IPAM Summer School 2012

Tutorial on

Optimization methods for machine learning

Jorge Nocedal

Northwestern University

Overview

1. We discuss some characteristics of optimization problems arising in deep learning, convex logistic regression and inverse covariance estimation.

2. There are many tools at our disposal: first and second order methods; batch and stochastic algorithms; regularization; primal and dual approaches; parallel computing

3. Yet the state of affairs with neural nets is confusing to me: too many challenges are confronted at once: local vs local minima, nonlinearity, stochasticity, initializations, heuristics

Open Questions

We need to isolate questions related to optimization, and study them in a controlled setting

A key question is to understand the properties of stochastic vs batch methods in the context of deep learning.

After some clarity is obtained, we need to develop appropriate algorithms and work complexity bounds, both in a sequential and a parallel setting

Organization

I will discuss a few of optimization techniques and provide some insights into their strengths and weaknesses

We will contrast the deterministic and stochastic settings. This motivates dynamic sample selection methods

We emphasize the need for general purpose techniques, second order methods and scale invariance, vs heuristics

My Background

Most of my practical experience in optimization methods for machine learning is for speech recognition (Google)

But I am aware of many tests done in a variety of machine learning applications due to my involvement in L-BFGS, Newton-CG methods (Hessian-free), dynamic sampling methods, etc

I am interested in designing new optimization methods for machine applications, in particular for deep learning

Intriguing Statements

‘The best optimization algorithms are not the best learning algorithms’

Bottou and Bousquet (2009)

…when taking into account the Estimation and Optimization Errors

``Stochastic methods best in spite of being worst optimization methods”

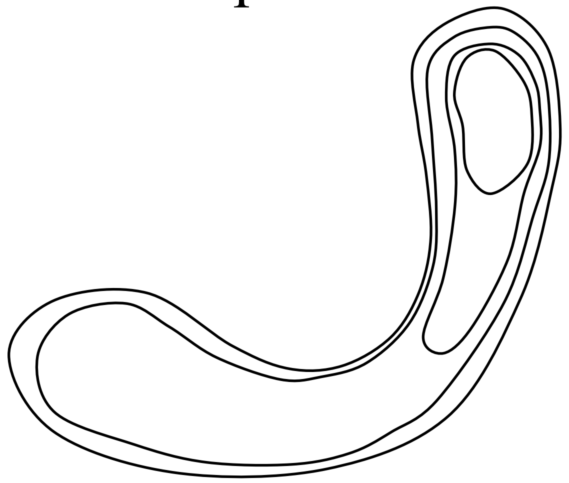
Part I

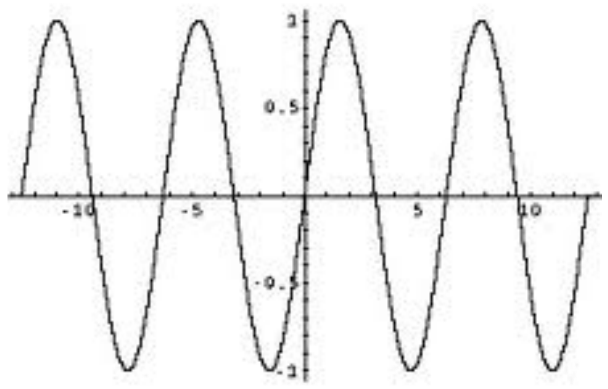
Problem Characteristics and Second Order Methods

Nonlinearity, Ill Conditioning

Neural nets are far more nonlinear than the functions minimized in many other applications Farabet

The rate of convergence of an optimization algorithm is still important even though in practice one stops the iteration far from a minimizer”



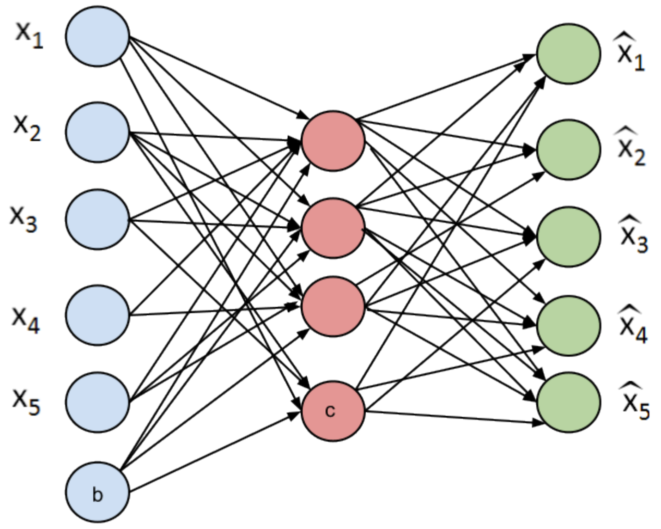


An objective function (Le,…Ng)

$$\min_{W,b,c} \quad \frac{\lambda}{m}\sum_{i=1}^{m} \|\sigma(W^T\sigma(Wx^{(i)}+b)+c)-x^{(i)}\|_2^2 + S(W,b,x^{(1)},\cdots,x^{(m)})$$

where $\sigma(\cdot)$ is the activation function, b and c are the biases, and $S(\cdot)$ is some sparse penalty function *(ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning:*

sparse autoencoder



General Formulation

$$\boxed{\min \; J(w) = \frac{1}{m}\sum_{i=1}^{m} \ell(w;(z_i, y_i)) + \nu \| w \|_1}$$

(z_i, y_i) training data

z_i vector of features; y_i labels

Loss function (logistic, least squares, etc): $\ell(w;(z_i, y_i))$

Ignore regularization term at first: unconstrained optimization

$$\boxed{\min \; f(x)}$$

The Newton-CG Method (Hessian Free)

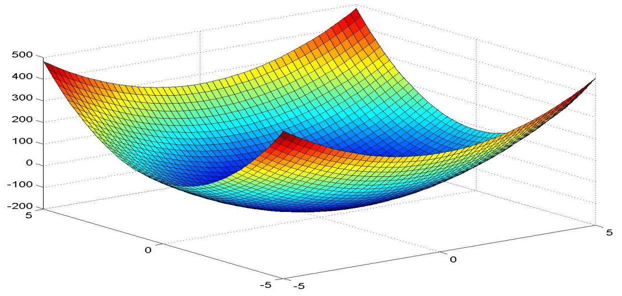
Problem: $\min f(x)$

$$\boxed{\nabla^2 f(x_k)p = -\nabla f(x_k) \qquad x_{k+1} = x_k + \alpha p}$$

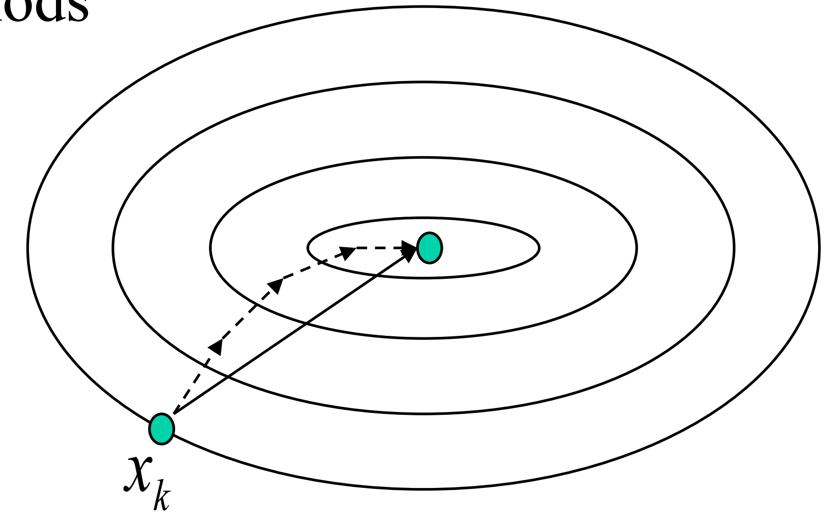
- This is a linear system of equations of size n
- Apply the Conjugate Gradient (CG) method to compute an approximate solution to this system
- CG is an iterative method endowed with very interesting properties (optimal Krylov method)
- Hessian need not be computed explicitly
- Continuum between 1st and 2nd order methods

Newton-CG: the Convex Case

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



- We show below that
  any number of CG iterations yield a productive step
- Not true of other iterative methods
- better direction and length
  than 1st order methods



The conjugate gradient method

Two equivalent problems

$$\min\ \phi(x) = \frac{1}{2} x^T A x - b^T x \qquad \nabla\phi(x) = Ax - b$$

$$\text{solve} \quad Ax = b \qquad r = Ax - b$$

$$p_k = -r_k + \beta_k p_{k-1}$$

$$\beta_k = \frac{p_{k-1}^T A r_k}{p_{k-1}^T A p_{k-1}}$$

$$x_{k+1} = x_k + \alpha_k p_k,$$

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$$

Only product Ap_k is needed

Hessian-free

Choose some initial point: x_0

Initial direction: $p_0 = -r_0$

For $x_0 = 0, \quad -r_0 = b$

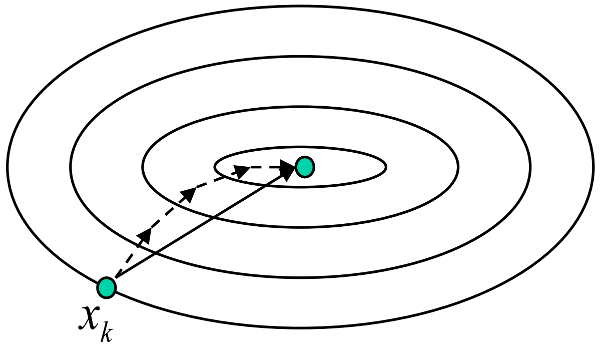
Interaction between CG and Newton

We noted $-r = b$ if $x_0 = 0$

For the linear system

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

$$r = Ax - b \quad \rightarrow \quad b = -\nabla f(x_k)$$



Conclusion: if we terminate the CG algorithm after 1 iteration we obtain a steepest descent step

Newton-CG Framework

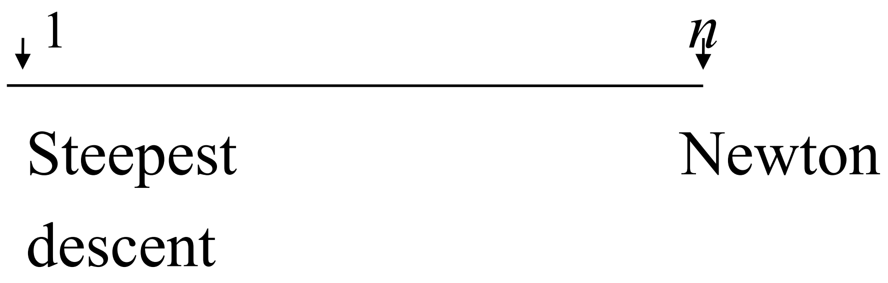
Theorem (Newton-CG with any number of CG steps)

Suppose that f is strictly convex. Consider the iteration

$$\nabla^2 f(x_k)p = -\nabla f(x_k) + r \qquad x_{k+1} = x_k + \alpha p$$

where α is chosen by a backtracking Armijo line search.

Then $\{x_k\} \to x_*$



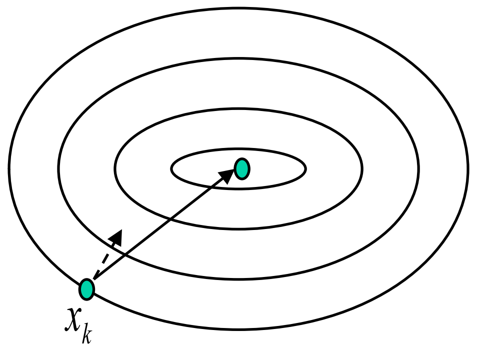
Rates of Convergence – Scale Invariance

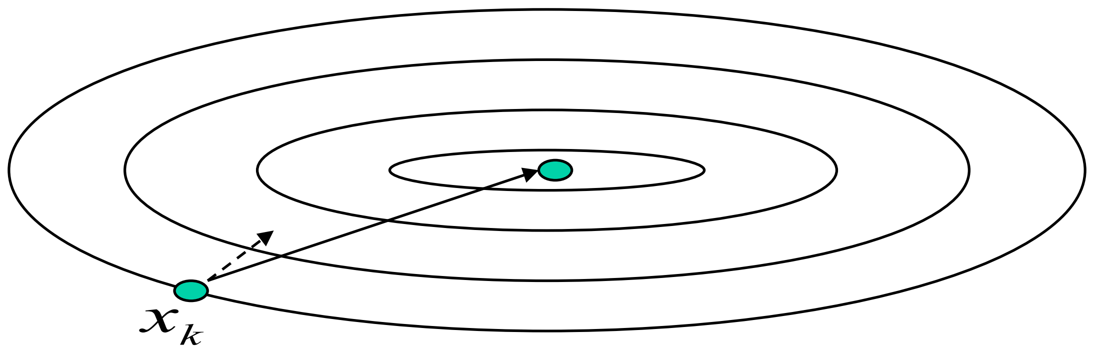
The rate of convergence can be:

linear   superlinear   quadratic

depending on the accuracy of the CG solution

- It inherits some of the scale invariance properties of the exact Newton method: affine change of variables $x \leftarrow Dx$





Newton-CG– The Nonconvex Case

If Hessian is not positive definite solve modified System

$$[\nabla^2 f(x_0) + \gamma I]\, p = -\nabla f(x_0) \qquad \gamma > 0$$

If γ is large enough system is positive definite

Let $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$ be the eigenvalues of $\nabla^2 f(x_0)$.

Then the eigenvalues of $[\nabla^2 f(x_0) + \gamma I]$ are:

$$\lambda_1 + \gamma \le \lambda_2 + \gamma \le \ldots \le \lambda_n + \gamma$$

Difficult to choose γ. Trust region method learns γ

The Nonconvex Case: Alternatives

Replace $\nabla^2 f(x_k)$ by a positive definite approximation

$$Bp = -\nabla f(x_0)$$

Option 1: Gauss-Newton Matrix $J(x_k)J(x_k)^T$

Option 2: Stop CG early - negative curvature

Option 3: Trust region approach

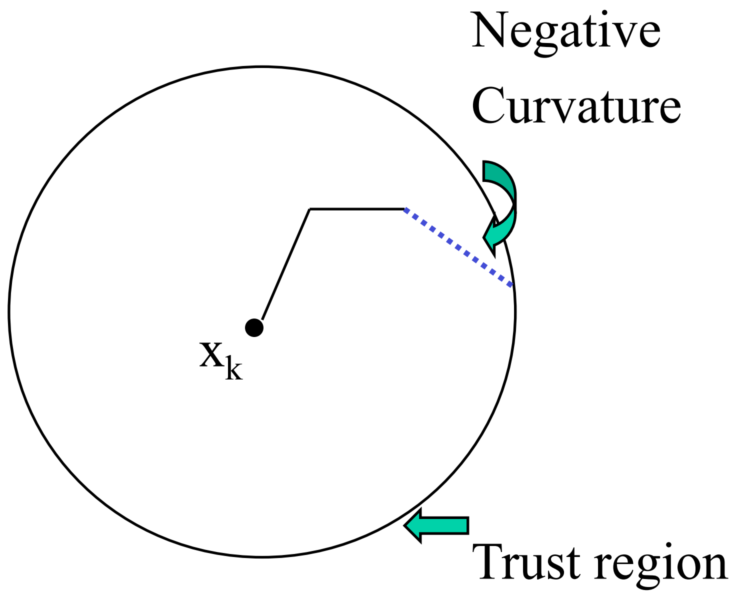
For least squares Gauss-Newton matrix can be seen as an optimal Inexact CG equivalent to solving in a “subspace inverse”

The Nonconvex Case: CG Termination

$$\nabla^2 f(x_k)p = -\nabla f(x_0)$$

Iterate until negative curvature is encountered:

$$v^T \nabla f(x_k) v < 0$$



$$\begin{array}{ll} \min & q(d) = d^T \nabla^2 f(x_k) d + \nabla f(x_k)^T d + f(x_k) \\ \text{s.t.} & \| d \| \le \Delta \end{array}$$

History of Newton-CG

1. Proposed in the 1980s for unconstrained optimization and systems of equations (Polyak 1960 (bounds))
2. Hessian-free option identified early on
3. Trust region (1980s): robust technique for choosing γ
4. Considered today a premier technique for large problems (together with nonlinear CG and L-BFGS)
5. Used in general nonlinear programming: Interior Point, Active Set, Augmented Lagrangian methods
6. Application to stochastic problems (machine learning)

   Martens (2010), Byrd, Chin, Neveitt, Nocedal (2011)

Newton-CG and global minimization

1. I know of no argument to suggest that Newton-like methods are better able at locating lower minima than 1st order methods
2. Some researchers report success with “Hessian-free methods”
   Martens (2010). Algorithms plagued with heuristics
3. Trust region methods should be explored
4. Properties of objective functions should be understood: do we want to locate global minimizer?
5. More plausible for stochastic gradient descent

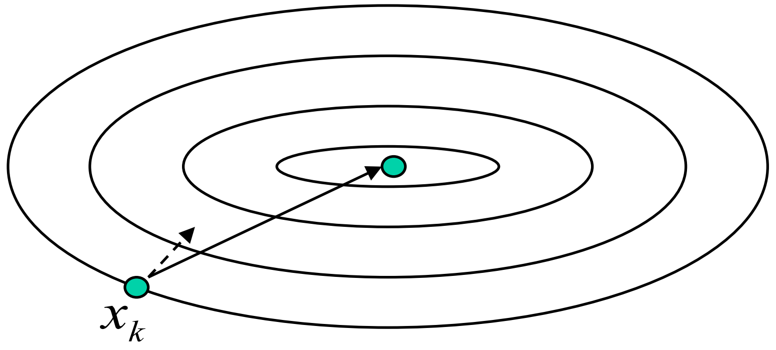
Understanding Newton's Method

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

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

$$\nabla^2 f(x_k) p = -\nabla f(x_k) \;\rightarrow\; p = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$$

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



- direction points along

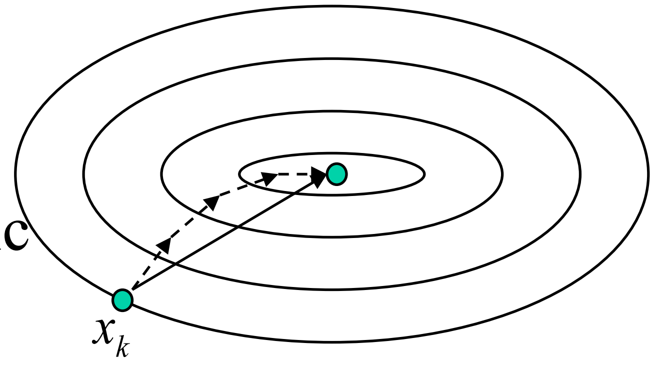
  eigenvectors corresponding to smallest eigenvalues

-- get direction and length

Inexact Newton's Method

If we can compute the Newton direction by gradually minimizing The quadratic, we might obtain efficiency and regularization (steplength control)

Therefore we need to explicitly consider the minimization of a quadratic model of the objective function f



$$\min \quad q(d) = d^T \nabla^2 f(x_k) d + \nabla f(x_k)^T d + f(x_k)$$

Enter the Conjugate Gradient method

$$\min \;\; \phi(d) = d^T \nabla^2 f(x_k) d + \nabla f(x_k)^T d + f(x_k)$$

exact solution (convex case)

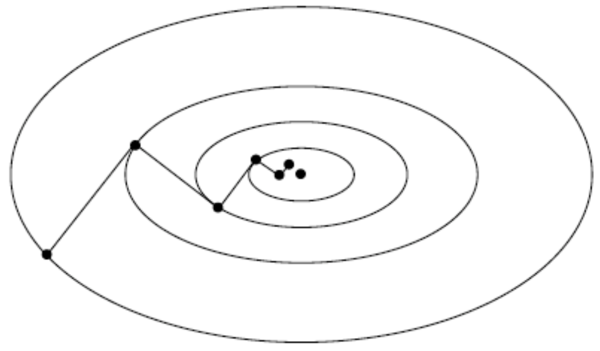
$$\nabla \phi(d) = \nabla^2 f(x_k) d + \nabla f(x_k) = 0 \qquad \text{Newton step}$$

- The (linear) CG method is an iterative method for solving linear positive definite systems or minimizing the corresponding quadratic model.

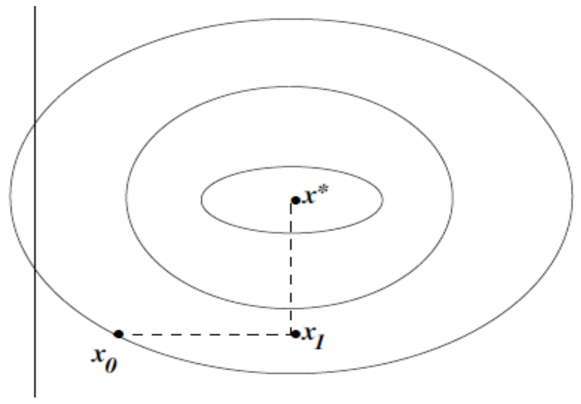
- Key property: expanding subspace minimization

Tends to minimize first along the largest eigenvectors

$$A = \nabla^2 f(x_k) \qquad b = -\nabla f(x_k)$$



Steepest descent

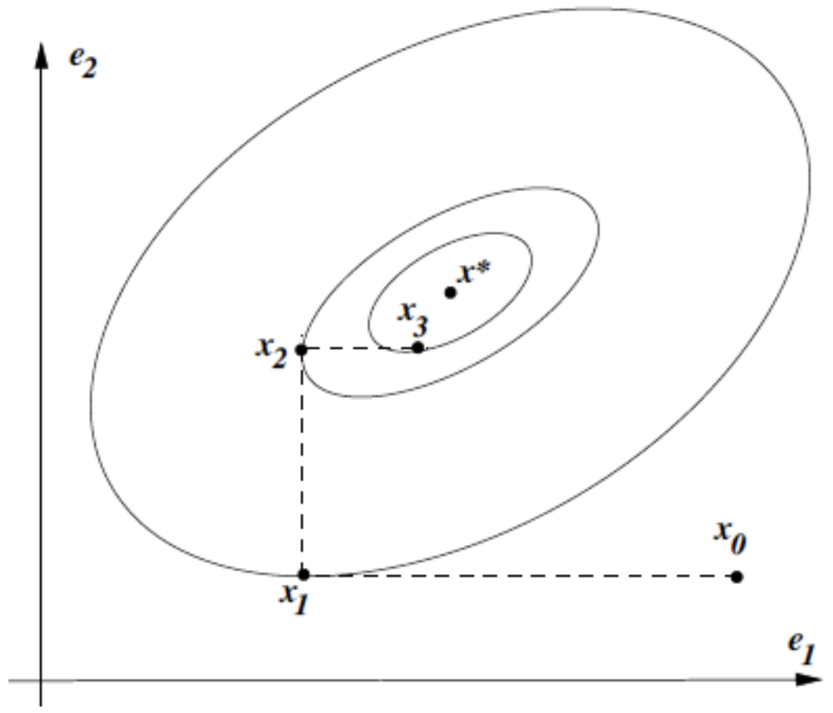


Coordinate relaxation

$$\min \phi(x)$$

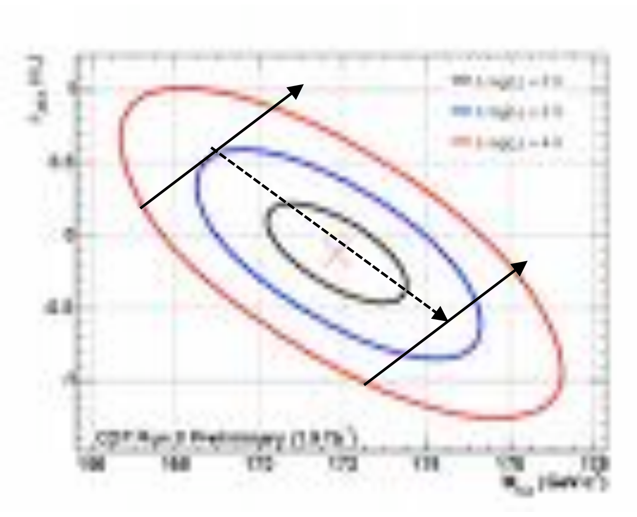
Coordinate relaxation does not terminate in n steps!

The axis are no longer aligned with the coordinate directions



Conjugate directions always
 work

(lead to solution in n steps)

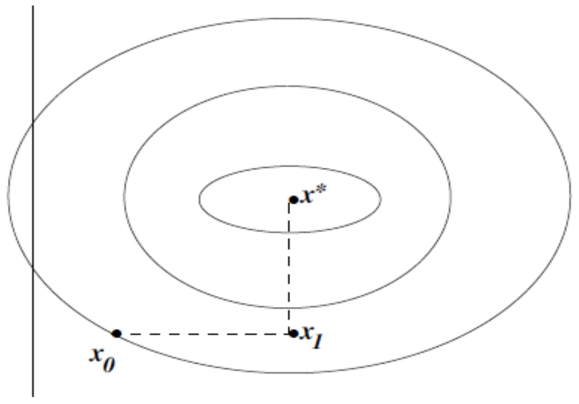


Expanding Subspace Minimization

Each coordinate minimization determines 1 component of the solution.

Thus we minimize over an expanding subspace and we must have

$$\nabla \phi(x_k)^T p_i = 0 \qquad i = 0,1,\ldots,k-1$$



Or equivalently

$$r_k^T p_i = 0 \qquad i = 0,1,\ldots,k-1$$

Steepest descent does not have this property.

There are many conjugate direction methods. One of them is special….

Monotonicity

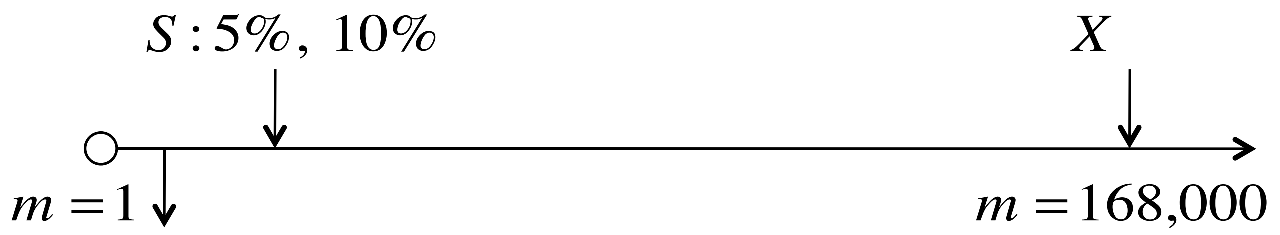
Theorem: Suppose that the CG method is started at zero. Then the approximate solutions satisfy $\| d^0 \| \leq \ldots \leq \| d^i \|$

This provides regularization of the step

-Another choice used in practice is to start the CG method with the solution obtained at the previous iteration (monotonicity lost)

Hessian Sub Sampling

Hessian Sub-Sampling for Newton-CG



Function, gradient: large sample X (batch)
Curvature information: small sample S

- Newton-like methods very robust with respect to choice of Hessian

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

Stochastic Optimization Problem: (J(w)

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} \ell(w;(z_i, y_i))$$

Choose random sample of training points X

$$\boxed{J_X(w) = \frac{1}{|X|}\sum\nolimits_{i \in X} l(w;(z_i, y_i))}$$

whose expectation is $J(w)$

very small X : online, stochastic

large X : batch

A sub-sampled Hessian Newton method

Choose subsample $S \subset X \Longrightarrow \nabla^2 J_S(w) = \sum_S \nabla^2 \ell(w; z_i, y_i)$

$$\boxed{\nabla^2 J_S(w_k) d_k = -\nabla J_X(w_k) \qquad w_{k+1} = w_k + d_k}$$

- Coordinate size of subsample with number of CG steps
- Example: S=5% and 10 CG steps
- total step computation ~ 1 function evaluation
- Similar in cost to steepest descent … but much faster
- Experiments with logistic function: unit step acceptable

Hessian-vector Product without Computing Hessian

Given a function $f : R^n \to R$ and a direction d

Goal: compute $\nabla^2 f(x_k)d$ exactly

Define the function $\Phi(x;d) = \nabla f(x)^T d$

$$\Rightarrow \quad \boxed{\nabla_x \Phi(x;d) = \left(\frac{\partial \nabla f(x)^T d}{\partial x} \right) = \nabla^2 f(x)^T d}$$

Example

$$f(x) = \exp(x_1 x_2)$$

$$\nabla\Phi(x;v) = \nabla f(x)^T v = \left(x_2 \exp(x_1 x_2)\right) v_1 + \left(x_1 \exp(x_1 x_2)\right) v_2$$

$$\nabla_x^2 \Phi(x) v = \begin{bmatrix} \left(x_2^2 \exp(x_1 x_2)\right) v_1 + \left(\exp(x_1 x_2) + x_1 x_2 \exp(x_1 x_2)\right) v_2 \\ \left(\exp(x_1 x_2) + x_1 x_2 \exp(x_1 x_2)\right) v_1 + \left(x_1^2 \exp(x_1 x_2)\right) v_2 \end{bmatrix}$$

Cost of Hessian-vector product comparable to cost of one gradient (factor of 3-5)

Logistic Regression

$$J_X(w) = \sum_{i \in X} l(w;(z_i, y_i)) \quad \Rightarrow$$

$$\boxed{\nabla^2 J_X(w) d = \sum_{i \in X} h(w;(z_i, y_i)) P(i) d}$$

- Cost of Hessian-vector product decreases linearly with |X|
- Hessian-vector product parallelizes, same as function

The Algorithm

Function sample X given (and fixed)

```
Choose subsample $S_k$   $(|S_k| << |X|)$
Solve
    $\nabla^2 J_S(w_k) d_k = -\nabla J_X(w_k)$
by Hessian-free CG method
    $w_{k+1} = w_k + \alpha_k d_k$        (Armijo)
Resample $S_{k+1}$   $(|S_{k+1}| << |X|)$
```

Rather than one algorithm, this is general technique;

Can derive sub-sampled L-BFGS method

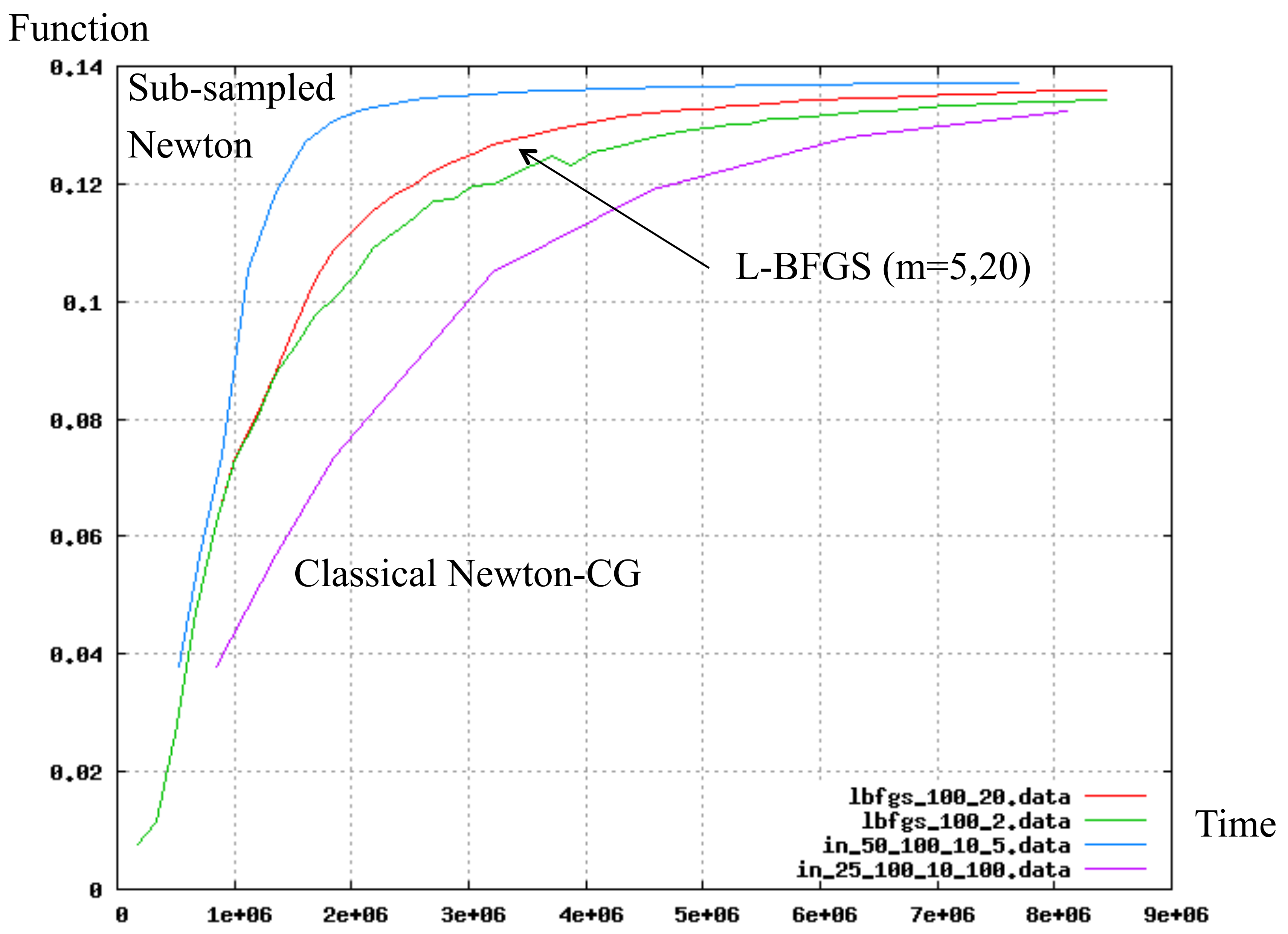
Byrd, Chin, Neveitt, Nocedal (2011)

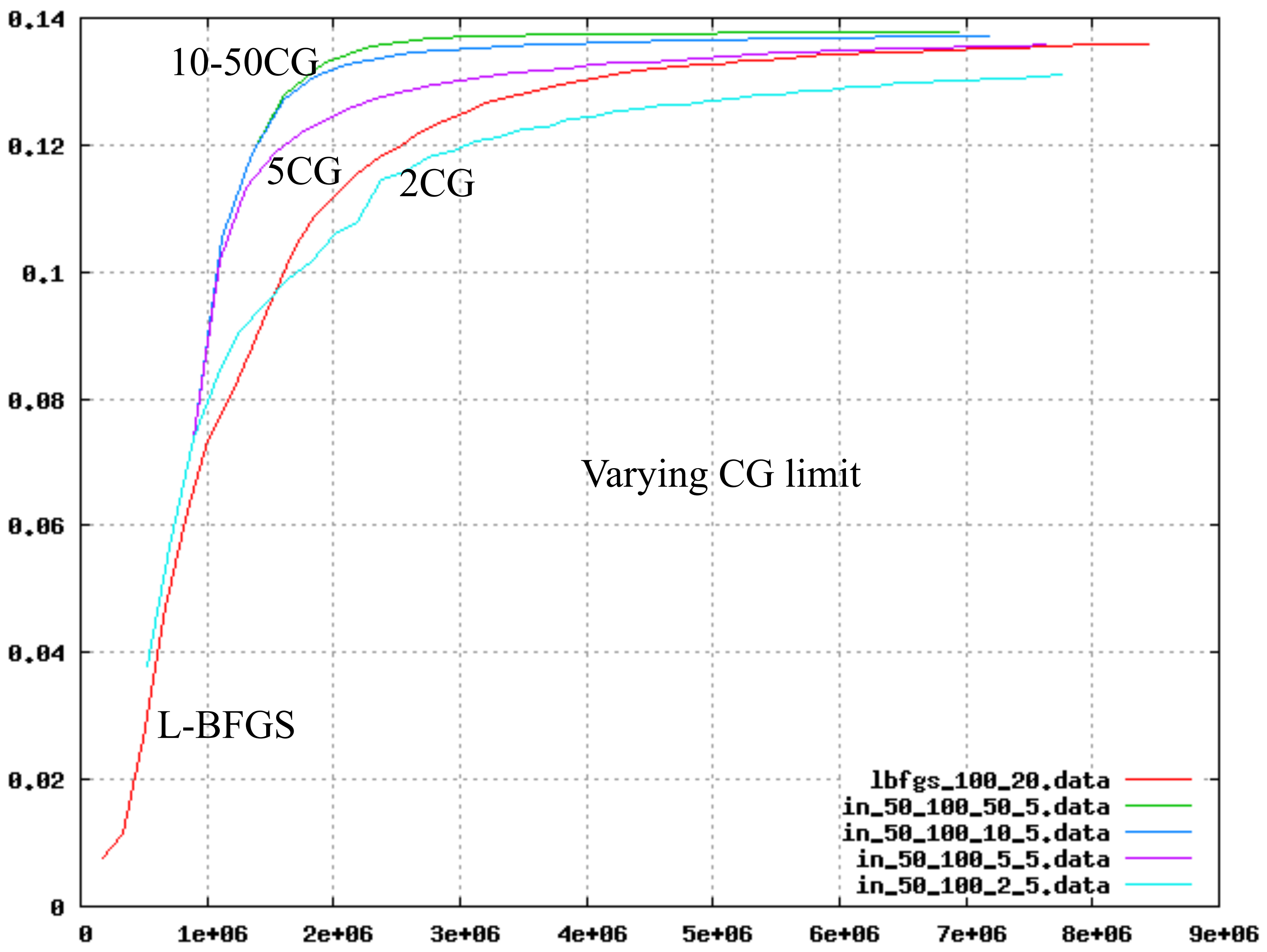
Speech Recognition Problem

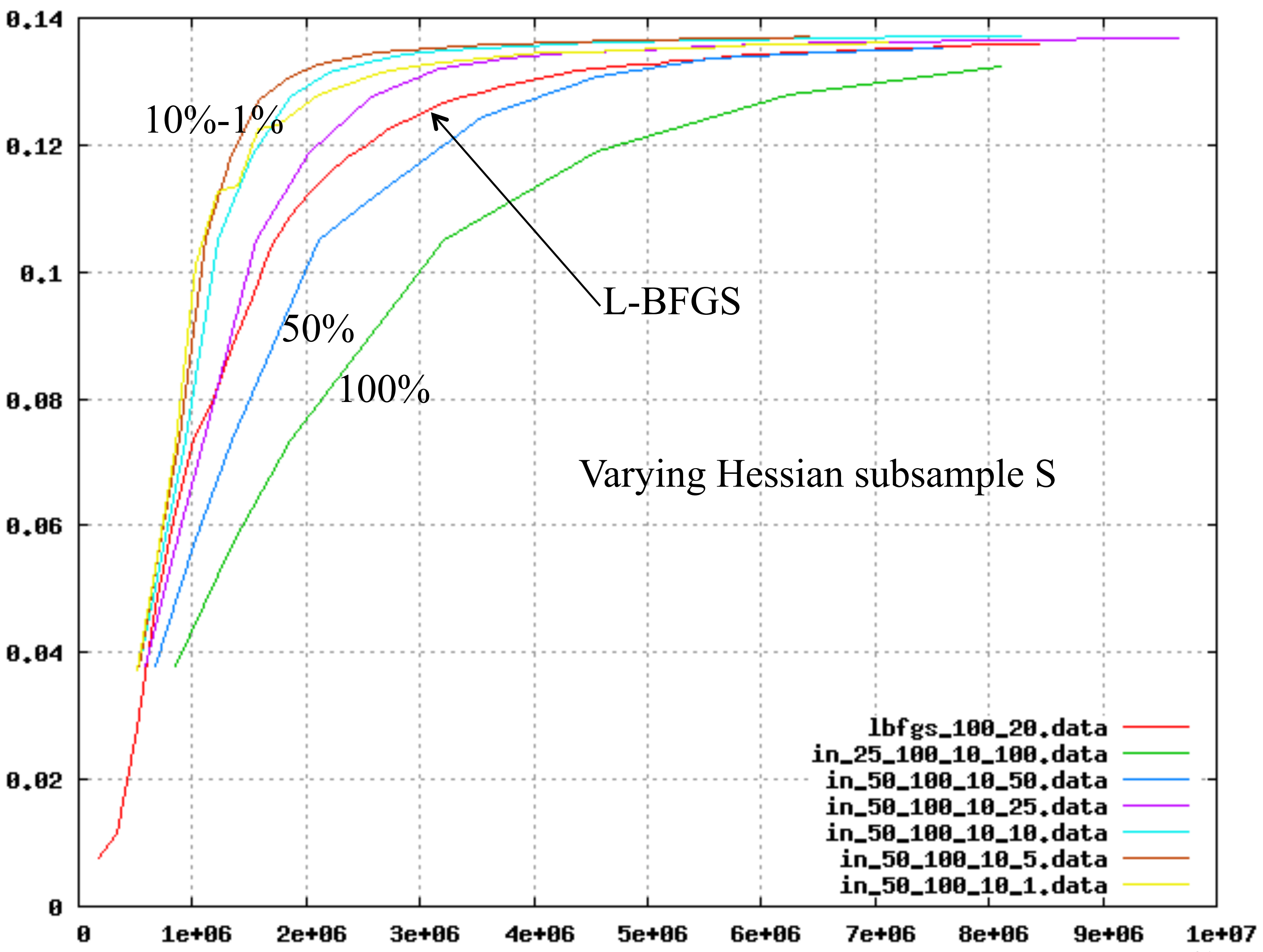
Characteristics:

- 168,000 training points
- 10,100 parameters (variables)
- Hessian subsample: 5%
- Solved on workstation

$$J(w) = \sum_{h=1}^{N} \ln \sum_{i=0}^{NC} \exp(\sum_{j=1}^{NF} w_{ij} f_{hj}) - \sum w_{c_h^* j} f_{hj}$$


Function
Time
Sub-sampled Newton
L-BFGS (m=5,20)
Classical Newton-CG
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1e+06
2e+06
3e+06
4e+06
5e+06
6e+06
7e+06
8e+06
9e+06
lbfgs_100_20.data
lbfgs_100_2.data
in_50_100_10_5.data
in_25_100_10_100.data


10-50CG
5CG
2CG
Varying CG limit
L-BFGS
lbfgs_100_20.data
in_50_100_50_5.data
in_50_100_10_5.data
in_50_100_5_5.data
in_50_100_2_5.data
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1e+06
2e+06
3e+06
4e+06
5e+06
6e+06
7e+06
8e+06
9e+06



0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
1e+06
2e+06
3e+06
4e+06
5e+06
6e+06
7e+06
8e+06
9e+06
1e+07
10%-1%
50%
100%
L-BFGS
Varying Hessian subsample S
lbfgs_100_20.data
in_25_100_10_100.data
in_50_100_10_50.data
in_50_100_10_25.data
in_50_100_10_10.data
in_50_100_10_5.data
in_50_100_10_1.data

Summary of results

Probability	speedup
10%	1.7
12%	2.0
13.5%	4.2

Preconditioning?

How many CG iterations?

For quadratic with Gaussian noise, how to relate number of CG iterations to noise level?

Algorithmic solution.

$$\boxed{\nabla^2 J_S(w_k) d_k = -\nabla J_X(w_k) + r_k}$$

$$\begin{aligned} r_k^X &= \nabla^2 J_X(w_k) d_k + \nabla J_X(w_k) \\ &= \underbrace{\nabla^2 J_S(w_k) d_k + \nabla J_X(w_k)}_{\text{Iteration residual}} + \underbrace{[\nabla^2 J_X(w_k) + \nabla^2 J_S(w_k)] d_k}_{\text{Hessian error}} \end{aligned}$$

Implementation

After every matrix-vector product compute

$$s\{\nabla^2 J_S(w_k)p_k\}/\| p_k \|$$

As an estimate to

$$r_k^X = \underbrace{\nabla^2 J_S(w_k)d_k + \nabla J_X(w_k)}_{\text{Iteration residual}} + \underbrace{[\nabla^2 J_X(w_k) - \nabla^2 J_S(w_k)]d_k}_{\text{Hessian error}}$$

Set CG stop test to balance errors, use sample variance

Convergence - Scale Invariance

$$\nabla^2 J_S(w_k) d_k = -\nabla J_X(w_k) \qquad w_{k+1} = w_k + \alpha_k d_k$$

Theorem: Suppose loss function *l(w)* is strictly convex. For any subsample size |S| and for any number of CG steps

$$w_k \rightarrow w^*$$

- Newton-like method? 1st order method?
- Scale invariant: $x \leftarrow Ax$
- $\alpha_k = 1$ at almost all iterations