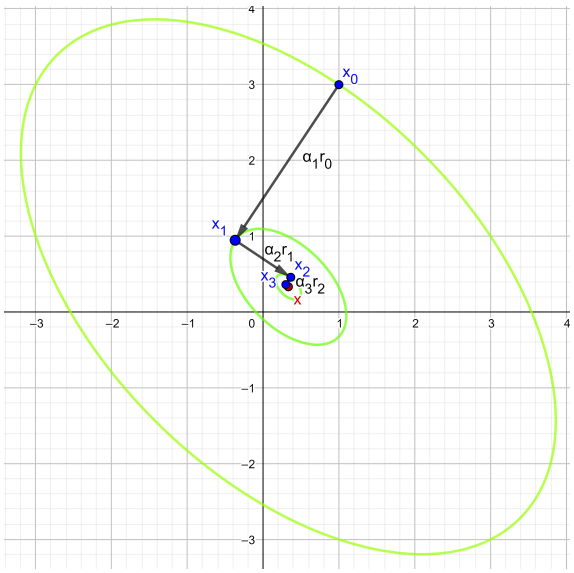
$$x^{(k)} = x^{(k-1)} + \alpha_k v^{(k)} = x^{(k-1)} + \frac{v^{(k)T} r^{(k-1)}}{v^{(k)T} A v^{(k)}} v \tag{1}$$

Metóda združených smerov

Predchádzajúcu metódu môžeme v rovine geometrický opísať nasledovne. Vrstevnice kvadratickej funkcie F sú elipsy (lebo A je kladne definitná symetrická matica). Bod $x^{(0)}$ v ktorom začíname algoritmus teda leží na nejakej elipse $e^{(0)}$. Zostrojíme kolmicu na túto elipsu v bode $x^{(0)}$. To bude priamka ktorej smerový vektor je vektor rezidua $r^{(0)}$, čo je mimochodom aj nejaký násobok gradientu f v bode $x^{(0)}$. Táto priamka je dotyčnicou nejakej ďalšej vrstevnice $e^{(1)}$ a bod dotyku je nové približné riešenie $x^{(1)}$. Ďalšie priblíženie k riešeniu dostaneme opakovaním tejto "konštrukcie"(obr. 1). V



Obr. 1: Metóda najväčšieho spádu s vrstevnicami

prípade viacrozmerného prípadu je postup podobný. Konštruujeme kolmice na vrstevnice podobným spôsobom, a tým konštruujeme postupne minimá kvadratickej funkcie v daných smeroch.

Čo si môžeme všimnúť z vyššie uvedenej konštrukcie je, že bod $x^{(2)}$ nie je minimálny vzhľadom na smer určený vektorom $r^{(0)}$, pretože posunom v smere $r^{(2)}$, čo je len nejaký násobok vektora $r^{(0)}$ dostaneme $x^{(3)}$, v ktorom má f menšiu hodnotu ako v predošlom $x^{(2)}$. Ponúka sa otázka, či nevieme vektory $v^{(k)}$ v iteračnom procese (1) voliť tak, aby sme v danom iteračnom kroku získali bod minima danej kvadratickej funkcie, ktorý bude minimálny

vzhľadom na všetky predchádzajúce smery. Poďme túto myšlienku formalizovať.

V k-tom kroku minimalizácie dostávame

$$x^{(k)} = x^{(k-1)} + \alpha_k v^{(k)}.$$

Ak zvolíme niektorý z predchádzajúcich smerov $v^{(i)}$, $i < k$, ako nový smer posunu, tak

$$x^{(k+1)} = x^{(k)} + \alpha v^{(i)},$$

pričom

$$\alpha = \frac{v^{(i)T} r^{(k)}}{v^{(i)T} A v^{(i)}}.$$

Ak ale predpokladáme, že v $x^{(k)}$ je minimum vzhľadom na predchádzajúce smery, musí byť $\alpha = 0$, a teda

$$\begin{aligned} 0 &= v^{(i)T} r^{(k)} = v^{(i)T}(b - Ax^{(k)}) = v^{(i)T}(b - A(x^{(k-1)} + \alpha_k v^{(k)})) \\ &= v^{(i)T}((b - A(x^{(k-1)})) + \alpha_k A v^{(k)}) = v^{(i)T}(r^{(k-1)} + \alpha_k A v^{(k)}) \\ &= v^{(i)T} r^{(k-1)} + v^{(i)T} \alpha_k A v^{(k)} = v^{(i)T} \alpha_k A v^{(k)}. \end{aligned}$$

Posledná rovnosť vyplýva z toho, že $v^{(i)T} r^{(k-1)}$, pretože sme predpokladali, že $x^{(k)}$ je minimum vzhľadom na predchádzajúce smery. Preto pre nulovosť α potrebujeme $v^{(i)T} A v^{(k)} = 0$.

Keďže A je symetrická, kladne definitná, výraz $\langle x, y \rangle_A = y^T A x$ má všetky vlastnosti skalárneho súčinu. Tým pádom vzťah $v^{(i)T} A v^{(k)} = 0$ znamená, že vektory $v^{(i)}$ a $v^{(k)}$ sú A-ortogonálne.

Teraz môžeme zostrojiť algoritmus na riešenie systému $Ax = b$ s maticou rozmeru $n \times n$.

- Zvolíme A-ortogonálnu bázu $v^{(1)}, \ldots, v^{(n)}$.
- Zvolíme $x^{(0)}$.
- $r^{(0)} = b - Ax^{(0)}$.
- Pre $k = 1, \ldots, n$ spočítame

$$\begin{aligned} \alpha_k &= \frac{v^{(k)T} r^{(k-1)}}{v^{(k)T} A v^{(k)}} \\ x^{(k)} &= x^{(k-1)} + \alpha_k v^{(k)} \\ r^{(k)} &= b - Ax^{(k)} = b - A(x^{(k-1)} + \alpha_k v^{(k)}) = r^{(k-1)} - \alpha_k a v^{(k)} \end{aligned}$$

Po n krokoch dostávame minimum funkcie v n rôznych smeroch, čo znamená, že sme získali presné riešenie.

A-ortogonálnu bázu $\{v^{(i)}\}$ je možné získať z ľubovoľnej bázy $\{u^{(i)}\}$ Gramovou-Schmidtovou A-ortogonalizáciou.

$$\begin{aligned} v^{(1)} &= u^{(1)} \\ v^{(k+1)} &= u^{(k+1)} - \sum_{j=1}^{k} \frac{\langle u^{(k+1)}, v^{(j)} \rangle_A}{\langle v^{(j)}, v^{(j)} \rangle_A} v^j. \end{aligned}$$

Nevýhodou je že na výpočet $v^{(k+1)}$ potrebujeme mať uložené všetky vektory $v^{(1)},\ldots,v^{(k)}$, čo podstatne zvyšuje nároky na pamäť. Taktiež samotný výpočet bázy má zložitosť $O(n^3)$, čo nie je rýchlejšie ako GEM.

Miernou modifikáciou predošlého postupu získame nasledujúcu metódu.

Metóda združených gradientov

Idea tejto metódy je taká, A-ortogonálnu bázu z predošlého postupu počítame priebežne z reziduí metódy najväčšieho spádu.

- Zvolíme $x^{(0)}$, $r^{(0)} = b - Ax^{(0)}$, $v^{(1)} = r^{(0)}$.
- Pre $k = 1, \ldots, n$

$$\begin{aligned} \alpha_k &= \frac{v^{(k)T} r^{(k-1)}}{v^{(k)T} A v^{(k)}} \\ x^{(k)} &= x^{(k-1)} + \alpha_k v^{(k)} \\ r^{(k)} &= r^{(k-1)} - \alpha_k A v^{(k)} \\ v^{(k+1)} &= r^{(k)} - \sum_{j=1}^{k} \frac{\langle r^{(k)}, v^{(j)} \rangle_A}{\langle v^{(j)}, v^{(j)} \rangle_A} v^{(j)} \end{aligned}$$

Z predchádzajúcej časti máme $v^{(i)T} r^{(k)} = 0$ pre $i \leq k$, teda $v^{(i)} \perp r^{(k)}$ pre $i \leq k$. Taktiež vektory $\{v^{(1)}, \ldots, v^{(k)}\}$ generujú ten istý priestor ako $\{r^{(1)}, \ldots, r^{(k-1)}\}$. Z toho dostávame, že $r^{(k)} \perp r^{(i)}$ pre každé $i < k$. Potom

$$\langle r^{(k)}, v^{(j)} \rangle_A = v^{(j)T} A r^{(k)} = r^{(k)T} A v^{(j)} = r^{(k)T} \cdot \frac{r^{(j-1)} - r^{(j)}}{\alpha_j} = 0$$

pre $j < k$. Teda $r^{(k)}$ je A-ortogonálny na všetky v^j okrem posledného. Tým dostaneme modifikáciu predošlého algoritmu.

- Zvolíme $x^{(0)}$, $r^{(0)} = b - Ax^{(0)}$, $v^{(1)} = r^{(0)}$.
- Pre $k = 1, \ldots, n$

$$\begin{aligned} \alpha_k &= \frac{v^{(k)T} r^{(k-1)}}{v^{(k)T} A v^{(k)}} = \frac{r^{(k-1)T} r^{(k-1)}}{v^{(k)T} A v^{(k)}} && \text{(alpha)} \\ x^{(k)} &= x^{(k-1)} + \alpha_k v^{(k)} \\ r^{(k)} &= r^{(k-1)} - \alpha_k A v^{(k)} \\ v^{(k+1)} &= r^{(k)} - \sum_{j=1}^{k} \frac{\langle r^{(k)}, v^{(j)} \rangle_A}{\langle v^{(j)}, v^{(j)} \rangle_A} v^{(j)} = r^{(k)} + \frac{r^{(k)T} r^{(k)}}{r^{(k-1)T} r^{(k-1)}} v^{(j)} && \text{(vector)} \end{aligned}$$

QR-rozklad

Riešenie systémov $Ax = b$ použitím LU-rozkladu má nevýhodu v tom, že matice L a U z rozkladu môžu mať horšie číslo podmienenosti ako samotná matica A. Preto je často lepšie použiť iný typ rozkladu. Maticu systému rozložíme na súčin ortogonálnej matice Q a stupňovitej matice R (v prípade že A je štvorcová, tak R je horná trjuholníková). Pre maticu A rozmeru $m \times n$ dostávame rozklad

$$A = QR,$$

kde Q je ortogonálna matica rozmeru $m \times m$ a R je stupňovitá rozmeru $m \times n$.

Čo sa týka výpočtovej zložitosti hľadania riešenia systému $Ax = QRx = b$, máme

$$x = R^{-1} Q^T b,$$

čo vieme urobiť v $O(n^2)$. Teda celkovú rýchlosť určuje rýchlosť výpočtu QR-rozkladu, ktorý je $O(n^3)$. Uvedieme niekoľko algoritmov na výpočet QR-rozkladu matice A.

QR-rozklad pomocou GSO

Nech A je matica rozmeru $m \times n$, pričom $m \geq n$. Potom A a jej QR-rozklad vieme zapísať ako

$$A = QR = \begin{pmatrix} Q_1 & | & Q_2 \end{pmatrix} \begin{pmatrix} R_1 \\ O \end{pmatrix}$$

kde Q_1 je $m \times n$ matica s ortonormálnymi stĺpcami, R_1 je horná trojuholníková matica a O je nulová matica. Potom $Q_1 R_1 = A$, čo znamená, že stĺpce matice A sú lineárne kombinácie stĺpcov matice Q_1. Ak A má stĺpcovú hodnosť n, tak R_1 je regulárna a $Q_1 = A R_1^{-1}$. V tom prípade ale aj stĺpce matice

Q_1 sú lineárne kombinácie stĺpcov matice A, čo spolu s predošlým faktom implikuje, že A a Q_1 majú rovnaké stĺpcové priestory. Maticu Q_1 teda môžeme ľahko získať pomocou Gramovej-Schmidtovej ortogonalizácie. Ak i-ty stĺpec matice A označíme a_i a i-ty stĺpec matice Q_1 označíme q_i,

$$\begin{aligned}
q_1 &= \frac{a_1}{\|a_1\|} && \Rightarrow \quad a_1 = q_1\|a_1\| \\
q_2 &= \frac{a_2 - \langle a_2, q_1\rangle q_1}{\|a_2 - \langle a_2, q_1\rangle\|} && \Rightarrow \quad a_2 = q_2\|z_2\| + q_1\langle a_2, q_1\rangle \\
q_3 &= \frac{a_3 - \langle a_3, q_1\rangle q_1 - \langle a_3, q_2\rangle q_2}{\|a_3 - \langle a_3, q_1\rangle q_1 - \langle a_3, q_2\rangle q_2\|} && \Rightarrow \quad a_3 = q_3\|z_3\| + q_2\langle a_3, q_2\rangle + q_1\langle a_3, q_1\rangle \\
&\vdots
\end{aligned}$$

pričom z_i označuje čitateľa vo vyjadrení q_i.

Z vyjadrení stĺpcov a_i napravo, potom môžeme vyskladať matice Q_1 a R_1.

$$Q_1 = \begin{pmatrix} q_1 & q_2 & q_3 & \cdots & q_n \end{pmatrix}, \qquad R_1 = \begin{pmatrix} \|a_1\| & \langle a_2, q_1\rangle & \langle a_3, q_1\rangle & \cdots \\ 0 & \|z_2| & \langle a_3, q_2\rangle & \cdots \\ 0 & 0 & \|z_3\| & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Modifikovaná GSO

Zapíšme

$$A = QR = \begin{pmatrix} q_1 & q_2 & \cdots & q_n \end{pmatrix} \begin{pmatrix} r_1^T \\ r_2^T \\ \vdots \\ r_n^T \end{pmatrix} = q_1 r_1^T + q_2 r_2^T + \cdots + q_n r_n^T,$$

kde q_i sú stĺpce matice Q a r_i^T sú riadky matice R.

Potom $q_1^T A = r_1^T$ takže pomocou q_1 vieme spočítať celý prvý riadok matice R. Vo všeobecnosti ak poznáme q_i máme $q_i^T A = r_i^T$. Ďalej máme

$$A - q_1 r_1^T = \begin{pmatrix} 0 & a_2^{(1)} & \ldots & a_n^{(1)} \end{pmatrix} = A^{(1)} = q_2 r_2^T + \cdots + q_n r_n^T$$

Stĺpec $a_2^{(1)}$ je druhým stĺpcom matice $q_2 r_2^T$, pretože všetky ostatné matice $q_i r_i^T$ majú prvé dva stĺpce nulové. Takže $r_{22} = \|a_2^{(1)}\|$, $q_2 = \dfrac{a_2^{(1)}}{\|a_2^{(1)}\|}$ a

$$r_2^T = q_2^T A = q_2^T (A^{(1)} + q_1 r_1^T) = q_2^T A^{(1)}.$$

Následne pokračujeme s $A^{(2)} = A - q_1 r_1^T - q_2 r_2^T$ a celý postup opakujeme.

Zhrňme celý algoritmus:

- $A^0 = A$,
- Pre k=1,. . . ,n
  1. $r_{kk} = \|a_k^{(k-1)}\|_2$,
  2. $q_k = \dfrac{a_k^{(k-1)}}{r_{kk}}$,
  3. $r_k^T = q_k^T A^{(k-1)}$,
  4. $A^{(k)} = A^{(k-1)} - q_k r_k^T$.

Nevýhoda týchto dvoch metód je, že nie sú numericky stabilné. Ak počítame s konečnou presnosťou, nemôžeme očakávať, že získaná matica Q bude skutočne ortogonálna. Preto získanú maticu Q budeme považovať za ortogonálnu, ak $\|I - Q^T Q\| < \varepsilon$, pre dostatočne malé $\varepsilon > 0$. V prípade, že stĺpce matice A sú skoro lineárne závisle, tak pri počítaní s konečnou presnosťou matica Q môže byť veľmi zle ortogonálna, čo znamená, že $\|I - Q^T Q\|$ omnoho väčšie ako zvolené ϵ.