Truncated Newton Method

- approximate Newton methods
- truncated Newton methods
- truncated Newton interior-point methods

Prof. S. Boyd, EE364b, Stanford University
Newton’s method

• minimize convex $f : \mathbb{R}^n \to \mathbb{R}$

• Newton step $\Delta x_{nt}$ found from (SPD) Newton system

$$\nabla^2 f(x) \Delta x_{nt} = -\nabla f(x)$$

using Cholesky factorization

• backtracking line search on function value $f(x)$ or norm of gradient $\|\nabla f(x)\|

• stopping criterion based on Newton decrement $\lambda^2/2 = -\nabla f(x)^T \Delta x_{nt}$
  or norm of gradient $\|\nabla f(x)\|$
Approximate or inexact Newton methods

- use as search direction an approximate solution $\Delta x$ of Newton system

- idea: no need to compute $\Delta x_{nt}$ exactly; only need a good enough search direction

- number of iterations may increase, but if effort per iteration is smaller than for Newton, we win

- examples:
  - solve $\hat{H}\Delta x = -\nabla f(x)$, where $\hat{H}$ is diagonal or band of $\nabla^2 f(x)$
  - factor $\nabla^2 f(x)$ every $k$ iterations and use most recent factorization
Truncated Newton methods

- approximately solve Newton system using CG or PCG, terminating (sometimes way) early

- also called *Newton-iterative methods*; related to limited memory Newton (or BFGS)

- total effort is measured by cumulative sum of CG steps done

- for good performance, need to tune CG stopping criterion, to use just enough steps to get a good enough search direction

- less reliable than Newton’s method, but (with good tuning, good preconditioner, fast $z \rightarrow \nabla^2 f(x) \hat{z}$ method, and some luck) can handle very large problems
Truncated Newton method

• backtracking line search on $\|\nabla f(x)\|

• typical CG termination rule: stop after $N_{\text{max}}$ steps or

$$\eta = \frac{\|\nabla^2 f(x) \Delta x + \nabla f(x)\|}{\|\nabla f(x)\|} \leq \epsilon_{\text{pcg}}$$

• with simple rules, $N_{\text{max}}$, $\epsilon_{\text{pcg}}$ are constant

• more sophisticated rules adapt $N_{\text{max}}$ or $\epsilon_{\text{pcg}}$ as algorithm proceeds (based on, e.g., value of $\|\nabla f(x)\|$, or progress in reducing $\|\nabla f(x)\|$)

$$\eta = \min(0.1, \|\nabla f(x)\|^{1/2})$$ guarantees (with large $N_{\text{max}}$) superlinear convergence
CG initialization

• we use CG to approximately solve $\nabla^2 f(x) \Delta x + \nabla f(x) = 0$

• if we initialize CG with $\Delta x = 0$
  – after one CG step, $\Delta x$ points in direction of negative gradient
    (so, $N_{\text{max}} = 1$ results in gradient method)
  – all CG iterates are descent directions for $f$

• another choice: initialize with $\Delta x = \Delta x_{\text{prev}}$, the previous search step
  – initial CG iterates need not be descent directions
  – but can give advantage when $N_{\text{max}}$ is small
• simple scheme: if $\Delta x_{\text{prev}}$ is a descent direction ($\Delta x_{\text{prev}}^T \nabla f(x) < 0$) start CG from

$$\Delta x = \frac{-\Delta x_{\text{prev}}^T \nabla f(x)}{\Delta x_{\text{prev}}^T \nabla^2 f(x) \Delta x_{\text{prev}}} \Delta x_{\text{prev}}$$

otherwise start CG from $\Delta x = 0$
Example

\(\ell_2\)-regularized logistic regression

\[
\text{minimize} \quad f(w) = \left(\frac{1}{m}\right) \sum_{i=1}^{m} \log \left(1 + \exp(-b_i x_i^T w)\right) + \sum_{i=1}^{n} \lambda_i w_i^2
\]

- variable is \(w \in \mathbb{R}^n\)

- problem data are \(x_i \in \mathbb{R}^n, b_i \in \{-1, 1\}, i = 1, \ldots, m\), and regularization parameter \(\lambda \in \mathbb{R}_+^n\)

- \(n\) is number of features; \(m\) is number of samples/observations
Hessian and gradient

\[ \nabla^2 f(w) = A^T DA + 2\Lambda, \quad \nabla f(w) = A^T g + 2\Lambda w \]

where

\[ A = [b_1 x_1 \cdots b_m x_m]^T, \quad D = \text{diag}(h), \quad \Lambda = \text{diag}(\lambda) \]

\[ g_i = -(1/m)/(1 + \exp(Aw)_i) \]
\[ h_i = (1/m) \exp(Aw)_i/(1 + \exp(Aw)_i)^2 \]

we never form \( \nabla^2 f(w) \); we carry out multiplication \( z \rightarrow \nabla^2 f(w)z \) as

\[ \nabla^2 f(w)z = (A^T DA + 2\Lambda)z = A^T (D(Az)) + 2\Lambda z \]
Problem instance

• $n = 10000$ features, $m = 20000$ samples (10000 each with $b_i = \pm 1$)

• $x_i$ have random sparsity pattern, with around 10 nonzero entries

• nonzero entries in $x_i$ drawn from $\mathcal{N}(b_i, 1)$

• $\lambda_i = 10^{-8}$

• around 500000 nonzeros in $\nabla^2 f$, and 30M nonzeros in Cholesky factor
Methods

• Newton (using Cholesky factorization of $\nabla^2 f (w)$)

• truncated Newton with $\epsilon_{cg} = 10^{-4}$, $N_{max} = 10$

• truncated Newton with $\epsilon_{cg} = 10^{-4}$, $N_{max} = 50$

• truncated Newton with $\epsilon_{cg} = 10^{-4}$, $N_{max} = 250$
Convergence versus iterations

\[ \| \nabla f \| \]

- cg 10
- cg 50
- cg 250
- Newton

Prof. S. Boyd, EE364b, Stanford University
Convergence versus cumulative CG steps

![Graph showing convergence versus cumulative CG steps](image)
• convergence of exact Newton, and truncated Newton methods with $N_{\text{max}} = 50$ and 250 essentially the same, in terms of iterations

• in terms elapsed time (and memory!), truncated Newton methods far better than Newton

• truncated Newton with $N_{\text{max}} = 10$ seems to jam near $\|\nabla f(w)\| \approx 10^{-6}$

• times (on AMD270 2GHz, 12GB, Linux) in sec:

<table>
<thead>
<tr>
<th>method</th>
<th>$|\nabla f(w)| \leq 10^{-5}$</th>
<th>$|\nabla f(w)| \leq 10^{-8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton</td>
<td>1600</td>
<td>2600</td>
</tr>
<tr>
<td>cg 10</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>cg 50</td>
<td>17</td>
<td>26</td>
</tr>
<tr>
<td>cg 250</td>
<td>35</td>
<td>54</td>
</tr>
</tbody>
</table>
Truncated PCG Newton method

approximate search direction found via diagonally preconditioned PCG
• diagonal preconditioning allows $N_{\text{max}} = 10$ to achieve high accuracy; speeds up other truncated Newton methods

• times:

<table>
<thead>
<tr>
<th>method</th>
<th>$| \nabla f(w) | \leq 10^{-5}$</th>
<th>$| \nabla f(w) | \leq 10^{-8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton</td>
<td>1600</td>
<td>2600</td>
</tr>
<tr>
<td>cg 10</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>cg 50</td>
<td>17</td>
<td>26</td>
</tr>
<tr>
<td>cg 250</td>
<td>35</td>
<td>54</td>
</tr>
<tr>
<td>pcg 10</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>pcg 50</td>
<td>13</td>
<td>24</td>
</tr>
<tr>
<td>pcg 250</td>
<td>23</td>
<td>34</td>
</tr>
</tbody>
</table>

• speedups of 1600:3, 2600:5 are not bad (and we really didn’t do much tuning . . . )
Extensions

• can extend to (infeasible start) Newton’s method with equality constraints

• since we don’t use exact Newton step, equality constraints not guaranteed to hold after finite number of steps (but $\|r_p\| \to 0$)

• can use for barrier, primal-dual methods
Truncated Newton interior-point methods

- use truncated Newton method to compute search direction in interior-point method

- tuning PCG parameters for optimal performance on a given problem class is tricky, since linear systems in interior-point methods often become ill-conditioned as algorithm proceeds

- but can work well (with luck, good preconditioner)
Network rate control

rate control problem

\[
\begin{align*}
\text{minimize} & \quad -U(f) = - \sum_{j=1}^{n} \log f_j \\
\text{subject to} & \quad Rf \preceq c
\end{align*}
\]

with variable \( f \)

- \( f \in \mathbb{R}_{++}^n \) is vector of flow rates
- \( U(f) = \sum_{j=1}^{n} \log f_j \) is flow utility
- \( R \in \mathbb{R}^{m \times n} \) is route matrix (\( R_{ij} \in \{0, 1\} \))
- \( c \in \mathbb{R}^m \) is vector of link capacities
Dual rate control problem

dual problem

maximize \( g(\lambda) = n - c^T \lambda + \sum_{i=1}^{m} \log(r_i^T \lambda) \)

subject to \( \lambda \geq 0 \)

with variable \( \lambda \in \mathbb{R}^m \)

duality gap

\[
\eta = -U(f) - g(\lambda) \\
= - \sum_{j=1}^{n} \log f_j - n + c^T \lambda - \sum_{i=1}^{m} \log(r_i^T \lambda)
\]
Primal-dual search direction (BV §11.7)

primal-dual search direction $\Delta f$, $\Delta \lambda$ given by

$$(D_1 + R^T D_2 R) \Delta f = g_1 - (1/t) R^T g_2, \quad \Delta \lambda = D_2 R \Delta f - \lambda + (1/t) g_2$$

where $s = c - R f$,

$$D_1 = \text{diag}(1/f_1^2, \ldots, 1/f_n^2), \quad D_2 = \text{diag}(\lambda_1/s_1, \ldots, \lambda_m/s_m)$$

$$g_1 = (1/f_1, \ldots, 1/f_n), \quad g_2 = (1/s_1, \ldots, 1/s_m)$$
Truncated Newton primal-dual algorithm

primal-dual residual:

\[ r = (r_{\text{dual}}, r_{\text{cent}}) = (-g_2 + R^T \lambda, \ \text{diag}(\lambda)s - (1/t)1) \]

given \( f \) with \( Rf < c; \lambda > 0 \)

while \( \eta/g(\lambda) > \epsilon \)

\[ t := \mu m/\eta \]

compute \( \Delta f \) using PCG as approximate solution of

\[ (D_1 + R^T D_2 R)\Delta f = g_1 - (1/t)R^T g_2 \]

\[ \Delta \lambda := D_2 R\Delta f - \lambda + (1/t)g_2 \]

carry out line search on \( \|r\|_2 \), and update:

\[ f := f + \gamma \Delta f, \ \lambda := \lambda + \gamma \Delta \lambda \]
• problem instance
  – $m = 200000$ links, $n = 100000$ flows
  – average of 12 links per flow, 6 flows per link
  – capacities random, uniform on $[0.1, 1]$

• algorithm parameters
  – truncated Newton with $\epsilon_{cg} = \min(0.1, \eta/g(\lambda))$, $N_{\max} = 200$
    ($N_{\max}$ never reached)
  – diagonal preconditioner
  – warm start
  – $\mu = 2$
  – $\epsilon = 0.001$ (i.e., solve to guaranteed $0.1\%$ suboptimality)
Primal and dual objective evolution

\[ -U(f) \]
\[ g(\lambda) \]

cumulative PCG iterations

Prof. S. Boyd, EE364b, Stanford University
Relative duality gap evolution

cumulative PCG iterations

relative duality gap

Prof. S. Boyd, EE364b, Stanford University
Primal and dual objective evolution \((n = 10^6)\)
Relative duality gap evolution \( (n = 10^6) \)