Lecture 21

Better Iterative Methods; Google and Markov Chains

L. Olson

Department of Computer Science
University of Illinois at Urbana-Champaign

November 10, 2009

Quadratic Form

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c$$

is a *quadratic* form. Example:

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \quad b = \begin{bmatrix} 2 \\ -8 \end{bmatrix} \quad c = 0$$

```
syms x y
X = [x;y];
A = [3 2; 2 6]; b= [2; -8];
f = (1/2) * transpose(X) * A * X - transpose(b) * X;

ezsurfc(f)
hold on;
ezplot('(1/2) * (2 - 3*x)');
ezplot('(1/6) * (-8 - 2*x)');
```

Quadratic Form

Gradient: $\nabla f(x)$ points uphill

```
df = [diff(f,x); diff(f,y)]
[xx,yy] = meshgrid(linspace(-6,6,20));
U=subs(df(1),{x,y},{xx,yy});
V=subs(df(2),{x,y},{xx,yy});
quiver(xx,yy,U,V);
```

Quadratic Form

Lets look at the i^{th} component of ∇f:

$$\begin{aligned} f(x+he_i) &= \frac{1}{2}(x+he_i)^T A(x+he_i) - b^T(x+he_i) + c \\ &\approx \frac{1}{2}(x^T Ax + he_i^T Ax + x^T Ahe_i) - b^T(x+he_i) + c \end{aligned}$$

so

$$\begin{aligned} \frac{f(x+he_i) - f(x)}{h} &= \frac{\frac{1}{2}(he_i^T Ax + x^T Ahe_i) - he_i^T b}{h} \\ &= \text{i}^{th} \text{ component of } \frac{1}{2}(Ax + A^T x) - b \end{aligned}$$

So

$$\nabla f = \frac{1}{2}A^T x + \frac{1}{2}Ax - b$$

So if A is symmetric, then $Ax = b$ at the minimum of f.

So

$$\nabla f = \frac{1}{2}A^T x + \frac{1}{2}Ax - b$$

So if A is symmetric, then $Ax = b$ at the minimum of f. Now, if

A is positive definite, f is concave up.
A is negative definite, f is concave down.
A is positive semi-definite, f is concave up (with a line as the minimum).
A is indefinite, f has a saddle. (Think $x_1^2 - x_2^2$)

Conjugate Gradients

- Suppose that A is $n \times n$ symmetric and positive definite.
- Since A is positive definite, $x^T Ax > 0$ for all $x \in \mathbb{R}^n$.
- Define a quadratic function

$$f(x) = \frac{1}{2}x^T Ax - x^T b$$

- It turns out that $-\nabla f = b - Ax = r$
- Optimization methods look in a "search direction" and pick the best step:

$$x_{k+1} = x_k + \alpha s_k$$

  Choose α so that $f(x_k + \alpha s_k)$ is minimized in the direction of s_k.
- Find α so that f is minimized:

$$0 = \frac{d}{d\alpha} f(x_{k+1}) = \nabla f(x_{k+1})^T \frac{d}{d\alpha} x_{k+1} = -r_{k+1}^T \frac{d}{d\alpha}(x_k + \alpha s_k) = -r_{k+1}^T s_k.$$

Steepest Descent

- Find α so that f is minimized in the direction of $\nabla f = r$:

$$0 = \frac{d}{d\alpha} f(x_{k+1}) = \nabla f(x_{k+1})^T \frac{d}{d\alpha} x_{k+1} = -r_{k+1}^T \frac{d}{d\alpha}(x_k + \alpha r_k) = -r_{k+1}^T r_k.$$

- Pick α so that $\nabla f(x_{k+1})$ and r_k are orthogonal.
- Since $\nabla f(x_{k+1}) = -r_{k+1}$ we want

$$\begin{aligned} r_{k+1}^T r_k &= 0 \\ (b - A(x_k + \alpha r_k))^T r_k &= 0 \\ (b - Ax_k)^T r_k - \alpha (Ar_k)^T r_k &= 0 \\ r_k^T r_k &= \alpha r_k^T A r_k \end{aligned}$$

- So, the optimal search parameter is

$$\alpha = -\frac{r_k^T r_k}{r_k^T A r_k}$$

So far, we've *motivated* a method. We haven't been careful about the step direction s_k yet.

- Notice: convergence is staggered

So far, we've *motivated* a method. We haven't been careful about the step direction s_k yet.

- Notice: convergence is staggered
- Better: step toward a principle axis

So far, we've *motivated* a method. We haven't been careful about the step direction s_k yet.

- Notice: convergence is staggered
- Better: step toward a principle axis
- Notice: axis defined by eigenvectors of A

So far, we've *motivated* a method. We haven't been careful about the step direction s_k yet.

- Notice: convergence is staggered
- Better: step toward a principle axis
- Notice: axis defined by eigenvectors of A
- Compromise: find a step direction that is A-orthogonal to the previous

So far, we've *motivated* a method. We haven't been careful about the step direction s_k yet.

- Notice: convergence is staggered
- Better: step toward a principle axis
- Notice: axis defined by eigenvectors of A
- Compromise: find a step direction that is A-orthogonal to the previous
- Notice: take a search closest to r that is conjugate.

So far, we've *motivated* a method. We haven't been careful about the step direction s_k yet.

- Notice: convergence is staggered
- Better: step toward a principle axis
- Notice: axis defined by eigenvectors of A
- Compromise: find a step direction that is A-orthogonal to the previous
- Notice: take a search closest to r that is conjugate.
- Result: converges in at most n steps

Conjugate Gradients

1 $x_0 =$ `initial guess`
2 $r_0 = b - Ax_0$
3 $s_0 = r_0$
4 `for` $k = 0, 1, 2, \ldots$
5 $\quad \alpha_k = \frac{r_k^T r_k}{s_k^T A s_k}$
6 $\quad x_{k+1} = x_k + \alpha_k s_k$
7 $\quad r_{k+1} = r_k - \alpha_k A s_k$
8 $\quad \beta_{k+1} = r_{k+1}^T r_{k+1} / r_k^T r_k$
9 $\quad s_{k+1} = r_{k+1} + \beta_{k+1} s_k$

Convergence

- A-norm of the error is minimized in each step
- Bound:

$$\|e_k\|_A \leqslant 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e_0\|_A$$

- κ is the condition number
- for symmetric matrices, $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$

Preconditioning

We've seen iterations that replace A with a matrix Q such that $Q^{-1}A$ resembles the identity matrix. Congugate Gradient takes a different approach and solves $Ax = b$ more directly.
Can we combine these ideas?
Preconditioning is transforming the original problem $Ax = b$ with another matrix Q and then solving that modified problem. For example

- Left Preconditioning: $QAx = Qb$
- Right Preconditioning: $AQw = b$, $Qw = x$
- Symmetric preconditioning: $Q^T AQw = Q^T b$, $Qw = x$

For Congugate Gradient, the matrix must be symmetric, so the last form is the most common.
There are methods that generalize conjugate gradient for nonsymmetric matrices. See CS 457 for more details

Coming Up

- Google, Markov Chains, intro to Monte Carlo Simulations
- Eigenvalues: SVD, Power Method
- Least-Squares
- Monte Carlo Simulations

Google (basics)

many slides courtesy of T. Chartier at Davidson

- A component of Google's success can be attributed to the PageRank^{TM} algorithm developed by Google's founders
- algorithm determined entirely by link structure of the WWW
- recomputed once a month
- involves no content of any Web page
- How are the search results ordered?

Randomly Walking with Google

- start at any webpage
- randomly select a link and follow
- repeat
- what are the outcomes?

The outcomes of such a random walk are:

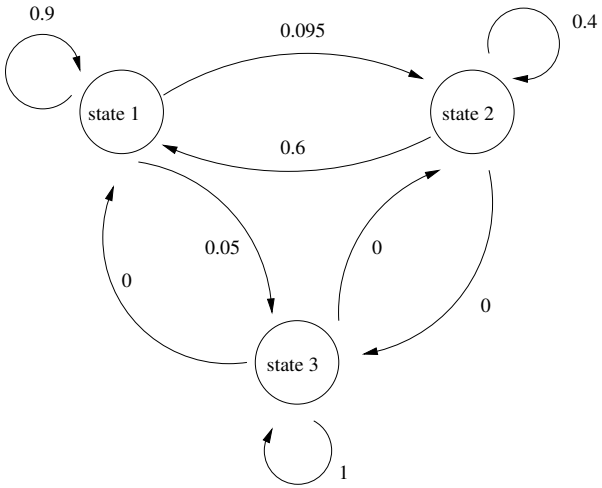
- a dead end on a page with no outgoing links
- a cycle where you end up where you began: known as a *Markov chain* or *Markov process*.
- The limiting probability that an infinitely dedicated random surfer visits any particular page is its PageRank.
- A page has high rank if other pages with high rank link to it.

Markov Chains

- Markov chains can model the behavior of a system that depends only on the previous experiment or state.
- That is, the next state of the system depends only on the current state where the outcome of each experiment is one of a discrete set of states.
- Markov chains require a transition matrix, P, where $P(j, i)$ equals the probability of going from state i to state j.

Markov chain model of a Broken Machine

A certain office machine has three states: Working, Temporarily Broken, and Permanently Broken. Each day it is in one of these states. The *transition diagram* between the states is shown below.



Therefore, the transition matrix is:

$$P = \begin{pmatrix} .9 & .6 & 0 \\ .095 & .4 & 0 \\ .005 & 0 & 1 \end{pmatrix}$$

Answers via Markov chains

In our previous model suppose we ask the question:

If the machine is working on Day 1, what is the probability that it is still functional (that is, not permanently broken) a year from now?

Stepping through Time with Markov

We can answer this question by computing the 365'th power of the transition matrix:

$$P = \begin{pmatrix} .9 & .6 & 0 \\ .095 & .4 & 0 \\ .005 & 0 & 1 \end{pmatrix}^{365} = \begin{pmatrix} 0.1779 & 0.1792 & 0 \\ 0.0284 & 0.0286 & 0 \\ 0.7937 & 0.7922 & 1 \end{pmatrix}$$

The first column of this matrix gives the probability of being in each state on Day 366 (with our assumption that it is working on Day 1, so $v^{(0)} = [1\ 0\ 0]'$). We see that the probability that the machine is permanently broken in 0.7937 and so there is about a 21% chance that the machine is still functional.

Steady State Solution

For this Markov chain the steady state solution is the vector $\mathbf{v} = [0\ 0\ 1]'$, as we can see by computing the eigenvalues and eigenvectors. That is, $P\mathbf{v} = \mathbf{v}$.

State 3 (Permanently Broken) is called an *absorbing state*. No matter what state we start in, we will eventually end up in State 3 with probability 1, and once in State 3 we can never leave.

Back to Google

- Let W be the set of Web pages that can reached by following a chain of hyperlinks starting from a page at Google.
- Let n be the number of pages in W.
- The set W actually varies with time, by the end of 2005, n was over 10 billion.
- Let G be the $n \times n$ connectivity matrix of W, that is, $G_{i,j}$ is 1 if there is a hyperlink from page i to page j and 0 otherwise.
- The matrix G is huge, but very sparse; its number of nonzeros is the total number of hyperlinks in the pages in W.

Google and Probability

- Let c_j and r_i be the column and row sums of G, respectively. That is,

$$c_j = \sum_i G_{i,j}, \qquad r_i = \sum_j G_{i,j}$$

- Then c_k and r_k are the indegree and outdegree of the k-th page. In other words, c_k is the number of links into page k and r_k is the number of links from page k.
- Let p be the fraction of time that the random walk follows a link.
- Google typically takes this to be $p = 0.85$.
- Then $1 - p$ is the fraction of time that an arbitrary page is chosen.

Google meets Markov

- Let A be an $n \times n$ matrix whose elements are $A_{i,j} = pG_{i,j}/c_j + \delta$ where $\delta = (1-p)/n$.
- This matrix is the transition matrix of the Markov chain of a random walk!
- Notice that A comes from scaling the connectivity matrix by its column sums.
- The j-th column is the probability of jumping from the j-th page to the other pages on the Web.

The problem

Can write A, the transition matrix, as

$$A = pGD + ez^T$$

where e is the vector of all ones and where ez^T account for dead linked pages and

$$D_{jj} = 1/c_j \text{ (or 0)} \quad z_j = \delta \text{ (or } 1/n)$$

Then $x = Ax$ can be written

$$(I - pGD)x = (z^T x)e = e$$

see `pagerank.m`

Eigenvectors and Google

Find $x = Ax$ and the elements of x are Google's PageRank. Remember $n > 10^{10}$ (as of 2005) and growing.

For any particular query, Google finds pages on the Web that match the query. The pages are then listed in the order of their PageRank.

Matlab

Moler's scripts

To search from a homepage, you will type a statement like:

```
[U,G] = surfer('http://www.uiuc.edu',n).
```

This starts at the given URL and tries to surf the Web until it has visited n pages. That is, an n by n matrix is formed.

Note, `surfer.m` is very primitive...

Matlab

Moler's scripts

- Download `surfer.m`, `pagerank.m` and `pagerankpow.m`
- `[U,G] = surfer('http://www.slashdot.org',20);`

  U = a cell array of n strings, the URLs of the nodes.

  G = an n-by-n sparse matrix with $G(i,j) = 1$ if node j is linked to node i.
- Next: `spy(G)`. This shows the nonzero structure of the connectivity matrix.
- Finally: `pagerank(U,G)` and the pagerank will be computed.

Numerics?

- what does this have to do with numerical methods?
- large sparse matrices (although Google avoids this)
- solution to a linear system (eigenvalue problem)
- Markov chains...

The markov process is important. Let's step back and look at some statistics

- randomness
- monte carlo
- markov chains
- metropolis algorithm

Brownian Motion is an example of a Markov process:
`http://galileo.phys.virginia.edu/classes/109N/more_stuff/Applets/bro`
(more later)

Why

- Central to the PageRank (and many many other applications in fincance, science, informatics, etc) is that we randomly process something
- what we want to know is “on average” what is likely to happen
- what would happen if we have an infinite number of samples?
- next: eigenvalues, SVD, pagerank, Monte Carlo