Samenvatting [G0Q57A] - Modellering en simulatie

Arne Van Den Kerchove

21 januari 2019

Inhoudsopgave

1 Modellen en simulaties

Dit hoofdstuk als er tijd over is

2 Numerieke lineaire algebra en toepassingen

2.1 QR-factorisatie

Definitie 1 *De volle QR-factorisatie van de matrix A wordt gegeven door*

$$A = QR$$

met q een $m \times m$ orthogonale matrix en R een $m \times n$ bovendriehoeksmatrix.

2.1.1 Gram-Schmidt orthogonalisatie

Algorithm 1 Gram-Schmidt-algoritme

```
procedure QRGramSchmidt
    for j = 1 to n do
        v_j = a_j
        for i = 1 to j − 1 do
            r_ij = q_i^T a_j
            v_j = v_j − r_ij q_i
        r_jj = ||v_j||_2
        q_j = v_j / r_jj
```

Complexiteit: $\mathcal{O}(2mn^2)$
Stabiliteit: niet stabiel

2.1.2 QR-factorisatie met Givens-rotaties

Definitie 2 *Een Givens-rotatie is een $m \times m$ orthogonale matrix van de vorm*

$$G_{ij} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

met

$$c^2 + s^2 = 1$$

Om een Givens-rotatie op te stellen die plaats (j, k) 0 maakt in matrix A, kies dan een element in dezelfde kolom (bv. het element boven (j, k)) op oplaats (i, k) en maak G_{ij} met

$$c = \frac{a_{ik}}{\sqrt{a_{ik}^2 + a_{jk}^2}} \text{ en } s = \frac{a_{jk}}{\sqrt{a_{ik}^2 + a_{jk}^2}}$$

Algorithm 2 Givens-rotatie-algoritme

procedure QRGIVENS
 $Q = \mathbb{1}$
 $R = A$
 for $j = 1$ to n **do**
  for $i = m$ to $j+1$ **do**
   $c = \frac{r_{i-1,j}}{\sqrt{r_{i-1,j}^2 + r_{i,j}^2}}$
   $s = \frac{r_{i,j}}{\sqrt{r_{i-1,j}^2 + r_{i,j}^2}}$
   $r_{i,j} = 0$
   $r_{i-1,j} = \sqrt{r_{i-1,j}^2 + r_{i,j}^2}$
   for $k = j+1$ to n **do**

$$\begin{bmatrix} r_{i-1,k} \\ r_{ik} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & s \end{bmatrix} \begin{bmatrix} r_{i-1,k} \\ r_{i,k} \end{bmatrix}$$

   for $k = 1$ to m **do**

$$\begin{bmatrix} q_{k,i-1} & q_{ki} \end{bmatrix} = \begin{bmatrix} q_{k,i-1} & q_{ki} \end{bmatrix} \begin{bmatrix} c & s \\ -s & s \end{bmatrix}$$

Complexiteit: $\mathcal{O}(3mn^2 - n^3)$
Stabiliteit: stabiel

2.1.3 QR-factorisatie met kolompivotering

Indien A niet van volle rang is, is het voor de stabiliteit beter om kolompivotering toe te passen. In stap j van het QR-algoritme met Givens-rotaties verwisselen we kolom j met de kolom p waarvan de 2-norm het grootst is.

Complexiteit: $\mathcal{O}(3mn^2 - n^3)$
Stabiliteit: stabieler voor rang-deficiënte matrices

2.2 Singuliere-waardenontbinding

Definitie 3 *De singuliere-waardenontbinding van matrix A wordt gegeven door*

$$A = \hat{U}\hat{\Sigma}V^T$$

waarbij $\hat{U}$ orthonormale kolommen heeft, $\hat{\Sigma}$ een diagonaalmatrix met de singuliere waarden is en V een orthogonale matrix is.

Eigenschappen van de SVD:

- De rang van A is gelijk aan de rang van Σ is gelijk aan het aantal niet-nul singuliere waarden.

- De eerste r kolommen van U vormen een basis voor de kolomruimte van A.
- De laatste $n-r$ kolommen van V vormen een basis voor de nulruimte van A.
- $A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$
- $normA_2 = \sqrt(\sigma_1^2 + \sigma_2^2 + ... + \sigma_r^2)$
- De singuliere waarden zijn de vierkantswortels van de eigenwaarden van $A^T A$. De kolommen van V zijn de bijhorende eigenvectoren.
- De singuliere waarden zijn de vierkantswortels van de n grootste eigenwaarden van AA^T. De eerste n kolommen van U zijn de bijhorende eigenvectoren.
- Als A symmetrisch is, zijn de singuliere waarden de absolute waarden van de eigenwaarden van A.

2.2.1 Lage rangbenadering

Definitie 4 *De ϵ-rang van een matrix A wordt gedefinieerd als*

$$rang(A, \epsilon) = \min_{\|A-B\|_2 \leq \epsilon} rang(B)$$

De matrix B ligt ϵ-dicht bij A als hij een rang heeft die de kleinste is onder alle matrices die ϵ-dicht bij A liggen.

Definitie 5 *Een rang k-benadering A_k met $(k \leq r)$ van A wordt berekend door de singuliere waardenontbinding te vermenigvuldigen, maar Σ te vervangen door een diagonaalmatrix met de k grootste singuliere waarden op de diagonaal.*

Hierdoor geldt de eigenschap

$$\|A - A_k\|_2 = min_{B \in \mathbb{R}^{m \times m} \text{rang}(B) \leq k} \|A - B\|_2 = \sigma_k + 1$$

2.3 Kleinste-Kwadratenbenadering

Om de coëfficiënten te bepalen wordt een Vandermondematrix A opgesteld. De te minimaliseren fout bij KK-benadering wordt gegeven door

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|_2 = \min_{x \in \mathbb{R}^n} \sqrt{\sum_{i=1}^{m} (b_i - \sum_{j=1}^{n} a_{i,j} x_j)^2}$$

met $r = b - Ax$ de *residuvector*.
Dit probleem kan opgelost worden door x te bepalen in

$$A^T A x = A^T b$$

De Vandermondematrix A is slecht geconditioneerd. We zoeken dus andere manieren om het KK-probleem op te lossen.

2.3.1 Oplossing met QR-ontbinding

Indien de QR-factorisatie van A bekend is, kan deze gebruikt worden om een oplossing voor het KK-probleem te vinden:

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2 = \min_{x\in\mathbb{R}^n} \|b - QRx\|_2 = \min_{x\in\mathbb{R}^n} \left\|Q^T b - RAx\right\|_2$$

Aangezien vermenigvuldiging vooraan met een orthogonale matrix de norm behoudt.
De vector $Q^T b = c$ kan opgesplitst worden in de volgende componenten: $\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$
met $c_1 \in \mathbb{R}^n$ en $c_2 \in \mathbb{R}^{m-n}$
Hieruit volgt:

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2 = \min_{x\in\mathbb{R}^n} \left\| \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} x \right\|_2$$

Volgens de stelling van Pythagoras geldt:

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2^2 = \min_{x\in\mathbb{R}^n} (\left\|c_1 - \hat{R}x\right\|_2^2 + \|c_2\|_2^2)$$

De vector x met coëfficiënten met minimale fout kan dus ook bekomen worden als x de oplossing van $\hat{R}x = c_1$

Complexiteit: $\mathcal{O}(mn) + \mathcal{O}n^2$ indien QR-factorisatie bekend.
Stabiliteit: stabiel indien A van volle rang.

2.4 Eigenwaardenproblemen

2.4.1 Methode van de machten

Deze methode vindt de eigenvector bij de grootste eigenwaarde en is geschikt voor ijle matrices.

Algorithm 3 Methode der machten

```
procedure EigenPowermethod
    q_0 random
    for k = 0, 1, 2, ... do
        take γ_k such that ||q_{k+1}||_2 = 1
        q_{k+1} = A q_k / γ_k
```

q_k zal convergeren naar $\gamma_1 x_1$
Voordelen:

- Eenvoudige berekeningen
- zeer efficiënt voor ijle matrices

Nadelen:

- Zeer trage convergentie als λ_1 niet sterk dominant is.

Convergentie: lineair, afhankelijk van afstand tussen grootste eigenwaarden.

2.4.2 Deelruimte-iteratie

We itereren nu niet meer op één vector (methode der machten) maar op de volledige ruimte opgespannen door een orthonormaal stel vectoren.

Voor n eigenwaarden:

Algorithm 4 Deelruimte-iteratie

```
1: procedure EigenPartialSpace
2:     Q̂_0 = [q_1^0  q_2^0  ...  q_n^0] random orthonormaal
3:     for k = 0, 1, 2, ... do
4:         P̂_k = AQ̂_{k-1}
5:         Q̂_k R̂_k = P̂_k door QR-factorisatie
```

Convergentie: lineair, afhankelijk van afstand tussen eigenwaarden.

2.4.3 QR-algoritme zonder shift

Algorithm 5 QR-algoritme zonder shift

```
1: procedure EigenQR
2:     for k = 1, 2, 3... do
3:         A_k = Q_k R_k door QR-factorisatie
4:         A_{k+1} = R_k Q_k
```

$\tilde{Q}_k = \begin{bmatrix} \tilde{q}_1^{(k)} & \tilde{q}_2^{(k)} & ... & \tilde{q}_i^{(k)} \end{bmatrix}$ convergeert naar de eigenvctoren van A.

Complexiteit: $\mathcal{O}(km^3)$

2.4.4 Omvorming tot Hessenbergmatrix

Het aantal stappen in het QR-algoritme kan teruggebracht worden door eerst de matrix om te vormen naar een *Hessenbergvorm* m.b.v. Givens-rotaties.

Complexiteit van omvorming: $\mathcal{O}(m^3)$

2.4.5 QR-algoritme met shift

De convergentie van het QR-algoritme kan versneld worden door een *shift* toe te passen.

Algorithm 6 QR-algoritme met shift

```
procedure EigenQRShift
    A = A_0 Hessenberg
    for k = 1, 2, 3... do
        κ = a_{m,m}^{(k)}
        A_k − κ𝟙 = Q_k R_k door QR-factorisatie
        A_{k+1} = R_k Q_k + κ𝟙
```

$\tilde{Q}_k = \begin{bmatrix} \tilde{q}_1^{(k)} & \tilde{q}_2^{(k)} & ... & \tilde{q}_i^{(k)} \end{bmatrix}$ convergeert naar de eigenvctoren van A.

Complexiteit: $\mathcal{O}(km^3)$
Convergentie: kubisch indien symmetrisch, anders lineair, afhankelijk van afstand tussen eigenwaarden.

2.5 Toepassingen in de grafentheorie

2.5.1 PageRank

2.5.2 Meest centrale knoop