16 Lecture 16, Feb 25

Announcements

- HW5 (Newton's method, handwritten digit recognition) due next Tue Mar 1 @ 11:59PM.
- Quiz 3 next Tue 3/1. In class, closed book.

Last time

- EM algorithm.
- EM example: finite Gaussian mixture.
- MM algorithm.
- MM example: PET imaging.

Today

- PET imaging: GPU computing.
- MM example: matrix completion.
- MM example: nonnegative matrix factorization (NNMF).
- Quasi-Newton method.

GPU

GPUs are ubiquitous: servers, desktops, and laptops.

NVIDIA GPUs	Tesla M2090	GTX 580	GT 650M
			
Computers	servers, cluster	desktop	laptop
			
Main usage	scientific computing	gaming	gaming
Current version	K40	GTX 980	GTX 900M
Memory	6GB	1.5GB	1GB
Memory bandwidth	177GB/sec	192GB/sec	80GB/sec
Number of cores	512	512	384
Processor clock	1.3GHz	1.5GHz	0.9GHz
Peak DP performance	666Gflops		
Peak SP performance	1332Gflops	1581Gflops	691Gflops
Release price	$2500	$500	OEM

MM example: Netflix and matrix completion

- Snapshot of the kind of data collected by Netflix. Only 100,480,507 ratings (1.2% entries of the 480K-by-18K matrix) are observed

ID	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6	...	movie 17,770
user 1	5	3	4	3	3	*NA*	...	1
user 2	4	*NA*	*NA*	*NA*	*NA*	*NA*	...	*NA*
user 3	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	...	*NA*
user 4	4	*NA*	*NA*	*NA*	*NA*	2	...	4
user 5	*NA*	*NA*	*NA*	5	*NA*	*NA*	...	*NA*
user 6	3	*NA*	*NA*	5	1	*NA*	...	3
user 7	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	...	*NA*
user 8	5	*NA*	5	*NA*	*NA*	*NA*	...	*NA*
user 9	*NA*	*NA*	*NA*	*NA*	3	*NA*	...	*NA*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
user 480,189	*NA*	5	*NA*	*NA*	*NA*	*NA*	...	*NA*

- Netflix challenge: impute the unobserved ratings for personalized recommendation. `http://en.wikipedia.org/wiki/Netflix_Prize`

- *Matrix completion problem.* Observe a very sparse matrix $\boldsymbol{Y} = (y_{ij})$. Want to impute all the missing entries. It is possible only when the matrix is structured, e.g., of *low rank*.

- Let $\Omega = \{(i,j) : \text{observed entries}\}$ index the observed entries and $P_\Omega(\boldsymbol{M})$ denote the projection of matrix $\boldsymbol{M}$ to Ω. The problem

$$\min_{\text{rank}(\boldsymbol{X}) \le r} \frac{1}{2} \|P_\Omega(\boldsymbol{Y}) - P_\Omega(\boldsymbol{X})\|_{\text{F}}^2 = \frac{1}{2} \sum_{(i,j) \in \Omega} (y_{ij} - x_{ij})^2$$

  unfortunately is non-convex and difficult.

- *Convex relaxation* (Mazumder et al., 2010)

$$\min_{\boldsymbol{X}} f(\boldsymbol{X}) = \frac{1}{2} \|P_\Omega(\boldsymbol{Y}) - P_\Omega(\boldsymbol{X})\|_{\text{F}}^2 + \lambda \|\boldsymbol{X}\|_*,$$

  where $\|\boldsymbol{X}\|_* = \|\sigma(\boldsymbol{X})\|_1 = \sum_i \sigma_i(\boldsymbol{X})$ is the nuclear norm.

- Majorization step:

$$\begin{aligned} f(\boldsymbol{X}) &= \frac{1}{2}\sum_{(i,j)\in\Omega}(y_{ij}-x_{ij})^2+\frac{1}{2}\sum_{(i,j)\notin\Omega}0+\lambda\|\boldsymbol{X}\|_* \\ &\leq \frac{1}{2}\sum_{(i,j)\in\Omega}(y_{ij}-x_{ij})^2+\frac{1}{2}\sum_{(i,j)\notin\Omega}(x_{ij}^{(t)}-x_{ij})^2+\lambda\|\boldsymbol{X}\|_* \\ &= \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{Z}^{(t)}\|_{\mathrm{F}}^2+\lambda\|\boldsymbol{X}\|_* \\ &= g(\boldsymbol{X}|\boldsymbol{X}^{(t)}), \end{aligned}$$

  where $\boldsymbol{Z}^{(t)} = P_\Omega(\boldsymbol{Y})+P_{\Omega^\perp}(\boldsymbol{X}^{(t)})$. "fill in missing entries by current imputation"

- Minimization step:

  Rewrite the surrogate function

$$g(\boldsymbol{X}|\boldsymbol{X}^{(t)}) = \frac{1}{2}\|\boldsymbol{X}\|_{\mathrm{F}}^2-\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{Z}^{(t)})+\frac{1}{2}\|\boldsymbol{Z}^{(t)}\|_{\mathrm{F}}^2+\lambda\|\boldsymbol{X}\|_*.$$

  Let σ_i be the singular values of $\boldsymbol{X}$ and ω_i be the singular values of $\boldsymbol{Z}^{(t)}$. Observe

$$\begin{aligned} \|\boldsymbol{X}\|_{\mathrm{F}}^2 &= \|\sigma(\boldsymbol{X})\|_2^2=\sum_i\sigma_i^2 \\ \|\boldsymbol{Z}^{(t)}\|_{\mathrm{F}}^2 &= \|\sigma(\boldsymbol{Z}^{(t)})\|_2^2=\sum_i\omega_i^2 \end{aligned}$$

  and by the Fan-von Neuman's inequality

$$\mathrm{tr}(\boldsymbol{X}^\intercal\boldsymbol{Z}^{(t)})\leq\sum_i\sigma_i\omega_i$$

  with equality achieved if and only if the left and right singular vectors of the two matrices coincide. Thus we can choose $\boldsymbol{X}$ to have same singular vectors as $\boldsymbol{Z}^{(t)}$ and

$$\begin{aligned} g(\boldsymbol{X}|\boldsymbol{X}^{(t)}) &= \frac{1}{2}\sum_i\sigma_i^2-\sum_i\sigma_i\omega_i+\frac{1}{2}\omega_i^2+\lambda\sum_i\sigma_i \\ &= \frac{1}{2}\sum_i(\sigma_i-\omega_i)^2+\lambda\sum_i\sigma_i, \end{aligned}$$

  with minimizer given by $\sigma_i^{(t+1)} = (\omega_i-\lambda)_+$. "Singular value thresholding"

- Algorithm for matrix completion:

Initialize $\boldsymbol{X}^{(0)} \in \mathbb{R}^{m \times n}$

repeat

$\boldsymbol{Z}^{(t+1)} \leftarrow P_{\Omega}(\boldsymbol{Y}) + P_{\Omega^{\perp}}(\boldsymbol{X}^{(t)})$

Compute SVD: $\boldsymbol{U}\mathrm{diag}(\boldsymbol{w})\boldsymbol{V}^{\intercal} \leftarrow \boldsymbol{Z}^{(t+1)}$

$\boldsymbol{X}^{(t+1)} \leftarrow \boldsymbol{U}\mathrm{diag}[(\boldsymbol{w} - \lambda)_{+}]\boldsymbol{V}^{\intercal}$

until objective value converges

- "Golub-Kahan-Reinsch" algorithm takes $4m^2n + 8mn^2 + 9n^3$ flops for a $m \geq n$ matrix and is not going to work for 480K-by-18K Netflix matrix.

  Notice only top singular values are needed since low rank solutions are achieved by large λ. Lanczos/Arnoldi algorithm is the way to go. Matrix-vector multiplication $\boldsymbol{Z}^{(t)}\boldsymbol{v}$ is fast (why?)

MM example: non-negative matrix factorization (NNMF)

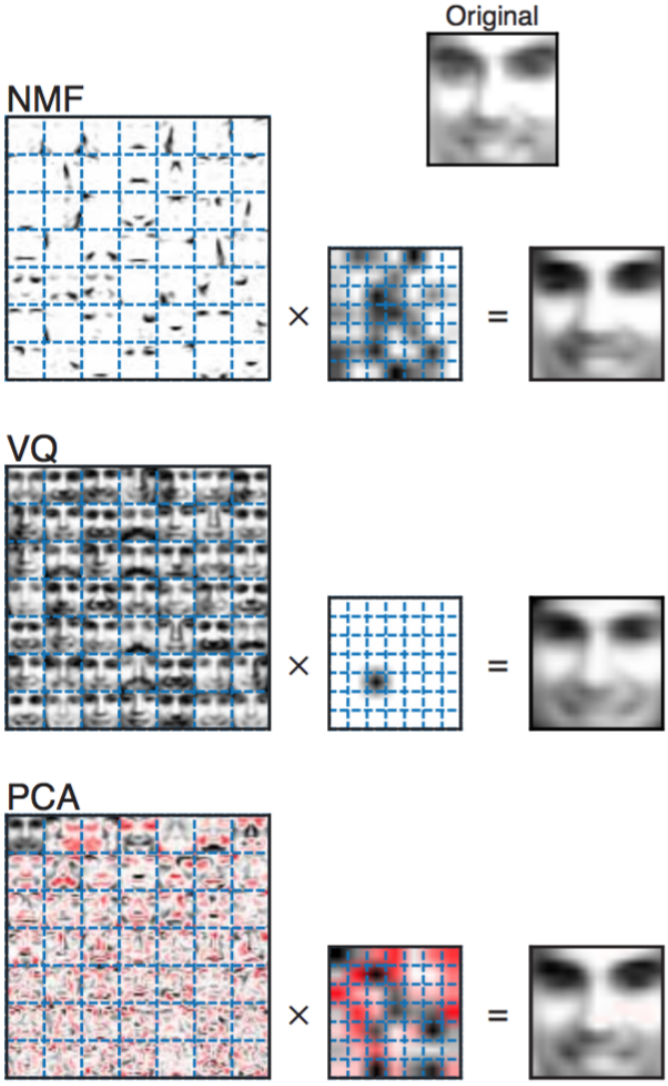


FIGURE 14.33. *Non-negative matrix factorization (NMF), vector quantization (VQ, equivalent to k-means clustering) and principal components analysis (PCA) applied to a database of facial images. Details are given in the text. Unlike VQ and PCA, NMF learns to represent faces with a set of basis images resembling parts of faces.*

- Nonnegative matrix factorization (NNMF) was introduced by Lee and Seung (1999, 2001) as an analog of principal components and vector quantization with applications in data compression and clustering.
- In mathematical terms, one approximates a data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ with non-

negative entries x_{ij} by a product of two low-rank matrices $\boldsymbol{V} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{W} \in \mathbb{R}^{r \times n}$ with nonnegative entries v_{ik} and w_{kj}. Consider minimization of the squared Frobenius norm

$$L(\boldsymbol{V}, \boldsymbol{W}) = \|\boldsymbol{X} - \boldsymbol{V}\boldsymbol{W}\|_{\mathrm{F}}^2 = \sum_i \sum_j (x_{ij} - \sum_k v_{ik} w_{kj})^2, \quad v_{ik} \geq 0, w_{kj} \geq 0,$$

which should lead to a good factorization.

- $L(\boldsymbol{V}, \boldsymbol{W})$ is not convex, but bi-convex. The strategy is to alternately update $\boldsymbol{V}$ and $\boldsymbol{W}$.

- The key is the majorization, via convexity of the function $(x_{ij} - x)^2$,

$$\left(x_{ij} - \sum_k v_{ik} w_{kj} \right)^2 \leq \sum_k \frac{a_{ikj}^{(t)}}{b_{ij}^{(t)}} \left(x_{ij} - \frac{b_{ij}^{(t)}}{a_{ikj}^{(t)}} v_{ik} w_{kj} \right)^2,$$

  where

$$a_{ikj}^{(t)} = v_{ik}^{(t)} w_{kj}^{(t)}, \quad b_{ij}^{(t)} = \sum_k v_{ik}^{(t)} w_{kj}^{(t)}.$$

- This suggests the alternating multiplicative updates

$$\begin{aligned} v_{ik}^{(t+1)} &= v_{ik}^{(t)} \frac{\sum_j x_{ij} w_{kj}^{(t)}}{\sum_j b_{ij}^{(t)} w_{kj}^{(t)}} \\ b_{ij}^{(t+.5)} &= \sum_k v_{ik}^{(t+1)} w_{kj}^{(t)} \\ w_{kj}^{(t+1)} &= w_{kj}^{(t)} \frac{\sum_i x_{ij} v_{ik}^{(t+1)}}{\sum_i b_{ij}^{(t+.5)} v_{ik}^{(t+1)}}. \end{aligned}$$

- In MATLAB or JULIA notation, the update is extremely simple

$$\begin{aligned} \boldsymbol{B}^{(t)} &= \boldsymbol{V}^{(t)} \boldsymbol{W}^{(t)} \\ \boldsymbol{V}^{(t+1)} &= \boldsymbol{V}^{(t)} \odot (\boldsymbol{X} \boldsymbol{W}^{(t)T}) \oslash (\boldsymbol{B}^{(t)} \boldsymbol{W}^{(t)T}) \\ \boldsymbol{B}^{(t+.5)} &= \boldsymbol{V}^{(t+1)} \boldsymbol{W}^{(t)} \\ \boldsymbol{W}^{(t+1)} &= \boldsymbol{W}^{(t)} \odot (\boldsymbol{X}^T \boldsymbol{V}^{(t+1)}) \oslash (\boldsymbol{B}^{(t+.5)T} \boldsymbol{V}^{(t)}), \end{aligned}$$

  where $\odot$ denotes elementwise multiplication and $\oslash$ denotes elementwise division. If we start with $v_{ik}, w_{kj} > 0$, parameter iterates stay positive.

- Database #1 from the MIT Center for Biological and Computational Learning (CBCL) (`http://cbcl.mit.edu`) reduces to a matrix X containing $m = 2,429$ gray-scale face images with $n = 19 \times 19 = 361$ pixels per face. Each image (row) is scaled to have mean and standard deviation 0.25.

- CPU: Intel Core 2 @ 3.2GHZ (4 cores), implemented using BLAS in the GNU Scientific Library (GSL)
  GPU: NVIDIA GeForce GTX 280 (240 cores), implemented using cuBLAS.

		CPU		GPU		
Rank r	Iters	Time	Function	Time	Function	Speedup
10	25459	1203	106.2653503	55	106.2653504	22
20	87801	7564	89.56601262	163	89.56601287	46
30	55783	7013	78.42143486	103	78.42143507	68
40	47775	7880	70.05415929	119	70.05415950	66
50	53523	11108	63.51429261	121	63.51429219	92
60	77321	19407	58.24854375	174	58.24854336	112

Quasi-Newton methods (KL 14.9)

- Quasi-Newton is probably the most successful "black-box" optimizers in use. E.g., implemented in the general purpose optimization routine `optim()` in R.

- Consider the general Newton scheme for minimizing $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^p$:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - s[\boldsymbol{A}^{(t)}]^{-1} \nabla f(\boldsymbol{x}^{(t)}),$$

  where $\boldsymbol{A}^{(t)}$ a pd approximation of the Hessian $d^2 f(\boldsymbol{x}^{(t)})$.

  - Pros: fast (quadratic) convergence
  - Cons: compute and store Hessian at each iteration (usually $O(np^2)$ cost in statistical problems), solving a linear system ($O(p^3)$ cost in general), human efforts (derive and implement gradient and Hessian, pd approximation, ...)

- Any pd $\boldsymbol{A}$ gives a descent algorithm – tradeoff between convergence rate and cost per iteration.

- Setting $\boldsymbol{A} = \boldsymbol{I}$ leads to the *steepest descent* algorithm. Picture: slow convergence (zig-zagging) of steepest descent (with exact line search) for minimizing a convex quadratic function (ellipse).

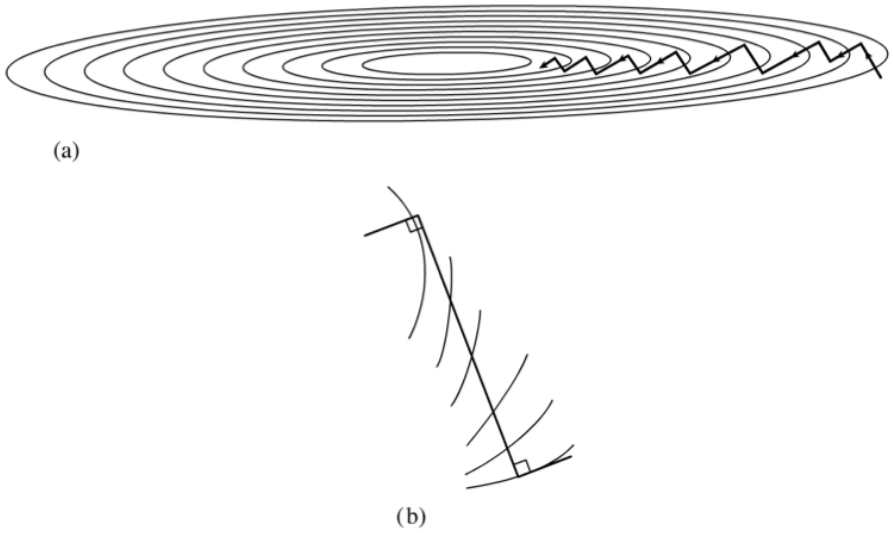


Figure 10.6.1. (a) Steepest descent method in a long, narrow "valley." While more efficient than the strategy of Figure 10.5.1, steepest descent is nonetheless an inefficient strategy, taking many steps to reach the valley floor. (b) Magnified view of one step: A step starts off in the local gradient direction, perpendicular to the contour lines, and traverses a straight line until a local minimum is reached, where the traverse is parallel to the local contour lines.

  How many steps does the Newton's method take for a convex quadratic f?

- Idea of Quasi-Newton: No analytical Hessian is required (still need gradient). Update approximate Hessian $\boldsymbol{A}$ according to the *secant condition*

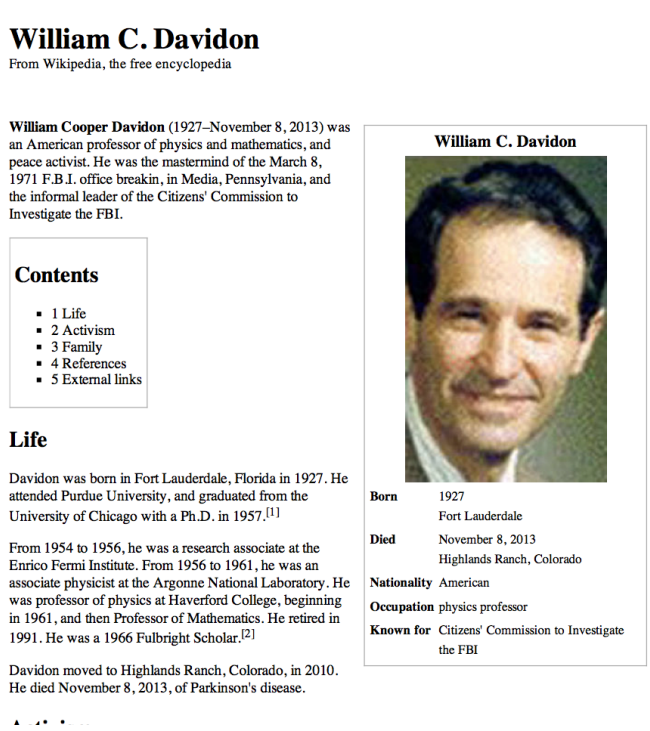
$$\nabla f(\boldsymbol{x}^{(t-1)}) - \nabla f(\boldsymbol{x}^{(t)}) \approx [d^2 f(\boldsymbol{x}^{(t)})](\boldsymbol{x}^{(t-1)} - \boldsymbol{x}^{(t)}).$$

  Instead of computing $\boldsymbol{A}$ from scratch at each iteration, we update an approximation $\boldsymbol{A}$ to $d^2 f(\boldsymbol{x})$ which satisfies (1) p.d., (2) the secant condition, and (3) closest to the previous approximation.

  – Super-linear convergence, compared to the quadratic convergence of Newton's method. But each iteration only takes $O(p^2)$!

- History: Davidon was a physicist at Argonne National Lab in 50s and proposed the very first idea of quasi-Newton method in 1959. Later studied, implemented, and popularized by Fletcher and Powell. Davidon's original paper was not

accepted for publication ☹; 30 years later it appeared as the first article in the first issue of *SIAM Journal of Optimization* (Davidon, 1991).



William C. Davidon
From Wikipedia, the free encyclopedia

William Cooper Davidon (1927–November 8, 2013) was an American professor of physics and mathematics, and peace activist. He was the mastermind of the March 8, 1971 F.B.I. office breakin, in Media, Pennsylvania, and the informal leader of the Citizens' Commission to Investigate the FBI.

Contents

- 1 Life
- 2 Activism
- 3 Family
- 4 References
- 5 External links

Life

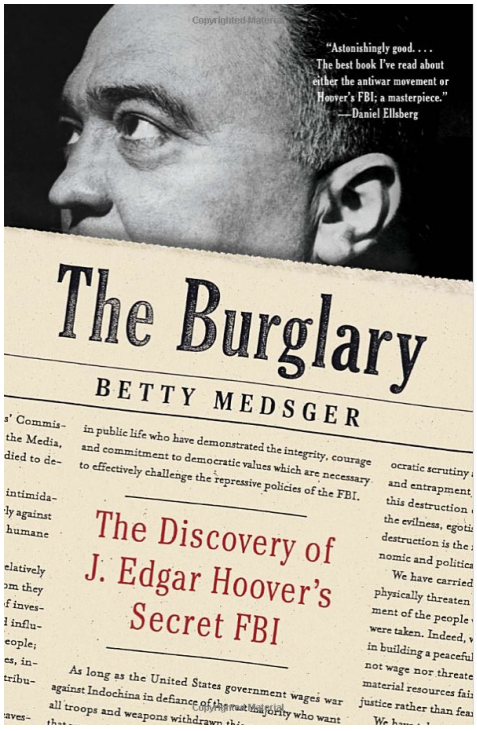
Davidon was born in Fort Lauderdale, Florida in 1927. He attended Purdue University, and graduated from the University of Chicago with a Ph.D. in 1957.[1]

From 1954 to 1956, he was a research associate at the Enrico Fermi Institute. From 1956 to 1961, he was an associate physicist at the Argonne National Laboratory. He was professor of physics at Haverford College, beginning in 1961, and then Professor of Mathematics. He retired in 1991. He was a 1966 Fulbright Scholar.[2]

Davidon moved to Highlands Ranch, Colorado, in 2010. He died November 8, 2013, of Parkinson's disease.

William C. Davidon

Born	1927 Fort Lauderdale
Died	November 8, 2013 Highlands Ranch, Colorado
Nationality	American
Occupation	physics professor
Known for	Citizens' Commission to Investigate the FBI



- Davidon-Fletcher-Powell (DFP) rank-2 update. Solve

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{A} - \boldsymbol{A}^{(t)}\|_{\mathrm{F}} \\ \text{subject to} \quad & \boldsymbol{A} = \boldsymbol{A}^{\mathsf{T}} \\ & \boldsymbol{A}(\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t-1)}) = \nabla f(\boldsymbol{x}^{(t)}) - \nabla f(\boldsymbol{x}^{(t-1)}) \end{aligned}$$

  for the next approximation $\boldsymbol{A}^{(t+1)}$. The solution is a low rank (rank 1 or rank 2) update of $\boldsymbol{A}^{(t)}$. The inverse is too thanks to Sherman-Morrison-Woodbury! $O(p^2)$ operations. Need to store a p-by-p dense matrix. Positive definiteness is guaranteed by the same trick you are using in HW5!

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) rank 2 update is considered by many the most effective among all quasi-Newton updates. BFGS imposes secant con-

dition on the inverse of Hessian $\boldsymbol{H} = \boldsymbol{A}^{-1}$.

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{H} - \boldsymbol{H}^{(t)}\|_{\text{F}} \\ \text{subject to} \quad & \boldsymbol{H} = \boldsymbol{H}^{\text{T}} \\ & \boldsymbol{H}[\nabla f(\boldsymbol{x}^{(t)}) - \nabla f(\boldsymbol{x}^{(t-1)})] = \boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t-1)}. \end{aligned}$$

Again $\boldsymbol{H}^{(t+1)}$ is a rank two update of $\boldsymbol{H}^{(t)}$. $O(p^2)$ operations. Still need to store a dense p-by-p matrix.

- Limited-memory BFGS (L-BFGS). Only store the secant pairs. Particularly useful for large scale optimization.
- L-BFGS-B: with box-constraints. Implemented in the general purpose optimization routine `optim()` in R.
- How to set the initial approximation $\boldsymbol{A}^{(0)}$? Identity or Hessian (if pd) or Fisher information matrix at starting point.

(Linear) Conjugate gradient method

- (Linear) Conjugate gradient is the top-notch iterative method for solving large, structured linear systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Earlier we talked about Jacobi and Gauss-Seidel as the classical iterative solvers.

Table 1. Kershaw's results for a fusion problem.

Method	Number of iterations
Gauss Seidel	208,000
Block successive overrelaxation methods	765
Incomplete Cholesky conjugate gradients	25

- *Linear* conjugate gradient method: for solving large linear systems of equations.

  History: Hestenes and Stiefel in 50s.

- Solve linear equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is pd, is equivalent to

$$\text{minimize } f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}.$$

  Note $\nabla f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} =: r(\boldsymbol{x})$.

- Consider a simple idea: coordinate descent, that is to update x_j alternatingly. Same as the Gauss-Seidel iteration.

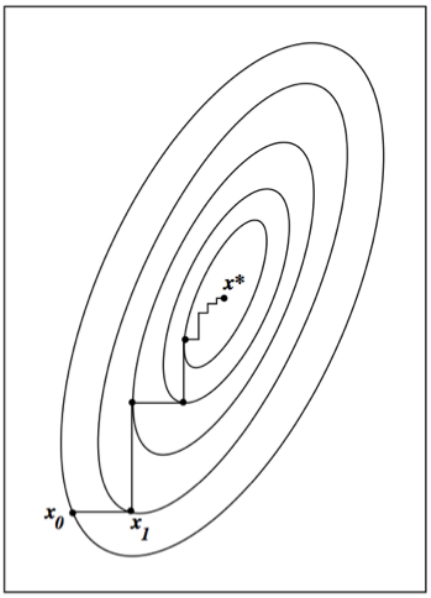


Figure 9.1
Coordinate search method makes slow progress on this function of two variables.

- A set of vectors $\{\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(l)}\}$ is said to be conjugate wrt $\boldsymbol{A}$ if

$$\boldsymbol{p}_i^\top \boldsymbol{A}\boldsymbol{p}_j = 0, \quad \text{for all } i \neq j.$$

- *Conjugate direction* method: Given a set of conjugate vectors $\{\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(l)}\}$, at iteration t, we search along the conjugate direction $\boldsymbol{p}^{(t)}$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)},$$

  where

$$\alpha^{(t)} = -\frac{\boldsymbol{r}^{(t)\top}\boldsymbol{p}^{(t)}}{\boldsymbol{p}^{(t)\top}\boldsymbol{A}\boldsymbol{p}^{(t)}}.$$

- Theorem: $\boldsymbol{x}^{(t)}$ converges to the solution in at most n steps.

  Intuition: Look at graph.

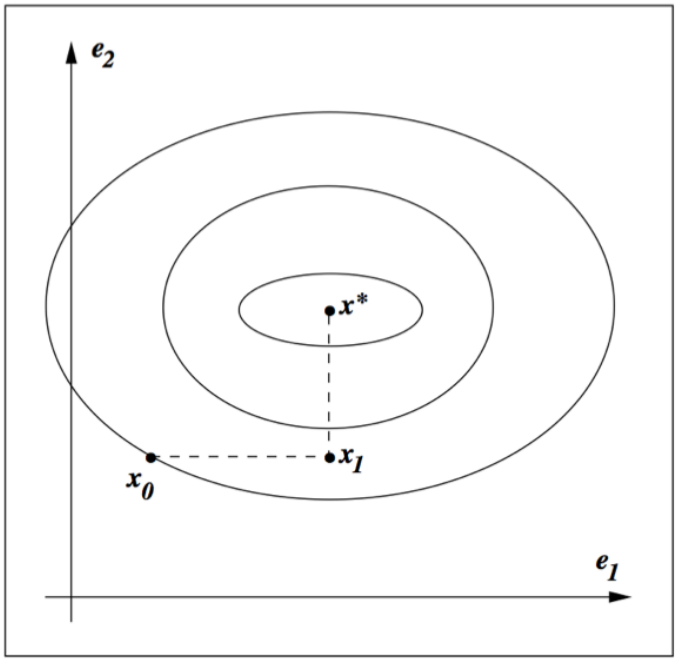


Figure 5.1 Successive minimizations along the coordinate directions find the minimizer of a quadratic with a diagonal Hessian in n iterations.

- *Conjugate gradient* method. Idea: generate $\boldsymbol{p}^{(t)}$ using only $\boldsymbol{p}^{(t-1)}$

$$\boldsymbol{p}^{(t)} = -\boldsymbol{r}^{(t)} + \beta^{(t)}\boldsymbol{p}^{(t-1)},$$

  where $\beta^{(t)}$ is determined by the conjugacy condition $\boldsymbol{p}^{(t-1)\top}\boldsymbol{A}\boldsymbol{p}^{(t)} = 0$

$$\beta^{(t)} = \frac{\boldsymbol{r}^{(t)\top}\boldsymbol{A}\boldsymbol{p}^{(t-1)}}{\boldsymbol{p}^{(t-1)\top}\boldsymbol{A}\boldsymbol{p}^{(t-1)}}.$$

- CG algorithm (preliminary version):

  Given $\boldsymbol{x}^{(0)}$
  Initialize: $\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$, $\boldsymbol{p}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$, $t = 0$
  while $\boldsymbol{r}^{(0)} \neq \boldsymbol{0}$ **do**
  $\alpha^{(t)} \leftarrow -\frac{\boldsymbol{r}^{(t)\top}\boldsymbol{p}^{(t)}}{\boldsymbol{p}^{(t)\top}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
  $\boldsymbol{r}^{(t+1)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(t+1)} - \boldsymbol{b}$
  $\beta^{(t+1)} \leftarrow \frac{\boldsymbol{r}^{(t+1)\top}\boldsymbol{A}\boldsymbol{p}^{(t)}}{\boldsymbol{p}^{(t)\top}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
  $\boldsymbol{p}^{(t+1)} \leftarrow -\boldsymbol{r}^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$

$t \leftarrow t + 1$
end while

- Theorem: With CG algorithm

  1. $\boldsymbol{r}^{(t)}$ are mutually orthogonal.
  2. $\{\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{r}^{(t)}\}$ is contained in the *Krylov* subspace of degree t for $\boldsymbol{r}^{(0)}$, denoted by

  $$\mathcal{K}(\boldsymbol{r}^{(0)}; t) = \text{span}\{\boldsymbol{r}^{(0)}, \boldsymbol{A}\boldsymbol{r}^{(0)}, \boldsymbol{A}^2\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{A}^t\boldsymbol{r}^{(0)}\}.$$

  3. $\{\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(t)}\}$ is contained in $\mathcal{K}(\boldsymbol{r}^{(0)}; t)$.
  4. $\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(t)}$ are conjugate wrt $\boldsymbol{A}$.

  The iterates $\boldsymbol{x}^{(t)}$ converge to the solution in at most n steps.

- CG algorithm (economical version): saves one matrix-vector multiplication.

  Given $\boldsymbol{x}^{(0)}$
  Initialize: $\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$, $\boldsymbol{p}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$, $t = 0$
  while $\boldsymbol{r}^{(0)} \neq \boldsymbol{0}$ **do**
  $\alpha^{(t)} \leftarrow \frac{\boldsymbol{r}^{(t)\top}\boldsymbol{r}^{(t)}}{\boldsymbol{p}^{(t)\top}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
  $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
  $\boldsymbol{r}^{(t+1)} \leftarrow \boldsymbol{r}^{(t)} + \alpha^{(t)}\boldsymbol{A}\boldsymbol{p}^{(t)}$
  $\beta^{(t+1)} \leftarrow \frac{\boldsymbol{r}^{(t+1)}\boldsymbol{r}^{(t+1)}}{\boldsymbol{r}^{(t)}\boldsymbol{r}^{(t)}}$
  $\boldsymbol{p}^{(t+1)} \leftarrow -\boldsymbol{r}^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
  $t \leftarrow t + 1$
  end while

- Computation cost per iteration is one matrix vector multiplication: $\boldsymbol{A}\boldsymbol{p}^{(t)}$.

  Consider PageRank problem, $\boldsymbol{A}$ has dimension $n \approx 10^{10}$ but is highly structured (sparse + low rank). Each matrix vector multiplication takes $O(n)$.

- Theorem: If $\boldsymbol{A}$ has r distinct eigenvalues, $\boldsymbol{x}^{(t)}$ converges to solution $\boldsymbol{x}^*$ in at most r steps.

Pre-conditioned conjugate gradient (PCG)

- Summary of conjugate gradient method for solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ or equivalently minimizing $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$:
  - Each iteration needs one matrix vector multiplication: $\boldsymbol{A}\boldsymbol{p}^{(t+1)}$. For structured $\boldsymbol{A}$, often $O(n)$ cost per iteration.
  - Guaranteed to converge in n steps.
- Two important bounds for conjugate gradient algorithm:

  Let $\lambda_1 \leq \cdots \leq \lambda_n$ be the ordered eigenvalues of a pd $\boldsymbol{A}$.

$$\begin{aligned} \|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2 &\leq \left(\frac{\lambda_{n-t} - \lambda_1}{\lambda_{n-t} + \lambda_1}\right)^2 \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2 \\ \|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2 &\leq 2\left(\frac{\sqrt{\kappa(\boldsymbol{A})} - 1}{\sqrt{\kappa(\boldsymbol{A})} + 1}\right)^t \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_{\boldsymbol{A}}^2, \end{aligned}$$

  where $\kappa(\boldsymbol{A}) = \lambda_n/\lambda_1$ is the condition number of $\boldsymbol{A}$.

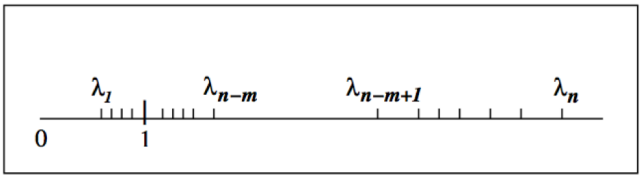


Figure 5.3 Two clusters of eigenvalues.

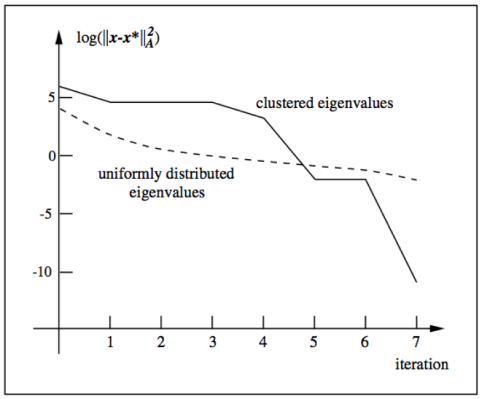


Figure 5.4 Performance of the conjugate gradient method on (a) a problem in which five of the eigenvalues are large and the remainder are clustered near 1, and (b) a matrix with uniformly distributed eigenvalues.

- Messages:
  - Roughly speaking, if the eigenvalues of $\boldsymbol{A}$ occur in r distinct clusters, the CG iterates will *approximately* solve the problem after $O(r)$ steps.
  - $\boldsymbol{A}$ with a small condition number ($\lambda_1 \approx \lambda_n$) converges fast.
- *Pre-conditioning*: Change of variables $\hat{\boldsymbol{x}} = \boldsymbol{C}\boldsymbol{x}$ via a nonsingular $\boldsymbol{C}$ and solve

$$(\boldsymbol{C}^{-T}\boldsymbol{A}\boldsymbol{C}^{-1})\hat{\boldsymbol{x}} = \boldsymbol{C}^{-T}\boldsymbol{b}.$$

  Choose $\boldsymbol{C}$ such that
  - $\boldsymbol{C}^{-T}\boldsymbol{A}\boldsymbol{C}^{-1}$ has small condition number, or
  - $\boldsymbol{C}^{-T}\boldsymbol{A}\boldsymbol{C}^{-1}$ has clustered eigenvalues
  - Inexpensive solution of $\boldsymbol{C}^T\boldsymbol{C}\boldsymbol{y} = \boldsymbol{r}$
- Preconditioned CG does not make use of $\boldsymbol{C}$ explicitly, but rather the matrix $\boldsymbol{M} = \boldsymbol{C}^T\boldsymbol{C}$.
- Preconditioned CG (PCG) algorithm:

  Given $\boldsymbol{x}^{(0)}$, pre-conditioner $\boldsymbol{M}$
  $\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}$
  solve $\boldsymbol{M}\boldsymbol{y}^{(0)} = \boldsymbol{r}^{(0)}$ for $\boldsymbol{y}^{(0)}$
  $\boldsymbol{p}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$, $t = 0$
  while $\boldsymbol{r}^{(0)} \neq \boldsymbol{0}$ **do**
    $\alpha^{(t)} \leftarrow \frac{\boldsymbol{r}^{(t)\,\mathsf{T}}\boldsymbol{y}^{(t)}}{\boldsymbol{p}^{(t)\,\mathsf{T}}\boldsymbol{A}\boldsymbol{p}^{(t)}}$
    $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
    $\boldsymbol{r}^{(t+1)} \leftarrow \boldsymbol{r}^{(t)} + \alpha^{(t)}\boldsymbol{A}\boldsymbol{p}^{(t)}$
    Solve $\boldsymbol{M}\boldsymbol{y}^{(t+1)} \leftarrow \boldsymbol{r}^{(t+1)}$ for $\boldsymbol{y}^{(t+1)}$
    $\beta^{(t+1)} \leftarrow \frac{\boldsymbol{r}^{(t+1)\,\mathsf{T}}\boldsymbol{y}^{(t+1)}}{\boldsymbol{r}^{(t)\,\mathsf{T}}\boldsymbol{r}^{(t)}}$
    $\boldsymbol{p}^{(t+1)} \leftarrow -\boldsymbol{y}^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
    $t \leftarrow t + 1$
  end while

  Remark: Only extra cost in the pre-conditioned CG algorithm is the need to solve the linear system $\boldsymbol{M}\boldsymbol{y} = \boldsymbol{r}$.

- Pre-conditioning is more like an art than science. Some choices include
  - Incomplete Cholesky. $\boldsymbol{A} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T$, where $\tilde{\boldsymbol{L}}$ is a sparse approximate Cholesky factor. Then $\tilde{\boldsymbol{L}}^{-1}\boldsymbol{A}\tilde{\boldsymbol{L}}^{-T} \approx \boldsymbol{I}$ (perfectly conditioned) and $\boldsymbol{M}\boldsymbol{y} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T\boldsymbol{y} = \boldsymbol{r}$ is easy to solve.
  - Banded pre-conditioners.
  - Choose $\boldsymbol{M}$ as a coarsened version of $\boldsymbol{A}$.
  - *Subject knowledge*. Knowledge about the structure and origin of a problem is often the key to devising efficient pre-conditioner. For example, see recent work of Stein et al. (2012) for pre-conditioning large covariance matrices. `http://epubs.siam.org/doi/abs/10.1137/110834469`

More buzzwords and softwares

Here are a few variants of CG that we should at least know the names and what they are for (so we can Google later in need).

- MINRES (minimum residual method): symmetric indefinite $\boldsymbol{A}$.
- Bi-CG (bi-conjugate gradient): unsymmetric $\boldsymbol{A}$.
- Bi-CGSTAB (Bi-CG stabilized): improved version of Bi-CG.
- GMRES (generalized minimum residual method): current *de facto* method for unsymmetric $\boldsymbol{A}$. E.g., PageRank problem.
- Lanczos method: top eigen-pairs of a large symmetric matrix.
- Arnoldi method: top eigen-pairs of a large unsymmetric matrix.
- Lanczos bidiagonalization algorithm: top singular triples of large matrix.

  Remark: For Lanczos/Arnoldi methods, the critical computation is still matrix vector multiplication $\boldsymbol{A}\boldsymbol{v}$.

Softwares:

- MATLAB:

- Iterative methods for solving linear equations:
  `pcg`, `bicg`, `bicgstab`, `gmres`, ...
- Iterative methods for top eigen-pairs and singular pairs:
  `eigs`, `svds`, ...
- Pre-conditioner:
  `cholinc`, `luinc`, ...

Get familiar with the reverse communication interface (RCI) for utilizing iterative solvers:

```
x = gmres(A, b)
x = gmres(@Afun, b)
eigs(A)
eigs(@Afun)
```

- Consider the PageRank problem. We want to find the top left eigenvector of the transition matrix

$$\boldsymbol{P} = p\boldsymbol{R}^{+}\boldsymbol{A} + \boldsymbol{z}\boldsymbol{1}_n^{\mathsf{T}},$$

where $\boldsymbol{R} = \mathrm{diag}(r_1, \ldots, r_n)$ and $z_j = (1-p)/n$ if $r_i > 0$ and $1/n$ if $r_i = 0$. Size of $\boldsymbol{P}$ can be huge: $n \approx 40$ billion web pages. How to call the `gmres` or `eigs` function?

- R: Try Google and good luck ...

(Nonlinear) Conjugate gradient method

- *Linear* conjugate gradient method is for solving linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, or equivalently, minimizing $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$.
- *Nonlinear* conjugate gradient is for nonlinear optimization

$$\text{minimize } f(\boldsymbol{x}).$$

- History: Fletcher and Reeves in 60s.
- Fletcher-Reeves CG for nonlinear minimization:

Given $\boldsymbol{x}^{(0)}$
Evaluate $\nabla f^{(0)} = \nabla f(\boldsymbol{x}^{(0)})$
Set $\boldsymbol{p}^{(0)} \leftarrow -\nabla f^{(0)}$, $t \leftarrow 0$
while $\nabla f^{(t)} \neq \mathbf{0}$ **do**
  Compute $\alpha^{(t)}$ and set $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
  Evaluate $\nabla f^{(t+1)} = \nabla f(\boldsymbol{x}^{(t+1)})$
  $\beta^{(t+1)} \leftarrow \frac{df^{(t+1)}\nabla f^{(t+1)}}{df^{(t)}\nabla f^{(t)}}$
  $\boldsymbol{p}^{(t+1)} \leftarrow -\nabla f^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
  $t \leftarrow t + 1$
end while

- Most cost is evaluation of objective function and its gradient. No matrix operations are needed. Appealing for large nonlinear optimization problems.
- Line search (choose $\alpha^{(t)}$) is necessary to get a descending algorithm.