Numerical and Scientific Computing (CSL361) **Major** **Nov 22, 2008**

Note: Maxm marks : 120. All questions carry equal marks. Maxm time: 2 hrs (strict). Please keep your answers short and together.

---

1. **Some general wisdom:**

   (a) Suppose you are using an iterative method like steepest descent to minimize a non-linear function $f(x)$ and you need to choose a convergence test. Would it be better to terminate the iteration when you find an iterate x_k for which $|f(x_k) - f(x_{k-1})|$ is small, or when $|x_k - x_{k-1}|$ is small? Why?

   (b) Let f be twice continuously differentiable in a region $\Omega \subset \mathbb{R}^n$. Show that a sufficient condition for a point x^* in the interior of Ω to be a local minimum point of f is that $\nabla f(x) = 0$ and that f is locally convex at x^*.

2. **Some optimization:** Prove the following for the *conjugate gradient* algorithm given as: $d_0 = r_0 = b - Ax_0$; $x_{k+1} = x_k + \alpha_k d_k$; $\alpha_k = \frac{r_k^T d_k}{d_k^T A d_k}$; $d_{k+1} = r_{k+1} + \beta_k d_k$; $\beta_k = -\frac{r_{k+1} A d_k}{d_k^T A d_k}$.

   The conjugate gradient algorithm is a conjugate direction procedure. Prove (by induction) that if it doesn't terminate at x_k, then

   (a) $[r_0, r_1, \ldots, r_k] = [r_0, Ar_0, \ldots, A^k r_k]$

   (b) $[d_0, d_1, \ldots, d_k] = [r_0, Ar_0, \ldots, A^k r_k]$

   (c) $d_k^T A d_i = 0$ for $i \leq k-1$

   (d) $\alpha_k = \frac{r_k^T r_k}{d_k^T A d_k}$

   (e) $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$

   Argue why is it a good procedure for solving sparse linear systems.

3. **Some least squares:** The minimal residual solution of the least squares problem $\mathbf{Ax} \approx \mathbf{b}$ for an over-determined system is given as

$$\min_{\mathbf{b}+\mathbf{r} \in \mathit{range}(\mathbf{A})} ||\mathbf{r}||_2^2$$

   where $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$. It is well known that this solution is biased towards errors in $\mathbf{b}$. For example, for fitting a straight line $y = c + mx$ through a set of points $\{x_i, y_i\}_{i=1,n}$, the function

$$f(c, m) = ||\mathbf{y} - c\mathbf{e} - m\mathbf{x}||_2^2 = ||\mathbf{y} - \mathbf{A}\begin{pmatrix} c \\ m \end{pmatrix}||_2^2$$

   (where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$, $\mathbf{y} = (y_1, y_2, \ldots, y_n)^t$, $\mathbf{e} = (1, 1, \ldots, 1)^t$ and $\mathbf{A} = [\mathbf{e} \;\; \mathbf{x}]$) measures square of vertical distances (along y axis) from the points to the straight line, thereby making the tacit assumption that model errors are confined to observed y coordinates only.

   When error is also present in $\mathbf{A}$ (e.g., the x coordinate also), can you argue that it may be better to minimize the the perpendicular distance of the points to the straight line and formulate this as a problem of the type: *minimize* $||Q\mathbf{n}||^2$ *subject to* $||\mathbf{n}|| = 1$? (Hint: you may find it useful to center the points by subtracting the mean).

Can you show that the above is equivalent to the *total least squares* formulation

$$\min_{\mathbf{b}+\mathbf{r} \ \in \ range(\mathbf{A}+\mathbf{E})} || \ [\mathbf{E} \quad \mathbf{r}] \ ||_2^2$$

and argue that the *TLS* solution can be derived from the best rank n approximation of $[\mathbf{A} \ \ \mathbf{b}]$.

4. **The "out of the blue" one:** Let us re-visit the *web search* problem, this time with the aim of finding a good algorithm to *rank* the pages. A set V of hyper-linked pages can be viewed as a directed graph $G = (V, E)$: the nodes correspond to pages, and a directed edge $(p, q) \in E$ if there is a link from p to q. Given a query string σ we can use any text based search scheme to identify a subgraph G_σ which is rich in σ, i.e., G_σ contains pages with σ and pages linked from these pages. The problem then is to find good *authorities* (pages to which large number of pages link) and *hubs* (pages that link to many related authorities).

   We can design an iterative algorithm that maintains and updates numerical weights for each page. With each page p we can associate a non-negative *authority weight* x_p and a non-negative *hub weight* y_p and maintain invariant their square sums to 1, i.e., $\sum_p (x_p)^2 = 1$ and $\sum_p (y_p)^2 = 1$. We can then view pages with large x and y values as being "better" authorities and hubs respectively. The iteration can be designed on the following principle: If p points to many pages with large x values, then it should receive a large y value; and if p is pointed to by many pages with large y values, then it should receive a large x value. This motivates the definition of two operations $\mathcal{A}$ and $\mathcal{H}$ on the weights as follows:

$$\begin{aligned} \mathcal{A}: &\quad x_p \leftarrow \textstyle\sum_{q:(q,p)\in E} y_q \\ \mathcal{H}: &\quad y_p \leftarrow \textstyle\sum_{q:(p,q)\in E} x_q \end{aligned}$$

   Of course, the updates have to be followed by normalization. Given the *adjacency matrix* A of the graph G_σ (the $(i, j)^{th}$ entry of A is 1 if $(p_i, p_j) \in G_\sigma$, 0 otherwise), the algorithm can be described as follows.

   $z =$ a normalized version of $(1, 1, \ldots, 1)^t$
   $x^{(0)} = z$; $y^{(0)} = z$;
   for $i = 1, 2, \ldots$
       Apply $\mathcal{A}$ operations: $x_i \leftarrow A^T y^{(i-1)}$
       Apply $\mathcal{H}$ operations: $y_i \leftarrow A x^{(i)}$
       Normalize $x^{(i)}$ and $y^{(i)}$

   What is happening? Can we find "good" *authorities* and *hubs* using the above algorithm and rank them? Under what conditions? What would be the fixed points?

   This algorithm is due to Prof. Jon Kleinberg, Department of Computer Science, Cornell University.