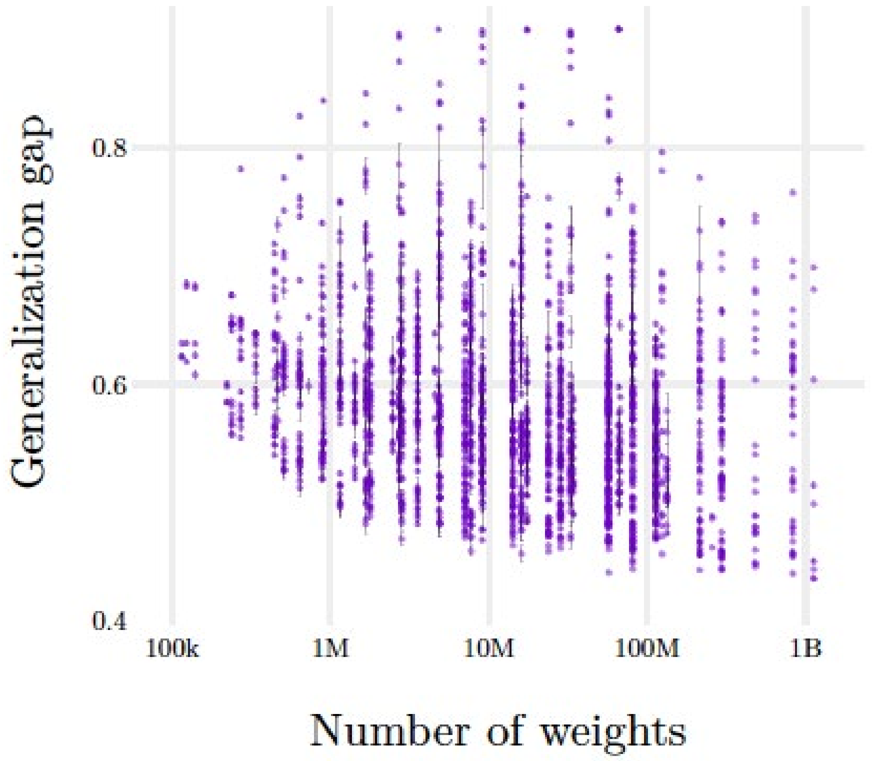
Observe features in acoustic frames, predict phrase “FREE SPEECH”

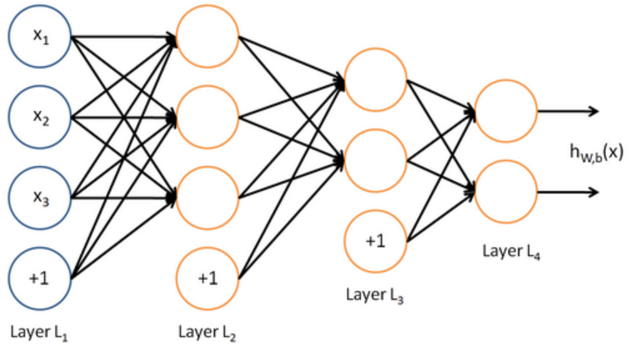


How have Neural Networks achieved such success?

- Not well understood
- Higher capacity DNN: good generalization – contrary to learning theory?

Novak et al. 2018





Notation

x_i : features $\quad$ y_i : label

Prediction function h that depends on unknown parameter w,

$$h(w;x) = \text{nonlinear}$$

Loss function $\ell(\bar{y},\hat{y})$ (e.g. cross entropy)

Solve optimization problem

$$\min_w \; F(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(w;x_i),y_i) + \frac{\lambda}{2}\|w\|_2^2$$

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

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

Formal statements of the objective functions

$$\min_w F(w) = \mathbb{E}_\xi[f(w;\xi)]$$

Loss over entire polulation

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

Many stochastic variance reduced methods for finite sum problem:

SAG, SAGA, SVRG,...: re-use information

Our goal: minimize F

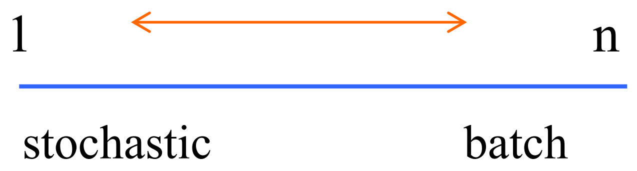
Stochastic gradient methods

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

Choose $X_k \subset \{1,\ldots,n\}$ uniformly and independently. Sampled gradient

$$\nabla F_{X_k}(w_k) = \frac{1}{|X_k|}\sum_{i\in X_k} \nabla f_i(w_k) \qquad w_{k+1} = w_k - \alpha_k \nabla F_{X_k}(w_k)$$

1. $|X_k| = 1$: stochastic gradient method
2. $|X_k| = n$: gradient method



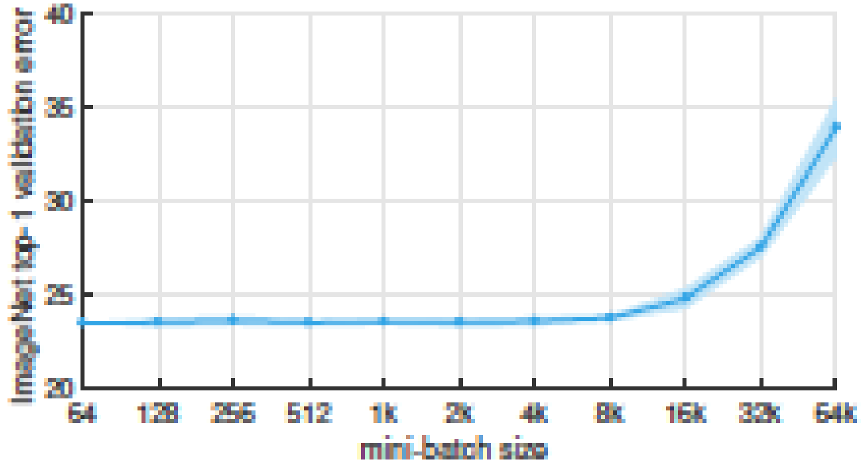
Why not use an increasingly accurate gradient method (with a much larger batch) which enables data parallelism and the use of 2nd order information?

The idea is gaining traction, but …

Two interesting new papers

Accuracy

Residual Network, Imagenet



Batch size

Goyal, He et al. 2017: From 29 hours of training time to 1 hour by increasing the batch size from 256 to 8k

Smith, Kindermans, Le (2017): batch of size 65k

We propose: Instead of choosing a fixed batch size: gradually increase it

Understanding SGD

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

- Why does it converge, and for what classes of functions?
- Do they include DNNs or only some?

For deterministic convex optimization: $\min F(w)$

$$F(w_{k+1}) - F(w_k) \leq -\alpha_k \| \nabla F(w_k) \|_2^2$$

For stochastic problem: $\min F(w) \equiv \mathbb{E}[f(w;\xi)]$

$$\boxed{\mathbb{E}[F(w_{k+1}) - F(w_k)] \leq -\alpha_k \| \nabla F(w_k) \|_2^2 + \alpha_k^2 \, \mathbb{E} \| \nabla f(w_k, \xi_k) \|^2}$$

$$\Downarrow$$

$$\nabla F_{X_k}(w_k)$$

Two algorithmic components:

- $\nabla F_x(w_k)$ is an unbiased estimator of $\nabla F(w_k)$ (or good angle...)
- Steplength $\alpha_k \to 0$ and rate is sublinear $O(1/k)$

  Constant steplength $\alpha_k = \alpha$. Linear convergence to a neighborhood

Fixed steplength Diminishing steplength

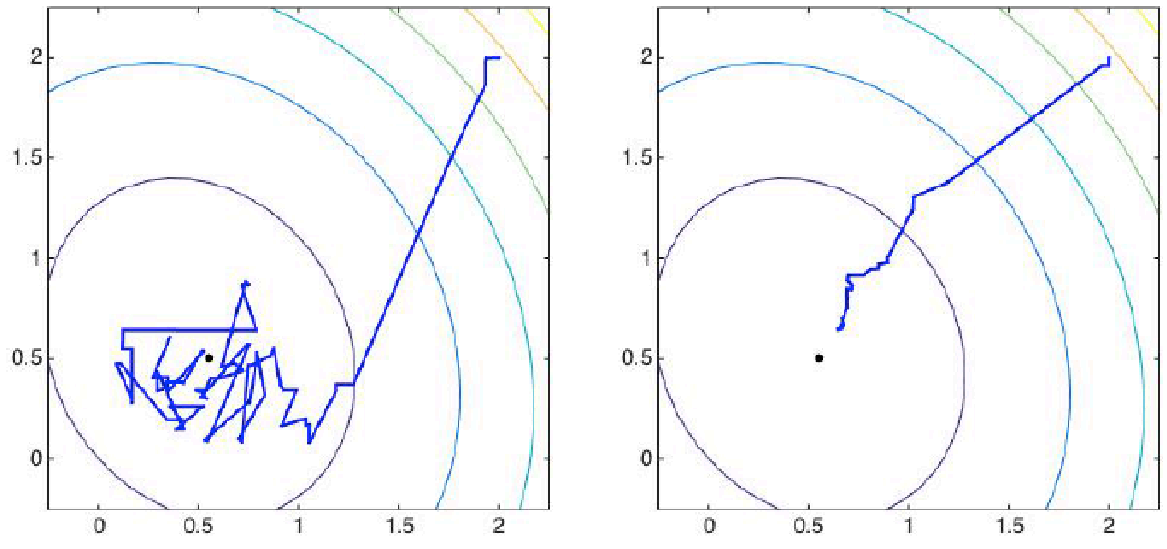


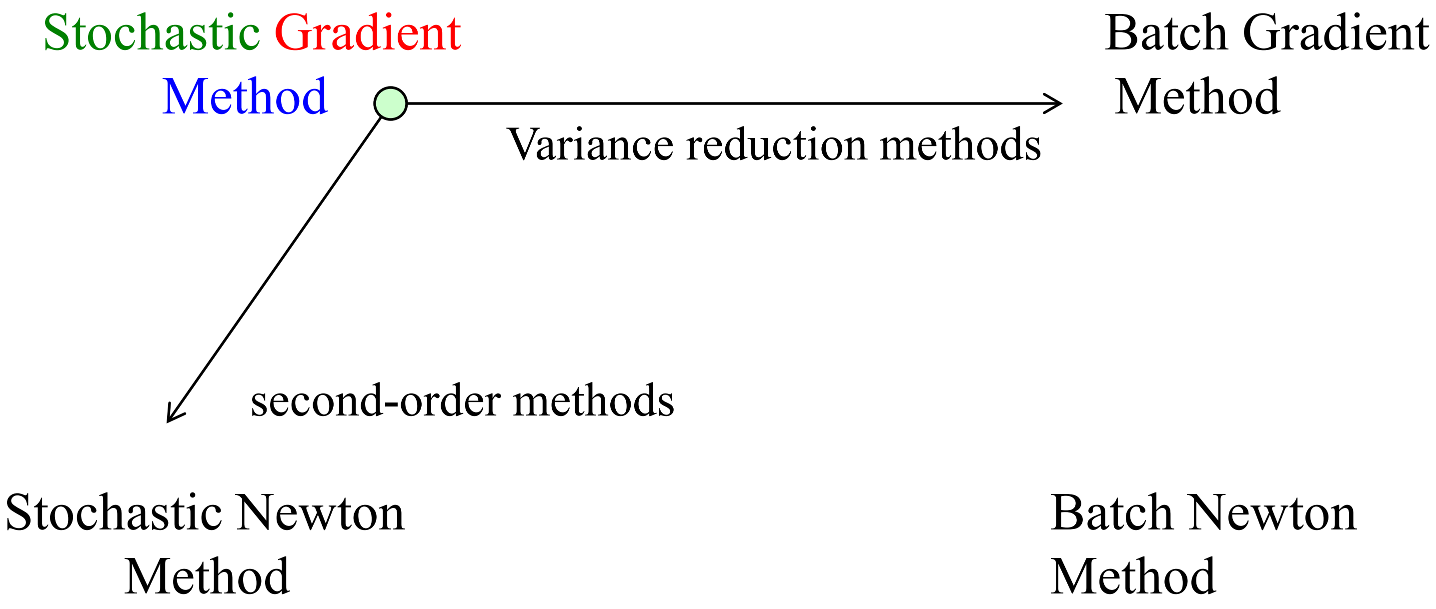
Figure: SG run with a fixed stepsize (left) vs. diminishing stepsizes (right)

Converges linearly to a neighborhood of the solution

Converges sub-linearly to the solution

Optimization panorama

Bottou, Curtis, Nocedal 2018



$$\boxed{\mathbb{E}[F(w_{k+1}) - F(w_k)] \le -\alpha_k \|\nabla F(w_k)\|_2^2 + \alpha_k^2 \mathbb{E}\|\nabla f(w_k,\xi_k)\|^2}$$

Enable second order information

- Start by considering Newton's method

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

Can subsample gradient: $\nabla F_{X_k}(w_k)$

- Can also subsample the Hessian
- Approximations to the Hessian with good statistical properties — Erdogdu & Montanari (2015)

How fast to increase sample?

Focus on Expected Risk F

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

Theorem: For strongly convex functions. If

a) $\alpha = \mu / L$

b) $|S_k| =$ constant

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

$\mu / L \approx$ condition number of Hessian

Then, $\mathbb{E}[\| w_k - w^* \|] \rightarrow 0$ at a linear rate and work complexity matches that of stochastic gradient method

Byrd, Chin, N. Wu, 2012

How to control the sample size in practice?

How to use progressive sampling in practice? $F(w)=\frac{1}{n}\sum_{i=1}^{n} f_i(w)$

First idea: Norm Test

Sample X_k should be large enough s.t.

$$\|\nabla F_{X_k}(w_k)-\nabla F(w_w)\| \le \theta \|\nabla F(w_k)\| \quad \theta \in (0,1)$$

or $$\frac{\mathbb{E}[\|\nabla F_i(w_k)-\nabla F(w_k)\|^2]}{|X_k|} \le \theta^2 \|\nabla F(w_k)\|^2$$

- Practice: leads to much faster increases in sample sizes than desired

Idea 2: Inner Product Test

Bollapragada, Byrd, N (2018)

Require only descent

$$\nabla F_{X_k}(w_k)^T \nabla F(w_k) > 0$$

Holds in expectation

$$\mathbb{E}[\nabla F_{X_k}(w_k)^T \nabla F(w_k)] = \| \nabla F(w_k) \|^2 > 0$$

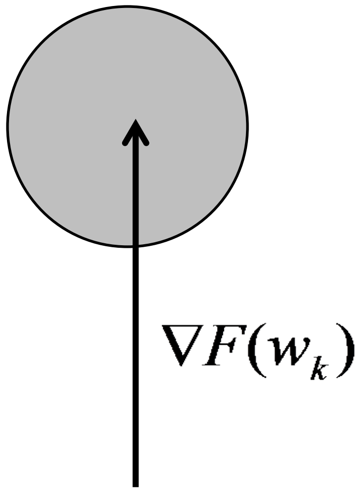
For descent to hold at most iterations, impose bounds on variance:

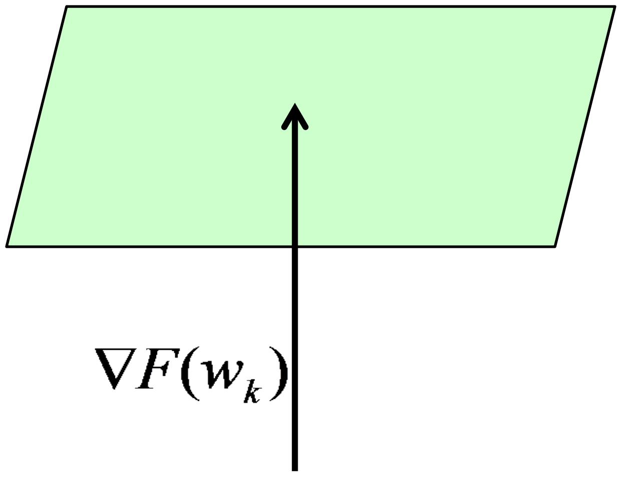
Sample X_k should be large enough s.t.

$$\boxed{\frac{\mathbb{E}[(\nabla F_i(w_k)^T \nabla F(w_k) - \| \nabla F(w_k) \|^2)^2]}{|X_k|} \le \theta^2 \| \nabla F(w_k) \|^4}$$

- Test designed to achieve descent sufficiently often

Comparison





- Norm test

Inner Product Test

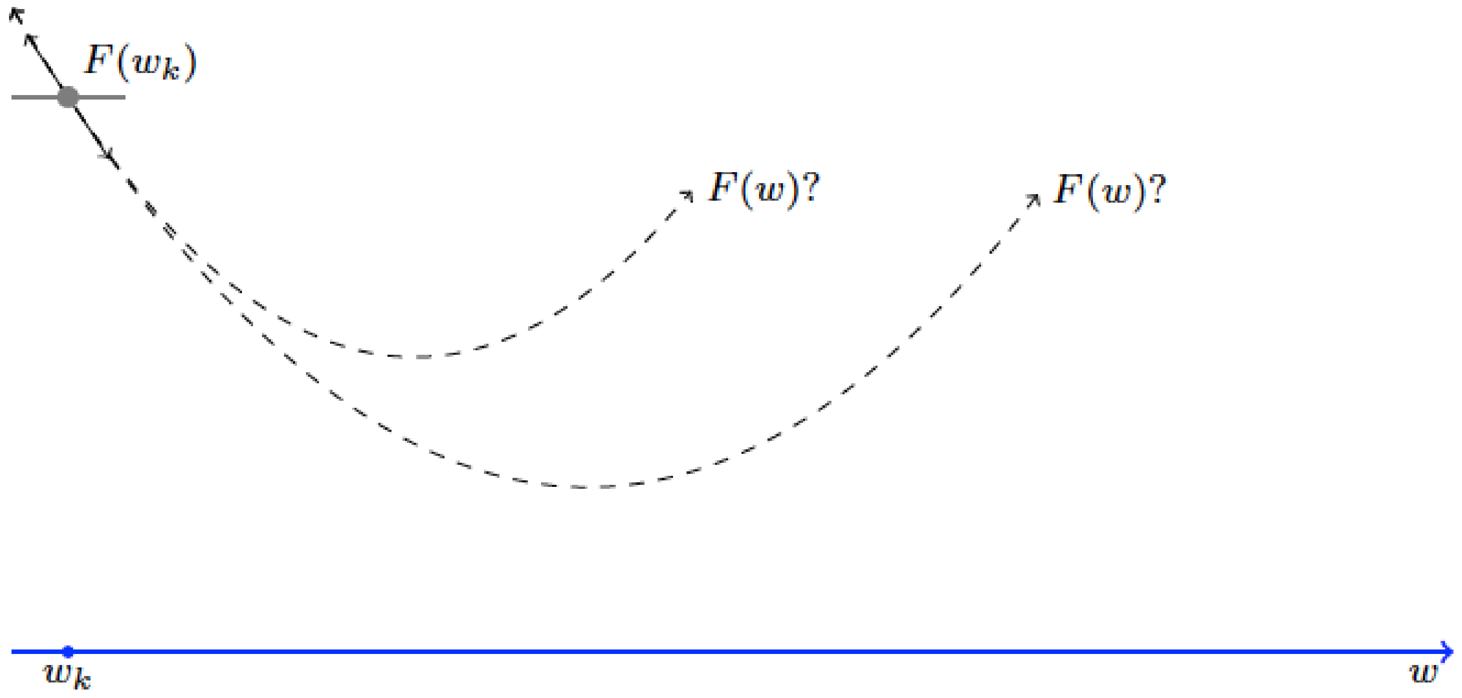
- Contrast to SG method
- Samples that satisfy Inner Product Test are smaller than for Norm Test

Lemma $|X_{IP}| \leq |X_N|$

Pause

On the Steplengths

$$\boxed{w_{k+1} = w_k - \alpha_k \nabla F_{X_k}(w_k)}$$

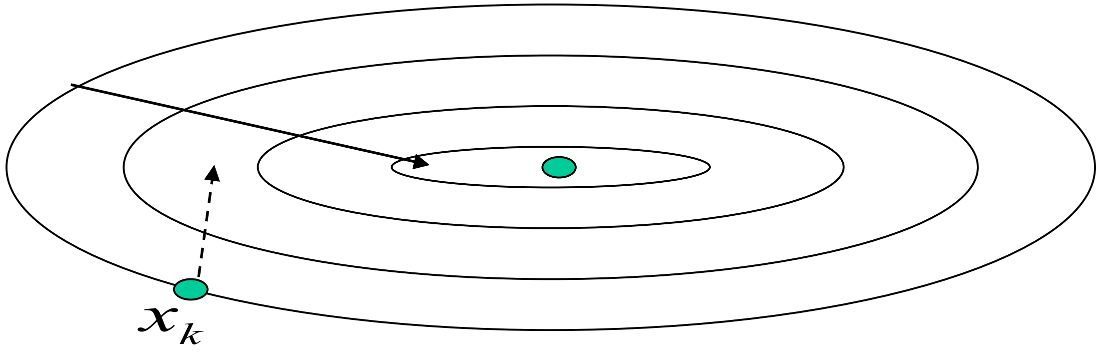


Scaling the Search Direction

- Different directions should be scaled differently
- For the noisy SGD method we will never find a formula for steplength that is universally practical
- Steplength tied up with noise suppression
- Mini-batching provides more freedom in choice of steplength

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

Deterministic Setting

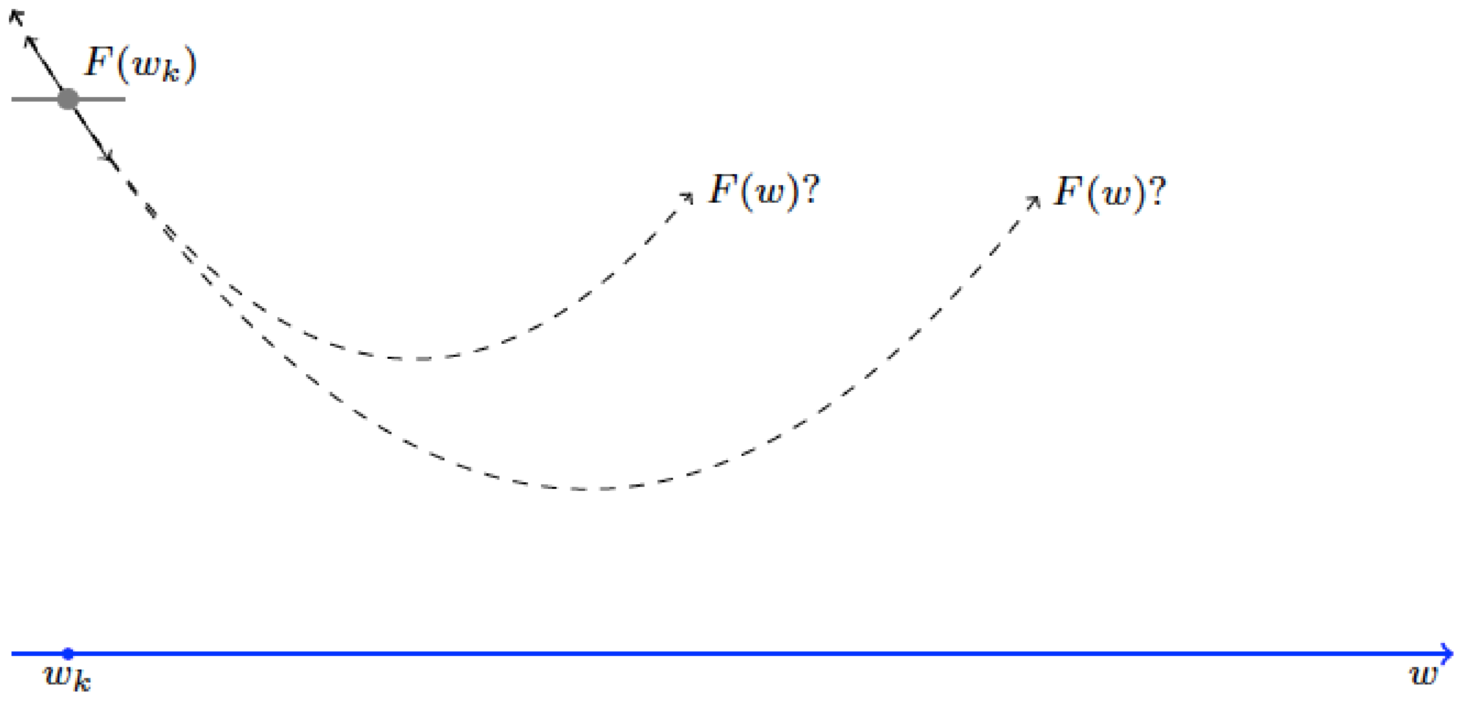


Scaling the Gradient Direction

Constant steplength (popular with theoreticians)

$$\alpha_k = 1/L \qquad L\text{: bound on } \|\nabla^2 F(w)\|$$

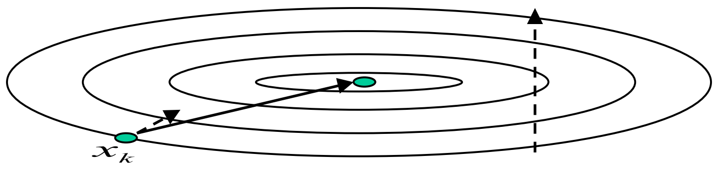
- Lipschitz constant L– the most conservative choice
- Adaptive (global) Lipschitz estimation – can be out of phase



Different gradient components should be scaled differently

$$w_{k+1} = w_k - \alpha_k D_k \nabla F_{X_k}(w_k)$$

1. Diagonal scaling (Adagrad, Adam)



2. Assumes knowledge along coordinate directions (difficult)
3. Generally not practical in deterministic optimization
4. Success of Adam and Adagrad explained through statistical arguments

Alternative:

- Instead of finding sophisticated steplength strategies, find method that produces well scaled directions
- Choice of steplength then becomes secondary
- Newton and quasi-Newton methods achieve this

Newton's method

1. An ideal iteration: scale invariant, local quadratic rate of convergence

$$w_{k+1} = w_k - \alpha_k \nabla^2 F(w_k)^{-1} \nabla F(w_k)$$

2. The Hessian contains a lot of information, but too costly to form/invert
3. How to approximate Newton's step?

Various Approaches:

1. Inexact Newton-CG – with subsampled Hessian
   - Computational unit: Hessian-vector product
2. Fischer Information – K-Fac   Martens &Grosse 2017
3. Quasi-Newton – shows much potential
   - Computational unit: gradient
4. Tensor based block diagonal structures   (Shampoo 2018)

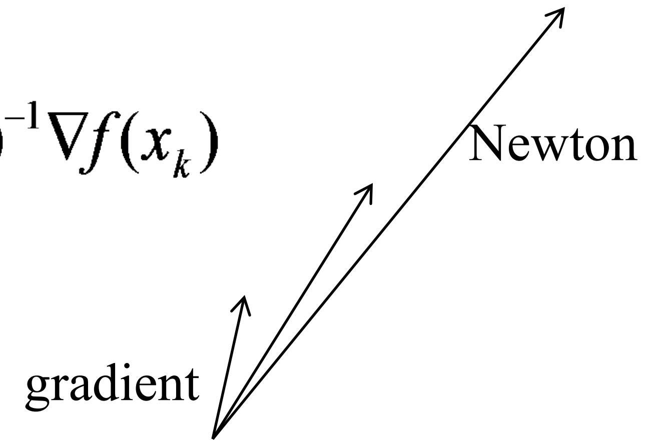
A Fundamental Equation for Newton's method

Strongly convex case (Hessian is positive definite)

$$\nabla^2 f(x_k) = \sum_{i=1}^{n} \lambda_i v_i v_i^T$$ **eigenvalue decomposition**

$$\nabla^2 f(x_k)^{-1} = \sum_{i=1}^{n} \frac{1}{\lambda_i} v_i v_i^T \qquad p = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$$

$$\boxed{p = -\sum_{i=1}^{n} \frac{1}{\lambda_i} v_i (v_i^T \nabla f(x_k))}$$



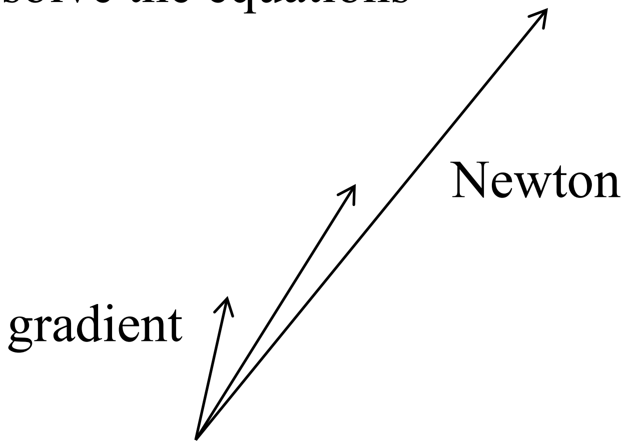
- direction points along eigenvectors corresponding to smallest eigenvalues
- Inexact Newton methods are based on this observation

Inexact Newton Method (Newton-CG)

… and on the important fact that an only matrix-vector products are needed by iterative methods like Conjugate Gradients to solve the equations

$$\nabla^2 f(x_k)p = -\nabla f(x_k)$$

A symmetric positive definite linear system
Many iterative methods; CG considered best
Increasing subspace minimization properties



Nonconvex Case:
Run CG until negative curvature is encountered; follow that direction
Sometimes called the Hessian-Free method in the ML community

Sub-sampled Hessian Newton Methods

Choose $X, S \subset \{1,2..,\}$, uniformly, independently from distribution P

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

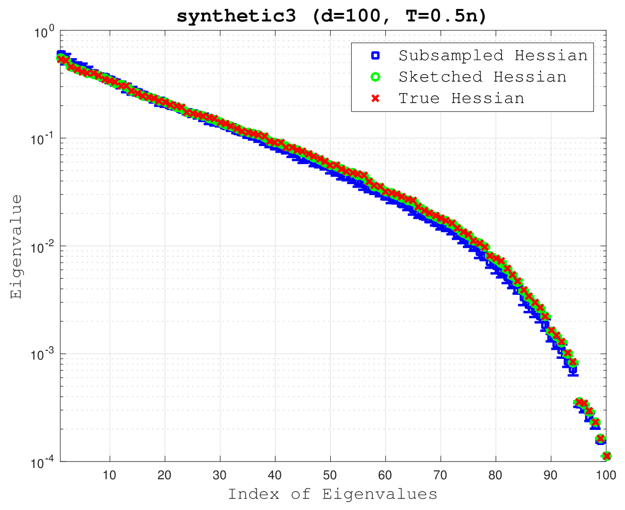
The stochastic nature of the objective creates opportunities:

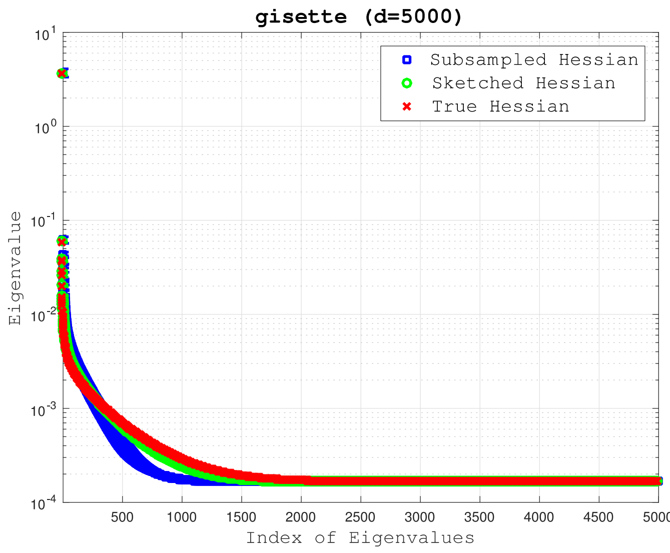
$$\boxed{\nabla^2 F_S(w_k)p = -\nabla F_X(w_k) \qquad w_{k+1} = w_k + \alpha_k p}$$

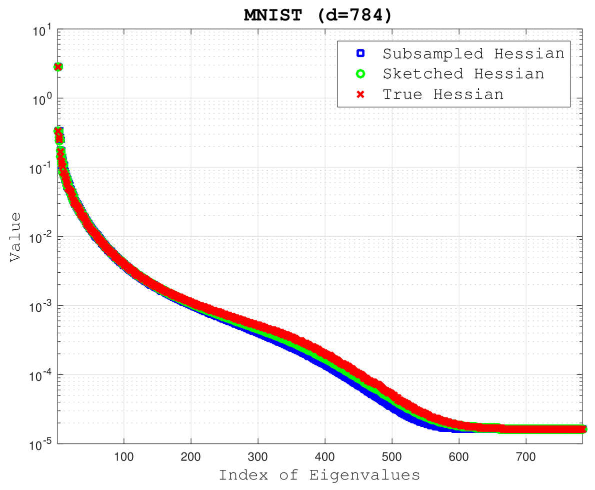
1. Subsampled gradient and Hessian (or other approximations)
2. How to coordinate choice of gradient and Hessian sampling?
3. Inexact solution of linear systems
4. What iterative method to use?
    - Conjugate gradient
    - Stochastic gradient Bullins 2016, Neumann

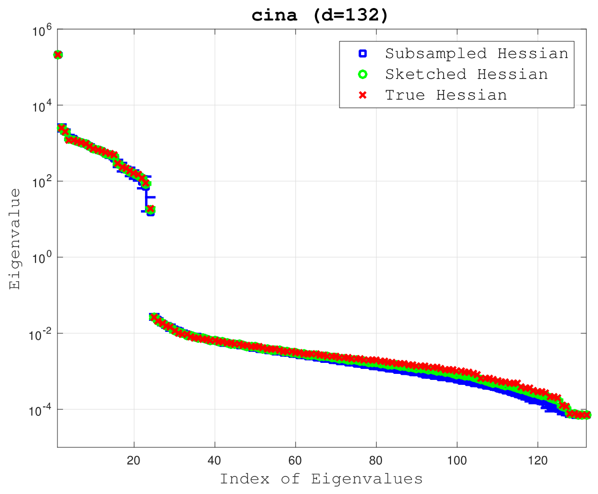
Eigenvalue Distribution of Hessian

Berahas, Bollapragada 2017









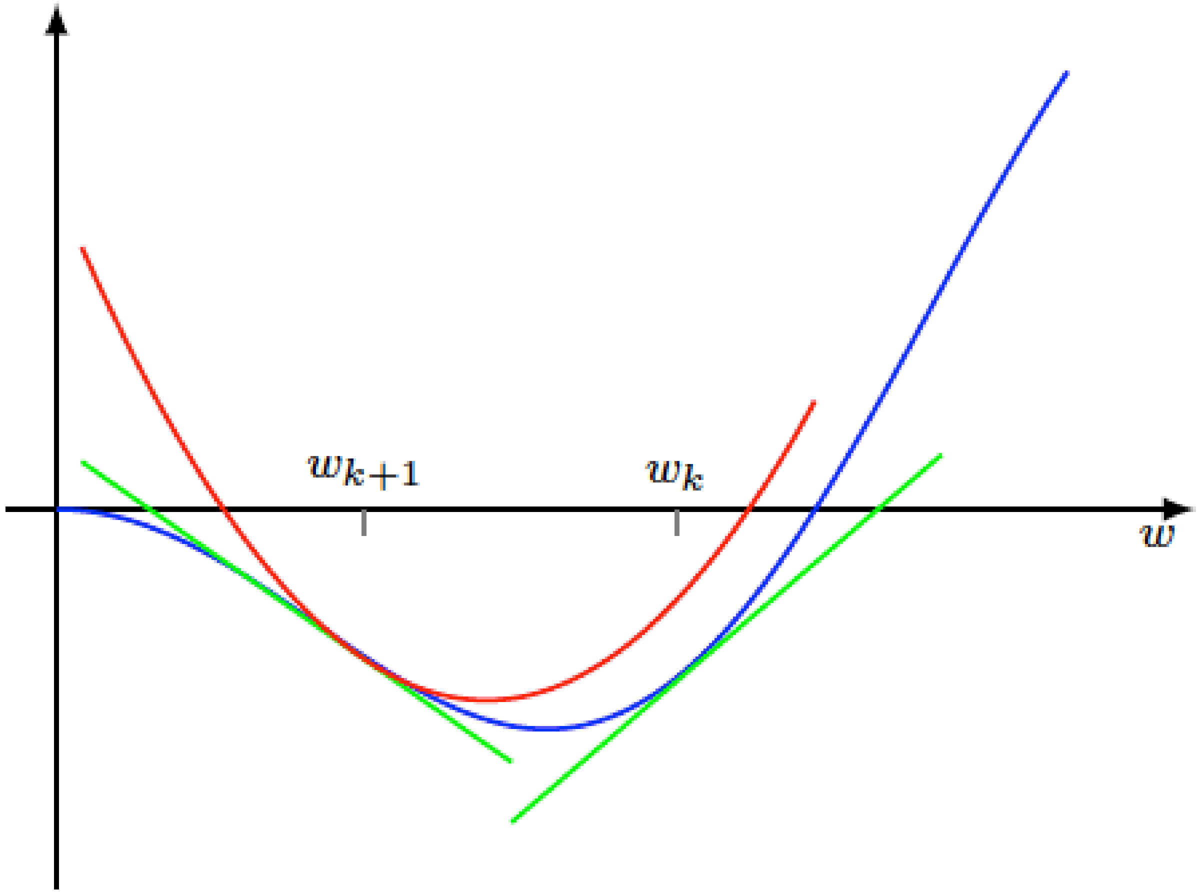
Quasi-Newton methods

A major idea in deterministic optimization

$$w_{k+1} = w_k - \alpha_k H_k \nabla F_{X_k}(w_k)$$

1. Learn curvature of problem on the fly through gradient differences
2. Incorporate curvature information that has been observed
3. Construct a dense Hessian approximation
4. Limited memory version L-BFGS avoids the use of matrices, requires storage and computation of *O(d)*

Only *approximate* second-order information with gradient displacements:



Secant equation $H_k v_k = s_k$ to match gradient of F at w_k, where

$$s_k := w_{k+1} - w_k \quad \text{and} \quad v_k := \nabla F(w_{k+1}) - \nabla F(w_k)$$

The BFGS method

Algorithm:

1. After performing a step, compute:

$$s = w_{k+1} - w_k \qquad y = \nabla F_x(w_{k+1}) - \nabla F_x(w_k)$$

2. $\rho = 1 / y^T s$

3. Update matrix:

$$\mathrm{H_k} = (I - \rho\, y\, s^T) H_{k-1} (I - \rho\, s\, y^T) + \rho\, s\, s^T$$

4. Search direction and iteration:

$$d_k = -H_k \nabla F_X(w_k) \qquad w_{k+1} = w_k + \alpha_k d_k$$

Progressive Sampling & Quasi-Newton Updating

Bollapragada, Shi 2018

$$w_{k+1} = w_k - \alpha_k H_k \nabla F_{X_k}(w_k)$$

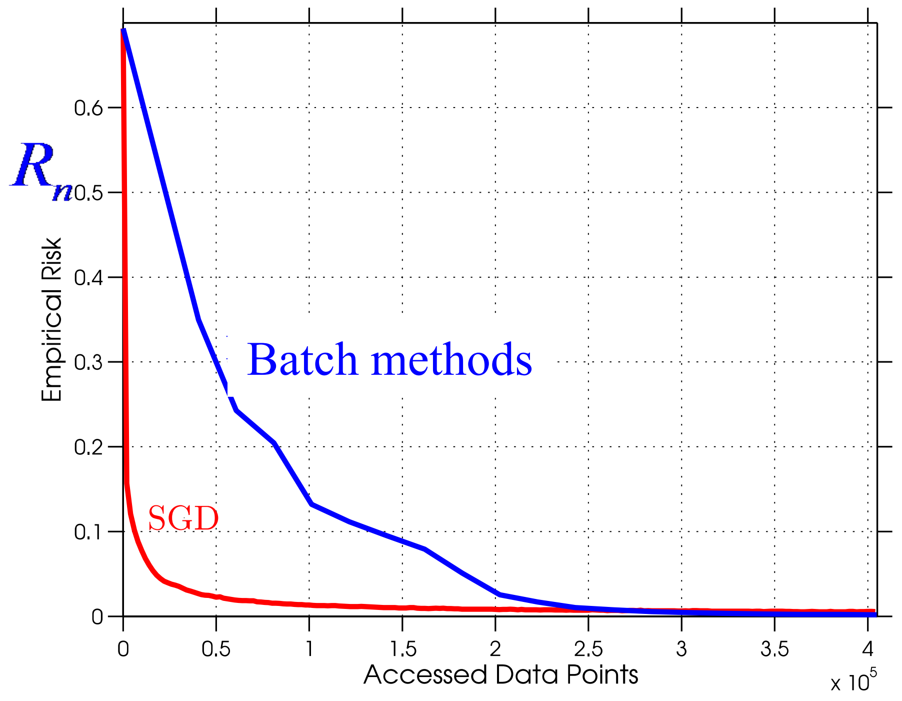
H_k updated by a careful (fault tolerant) form the the limited memory BFGS method

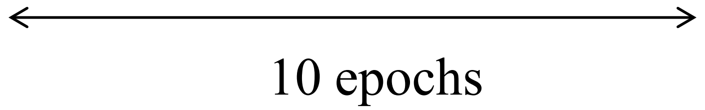
Line Search: Relaxing the sufficient decrease condition

$$F_{X_k}(w_k + \alpha_k p_k) \le F_{X_k}(w_k) + c_1 \alpha_k \nabla F_{X_k}(w_k)^T p_k + \epsilon_k$$

where ϵ_k is the noise level in the function

For years we observed this

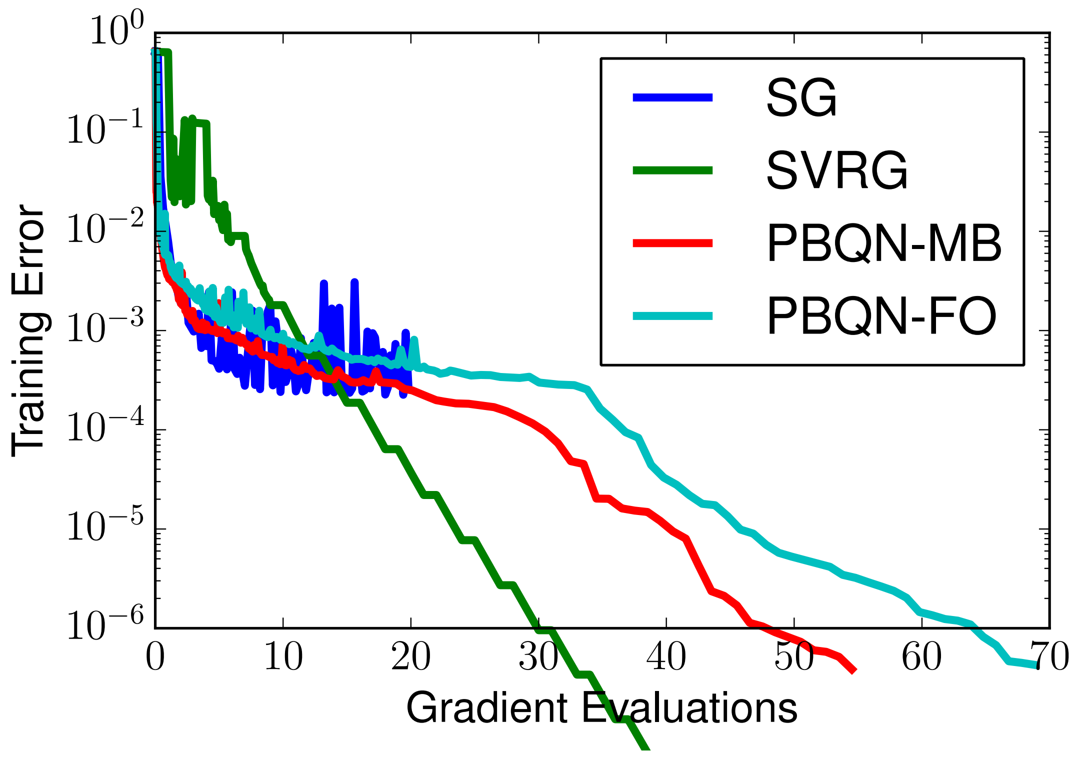




Logistic regression;
speech data

Fast initial progress
of SG followed by drastic
slowdown

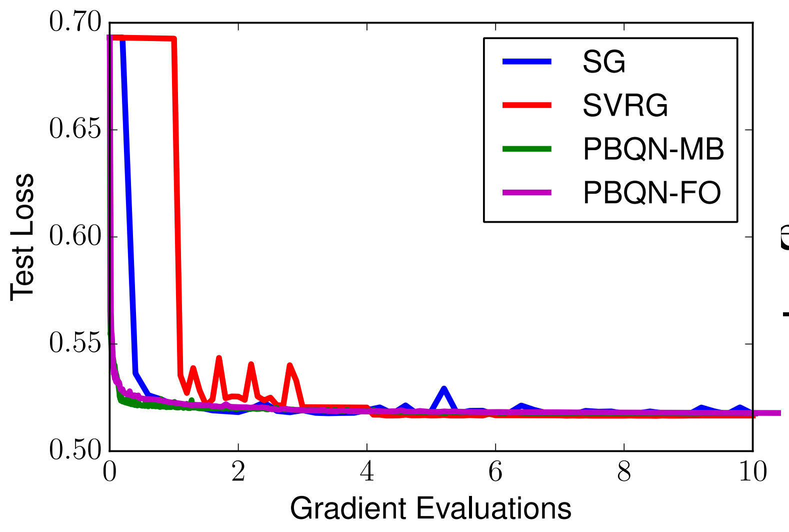
New Results: Progressive Sampling Quasi-Newton



Logistic Regression

- Results for DNN, in progress

Tests: Logistic Regression- Test Error



essive batching

-Newton method

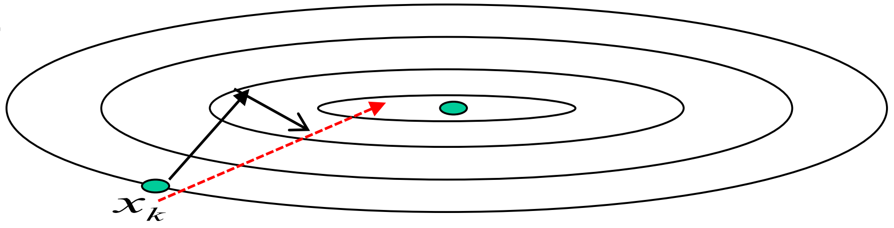
- Stochastic quasi-Newton methods with noisy gradients in the typical regime of the SG method have not proved effective.
- Bollapragada, Shi et al (2018) have shown that a surprisingly small batch (100, 200) offers opportunities for quasi-Newton methods

An Open Question: Momentum and Acceleration

- It appears to be quite popular for training deep neural
- But acceleration & momentum: not useful in deterministic optimization – where they originated
- Instead employ some 2nd order information using gradient differences
- Quasi-Newton and inexact Newton methods

Momentum (Heavy Ball Method)

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



Beware of 2-d pictures!

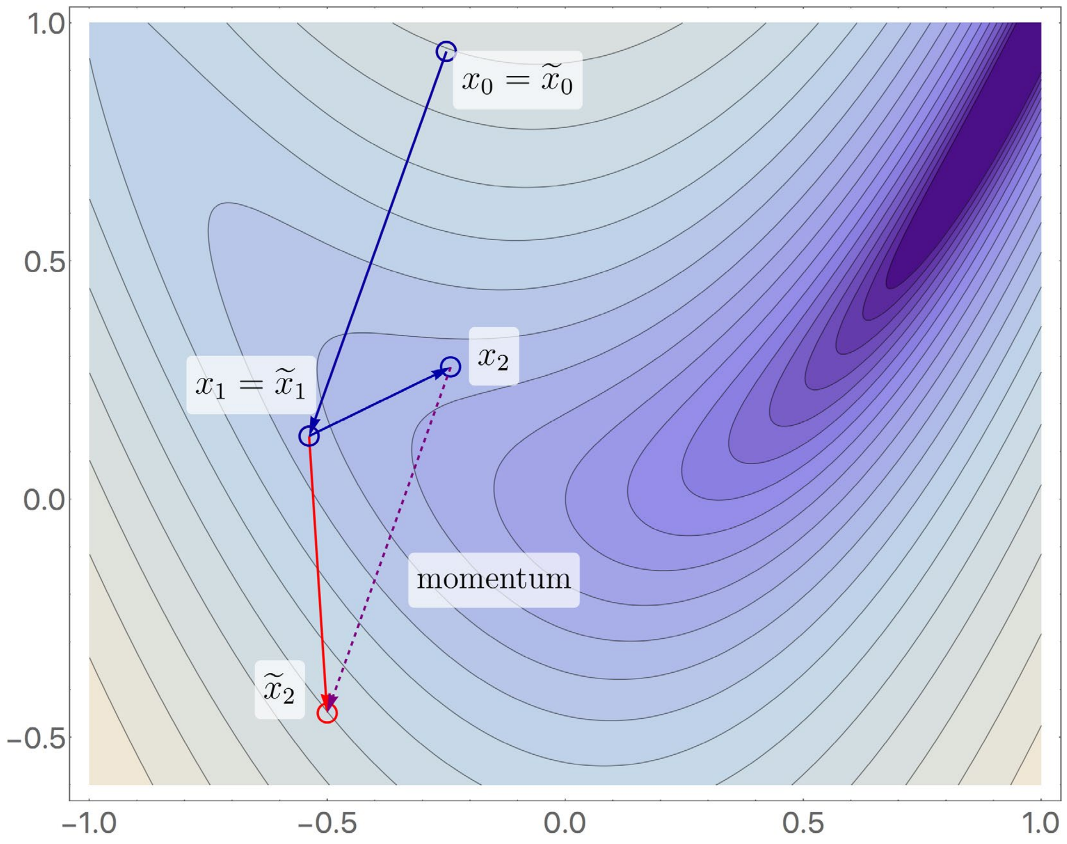
It is true that for convex quadratics the gradient method with momentum has a faster convergence rate than the pure gradient method
But:

- One needs a good estimate of the condition number of the Hessian

$$\alpha = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2} \qquad \beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$

- DNN are not quadratics!
- Gradient method + momentum is not convergent on convex functions
- There are better iterative methods (CG) for quadratics

Consider what momentum can do in the non-convex case



Gradient method with momentum; $\beta = 0.9$

But momentum works in practice $w_{k+1} = w_k - \alpha_k \nabla F(w_k) + \beta_k (w_k - w_{k-1})$

- Popular since (Sutskever et al. 2013)
- Conjecture: it is not a real momentum method; neither a linear dynamical system with friction, nor Nesterov's optimal iteration
- Instead: a form of iterate (or gradient) averaging

$$\hat{w}_k = \sigma \hat{w}_{k-1} + (1-\sigma) w_k$$

- Gap between practice and algorithmic understanding
- Useful to compare with the Conjugate Gradient method

$$w_{k+1} = w_k + \alpha_k p_k \qquad p_k = -\nabla F(w_k) + \beta_k p_{k-1}$$

Designed for quadratic objective functions; easy to compute parameters
Same form as momentum but requires no estimate of condition number
For deterministic quadratic problems momentum is not better than CG
A version for nonlinear problems is available (PR+; see my website)

Nesterov acceleration

$$x_{k+1} = y_k - \alpha_k \nabla F(y_k)$$
$$y_{k+1} = x_k + \beta_k (x_{k+1} - x_k)$$

Remarkable result:

- If eigenvalue information is available
- Rate of convergence is $O((1 - \frac{1}{\sqrt{\kappa}})^k)$

- But is it relevant to practice? FISTA
  - Even for convex problems, can it compete with quasi-Newton method?
  - Suppose estimate of condition number is not accurate
- Many complexity papers on acceleration:
  - Find a stationary point, escaping saddles, combine with other methods, etc.
  - Very pessimistic results
  - My view: need to move away from the complexity perspective and study smoothing & acceleration in a new statistical setting

End