Metoder för stora problem

Newton- och Kvasi-newton-metoderna har relativt stora krav på minne ($n^2/2$ element) och beräkningskraft per iteration ($\mathcal{O}(n^3)$ resp $\mathcal{O}(n^2)$).

På en snabb arbetsstation (Sun UltraSparc II, 200 Mhz) kan kostnaderna se ut så här:

n	Minne ($n^2/2$)	Tid $\mathcal{O}(n^3)$	Tid $\mathcal{O}(n^2)$
100	39 kB	13μs	13μs
1000	3.8 MB	13.9 s	1.3 s
10000	381 MB	3.86 h	2.2 min
100000	37 GB	161 d	3.6 h

Som synes får metoderna problem när n blir stort.

Metoder för stora problem, forts.

Det finns dock iterativa metoder för lösningar av linjära ekvationssystem som har betydligt lägre krav på minne och beräkningar per iteration.

Vi kommer att studera *konjugerade-gradient*-metoden för lösning av

$$Ax = b,$$

där A är symmetrisk och positivt definit.

Den kan också ses som en metod för att lösa

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

eftersom $\nabla f(x) = Ax - b$ och $\nabla^2 f(x) = A$ är positivt definit.

Konjugerade vektorer

Konjugerade gradient-metoden får sitt namn från att den genererar en mängd av vektorer $\{p_i\}$ som är *konjugerade* med avseende på en matris A, dvs

$$p_i^T A p_j = 0 \text{ om } i \neq j.$$

För specialfallet $A = I$ är de konjugerade vektorerna ortogonala.

Konjugerade vektorer, forts.

Antag att en mängd konjugerade vektorer $\{p_i\}$ är känd. Betrakta vektorn y som är en summa av $m+1$ sådana vektorer:

$$y = \sum_{i=0}^{m} \alpha_i p_i$$

och beräkna $f(y) = \frac{1}{2}y^T Ay - b^T y$:

$$\begin{aligned}
f(y) &= \frac{1}{2}\left(\sum_{i=0}^{m} \alpha_i p_i\right)^T A \left(\sum_{j=0}^{m} \alpha_j p_j\right) - b^T \left(\sum_{i=0}^{m} \alpha_i p_i\right) \\
&= \frac{1}{2}\sum_{i=0}^{m}\sum_{j=0}^{m} \alpha_i \alpha_j p_i^T A p_j - \sum_{i=0}^{m} \alpha_i b^T p_i \\
&= \frac{1}{2}\sum_{i=0}^{m} \alpha_i^2 p_i^T A p_i - \sum_{i=0}^{m} \alpha_i b^T p_i \\
&= \sum_{i=0}^{m} (\frac{1}{2}\alpha_i^2 p_i^T A p_i - \alpha_i b^T p_i).
\end{aligned}$$

Konjugerade vektorer, forts.

Vi kan då lätt minimera funktionen över alla vektorer på denna form:

$$\begin{aligned}\min_y f(y) &= \min_{\{\alpha_i\}} f\left(\sum_{i=0}^{m} \alpha_i p_i\right)\\ &= \min_{\{\alpha_i\}} \sum_{i=0}^{m} (\frac{1}{2}\alpha_i^2 p_i^T A p_i - \alpha_i b^T p_i)\\ &= \sum_{i=0}^{m} \min_{\alpha_i}(\frac{1}{2}\alpha_i^2 p_i^T A p_i - \alpha_i b^T p_i),\end{aligned}$$

vilket innebär att problemet reducerats till en summa av en-dimensionella minimeringsproblem.

Konjugerade vektorer, forts.

Vart och ett av dessa problem går att lösa genom att sätta derivatan med avseende på α_i och lika med noll:

$$\alpha(p_i^T A p_i) - b^T p_i = 0 \Rightarrow \alpha_i = \frac{b^T p_i}{p_i^T A p_i}.$$

Konjugerade gradient-metoden bestämmer iterativt mängden konjugerade vektorer $\{p_i\}$ och deras koefficienter $\{\alpha_i\}$.

Konjugerade gradient-metoden

Algoritm för konjugerade gradient-metoden:

- Låt $x_0 = 0, r_0 = b - Ax_0, p_{-1} = 0, \beta_0 = 0$ och bestäm konvergenstoleransen ϵ.
- Upprepa för $i = 0, 1, \ldots$
  - Om $\|r_i\| < \epsilon$, avsluta.
  - Om $i > 0$ beräkna $\beta_i = r_i^T r_i / r_{i-1}^T r_{i-1}$.
  - Låt $p_i = r_i + \beta_i p_{i-1}$.
  - Låt $\alpha_i = r_i^T r_i / p_i^T A p_i$.
  - Låt $x_{i+1} = x_i + \alpha_i p_i$.
  - Låt $r_{i+1} = r_i - \alpha_i A p_i$.

Konjugerade gradient-metoden, forts.

Det går att visa att:

- $r_i^T r_j = 0$ för $i \neq j$, dvs residualerna är ortogonala.
- $r_i^T p_j = 0$ för $i > j$, dvs alla residualer är ortogonala mot alla "äldre" sökriktningar.
- $p_i^T A p_j$ för $i \neq j$, dvs alla sökriktningar är konjugerade.

Konjugerade gradient-metoden, forts.

Det går också att visa att i exakt aritmetik konvergerar konjugerade gradient-metoden i k iterationer, där k är antalet distinkta egenvärden till A ($k \leq n$).

I finit aritmetik gäller dock inte detta, då ackumulerade avrundningsfel ger avsteg från egenskaperna ovan och därför kräver fler iterationer.

Konjugerade gradient-metoden konvergerar linjärt med konvergenskonstanten

$$C = \frac{\|x_{i+1}\|_A}{\|x_i\|_A} \leq \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1},$$

där felnormen definieras som $\|y\|_A^2 = \frac{1}{2} y^T A y$.

Pre-konditionering

Detta kan leda till långsam konvergens om $\kappa(A)$ är stort, vilket inte är ovanligt. Därför används metoden ofta med *pre-konditionering*, vilket innebär är i stället för att lösa ekvationssystemet

$$Ax = b \tag{1}$$

löser man det ekvivalenta ekvationssystemet

$$M^{-1}Ax = M^{-1}b,$$

där matrisen M kallas *pre-konditionerare*.

Pre-konditionerade konjugerade gradient-metoden

- Låt $x_0 = 0, r_0 = b - Ax_0, p_{-1} = 0, \beta_0 = 0$ och bestäm konvergenstoleransen ϵ.

- Upprepa för $i = 0, 1, \ldots$
  - Om $\|r_i\| < \epsilon$, avsluta.
  - Låt $z_i = M^{-1} r_i$.
  - Om $i > 0$ beräkna $\beta_i = r_i^T z_i / r_{i-1}^T z_{i-1}$.
  - Låt $p_i = z_i + \beta_i p_{i-1}$.
  - Låt $\alpha_i = r_i^T z_i / p_i^T A p_i$.
  - Låt $x_{i+1} = x_i + \alpha_i p_i$.
  - Låt $r_{i+1} = r_i - \alpha_i A p_i$.

Pre-konditionering, forts.

M introduceras i syfte att snabba upp konvergensen, vilket kan göras genom att minska $\kappa(M^{-1}A)$ eller antalet distinkta egenvärden i $M^{-1}A$.

Detta uppfylls fullt ut av $M = A \Rightarrow M^{-1}A = I$, men det innebär att vi först måste lösa ekvation (1), som vi håller på att lösa!

Alltså ska M "likna" A men vara lättare att invertera.

Vanliga val av pre-konditionerare är diagonala matriser eller (om A är gles) någon faktorisering där ifyllningen ignorerats (s k icke komplett faktorisering), dvs om

$$LL^T = A,$$

så är

$$L_0 L_0^T \approx A,$$

där L_0 har samma fyllnadsgrad som A.

Trunkerad-Newtons metod

Trunkerad-Newtons metod löser problemet

$$\min f(x)$$

genom att i varje iterationssteg finna en *approximativ* lösning till Newtons ekvation

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

med hjälp av någon iterativ metod, vanligen konjugerade gradient-metoden.

Den iterativa metoden stoppas innan den exakta lösningen uppnätts, därav namnet.

Givet den approximativa sökriktningen p_k görs sedan en linjesökning på vanligt sätt.

Trunkerad-Newtons metod, forts.

Bortsett från produkten $Av = \nabla^2 f(x_k)v$ används endast vektoroperationer ($\mathcal{O}(n)$) och endast vektorer behöver lagras.

Om produkten Av kan beräknas billigt (m a p tid och minne) för alla v så gör denna minimala förändring i algoritmen metoden lämplig även för stora problem.

Detta uppfylls t ex om A är gles (stora problem är ofta glesa), och/eller om A har en sådan struktur att elementen a_{ij} kan beräknas som en funktion av i och j.

Går inte detta kan produkten $\nabla^2 f(x_k)v$ approximeras med

$$\nabla^2 f(x_k)v \approx \frac{\nabla f(x_k + hv) - \nabla f(x_k)}{h},$$

för något litet h, dvs en differensapproximation som endast kräver vektoroperationer.

Avvägningar

Ju fler "inre" iterationer vi använder för att ta fram en approximation till lösningen av Newtons ekvation, desto "bättre" sökriktning borde vi kunna förvänta oss och därigenom färre "yttre" iterationer.

Speciellt, låt den inre iterationen avbrytas då

$$\|\nabla^2 f(x_k) + \nabla f(x_k)\| \leq \phi_k \|\nabla f(x_k)\|,$$

för någon tolerans $\phi_k \geq 0$.

Om $\phi_k = c < 1$ så konvergerar trunkerad-newtons metod linjärt med konvergenskonstant c. Om $\phi_k \to 0$ så konvergerar den superlinjärt. $\phi_k \leq \|\nabla f(x_k)\|$ ger kvadratisk konvergens. Specialfallet $\phi_k = 0$ ger oss Newtons metod.

Avvägningar, forts.

Snabbare "yttre" konvergens kostar naturligtvis fler "inre" iterationer, vilket innebär att en avvägning måste göras för att få snabbast möjliga total konvergens.

Dessutom kan en mer kostsam pre-konditionerare (t ex en icke komplett Cholesky-faktorisering i stället för en diagonalmatris) ibland ge snabbare total konvergens.

För stora problem lönar sig det ofta att lägga ner tid på att "trimma" algoritmerna då beräkningstiderna kan bli långa.