Modellering en Simulatie

Oefenzitting 1: QR-ontbinding

Academiejaar 2015–2016

1 Algoritmen voor de QR-ontbinding

De QR-ontbinding van een matrix $A \in \mathbb{R}^{m\times n}$ wordt gegeven door

$$A = QR,$$

waarbij $Q \in \mathbb{R}^{m\times m}$ een orthogonale matrix is en $R \in \mathbb{R}^{m\times n}$ een bovendriehoeksmatrix.

1.1 Givens

Opgave 1. Implementeer een algoritme voor het berekenen van een QR-ontbinding door middel van Givensrotaties. Gebruik daarvoor de functie `givens` in Matlab. Voorspel en bekijk de vorm van R na elke stap; waar verschijnen de nullen? Welke waarden veranderen?

Opgave 2. Hoe ga je de correctheid van je algoritme na?

1.2 Facultatief: Gram en Schmidt

Zij $A \in \mathbb{R}^{m\times n}$ met $m \geq n$. Het Gram–Schmidt orthogonalisatieproces kan eenvoudig verklaard worden met behulp van orthogonale projectoren. Herinner dat het doel eruit bestaat om een orthogonale basis te vinden voor de vectorruimte opgespannen door de eerste k kolommen van de matrix A. Hiervoor gaan we iteratief te werk. Het geval $k = 1$ is triviaal: we stellen $\mathbf{a}_1 = r_{1,1}\mathbf{q}_1$ voor een vector $\mathbf{q}_1$ van lengte 1. Het volgt onmiddellijk dat $r_{1,1} = \|\mathbf{a}_1\|$. Voor $k = 2$ proberen we te schrijven:

$$\mathbf{a}_2 = r_{1,2}\mathbf{q}_1 + r_{2,2}\mathbf{q}_2,$$

waarbij we opleggen dat $\mathbf{q}_1^T\mathbf{q}_2 := 0$, en waarin $r_{1,2}$ en $r_{2,2}$ onbekende scalaire waarden zijn en $\mathbf{q}_2$ een nog onbekende vector van lengte 1 is. We vinden de onbekenden dan als volgt. De projector $\mathbf{q}_1\mathbf{q}_1^T$ beschouwende, leert ons onmiddellijk dat

$$\mathbf{q}_1(\mathbf{q}_1^T\mathbf{a}_2) = r_{1,2}\mathbf{q}_1\underbrace{\mathbf{q}_1^T\mathbf{q}_1}_{1} + r_{2,2}\mathbf{q}_1\underbrace{\mathbf{q}_1^T\mathbf{q}_2}_{0} = r_{1,2}\mathbf{q}_1,$$

zodat $r_{1,2} = \mathbf{q}_1^T\mathbf{a}_2$. Dan vinden we eenvoudig

$$r_{2,2}\mathbf{q}_2 = \mathbf{a}_2 - r_{1,2}\mathbf{q}_1 = \mathbf{a}_2 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_2;$$

dit wil zeggen dat de gezochte $\mathbf{q}_2$ een gescaleerde versie is van de vector die overblijft na projectie van $\mathbf{a}_2$ op het orthogonale complement van de ruimte opgespannen door $\mathbf{q}_1$. Voor $k = 3$ zoeken we een manier om te schrijven

$$\mathbf{a}_3 = r_{1,3}\mathbf{q}_1 + r_{2,3}\mathbf{q}_2 + r_{3,3}\mathbf{q}_3,$$

*Begeleider: Nick Vannieuwenhoven (`nick.vannieuwenhoven@cs.kuleuven.be`).

†Opgesteld door: L. Vanherpe (2005–2010), N. Achtsis (2010–2014) en N. Vannieuwenhoven (2014–2016)

waarin we eisen dat $\mathbf{q}_1^T\mathbf{q}_2 = \mathbf{q}_1^T\mathbf{q}_3 = \mathbf{q}_2^T\mathbf{q}_3 = 0$ en dat $\mathbf{q}_3$ van lengte 1 is: $\mathbf{q}_3^T\mathbf{q}_3 = 1$. Merk op dat we al onze beperkingen inzake de vectoren $\mathbf{q}_i$, $i = 1, 2, 3$, beknopt kunnen neerschrijven in matrixvorm als volgt:

$$Q_3^T Q_3 = I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{waarin} \quad Q_3 = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \end{bmatrix}.$$

Om onze onbekenden te bepalen, gaan we als volgt te werk. Wanneer we projecteren volgens, respectievelijk, $\mathbf{q}_1\mathbf{q}_1^T$ en $\mathbf{q}_2\mathbf{q}_2^T$, dan vinden we, respectievelijk,

$$\mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_3 = r_{1,3}\mathbf{q}_1\mathbf{q}_1^T\mathbf{q}_1 + r_{2,3}\mathbf{q}_1\mathbf{q}_1^T\mathbf{q}_2 + r_{3,3}\mathbf{q}_1\mathbf{q}_1^T\mathbf{q}_3 = r_{1,3}\mathbf{q}_1 \quad \text{en} \quad \mathbf{q}_2\mathbf{q}_2^T\mathbf{a}_3 = r_{2,3}\mathbf{q}_2$$

waaruit onmiddellijk volgt dat $r_{1,3} = \mathbf{q}_1^T\mathbf{a}_3$ en $r_{2,3} = \mathbf{q}_2^T\mathbf{a}_3$. Dan kunnen we de overgebleven onbekenden $r_{3,3}$ en $\mathbf{q}_3$ eenvoudig vinden uit $r_{3,3}\mathbf{q}_3 = \mathbf{a}_3 - r_{1,3}\mathbf{q}_1 - r_{2,3}\mathbf{q}_2$. In het algemeen willen we $\mathbf{a}_k$ schrijven als

$$\mathbf{a}_k = r_{1,k}\mathbf{q}_1 + r_{2,k}\mathbf{q}_2 + \cdots + r_{k,k}\mathbf{q}_k$$

waarin

$$Q_k^T Q_k = I_k \qquad \text{met} \quad Q_k = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_k \end{bmatrix}$$

en I_k de eenheidsmatrix van orde k. Men gaat eenvoudig na dat

$$\mathbf{q}_i\mathbf{q}_i^T\mathbf{a}_k = r_{i,k}\mathbf{q}_i \quad \text{waaruit volgt dat} \quad r_{i,k} = \mathbf{q}_i^T\mathbf{a}_k$$

voor alle $i = 1, 2, \ldots, k-1$. Merk op dat we deze laatste betrekking in matrixvorm kunnen noteren als

$$\mathbf{s}_i := \begin{bmatrix} r_{1,k} \\ r_{2,k} \\ \vdots \\ r_{k-1,k} \end{bmatrix} = Q_{k-1}^T\mathbf{a}_k.$$

De overgebleven onbekenden vindt men dan uit:

$$\begin{aligned} r_{k,k}\mathbf{q}_k &= \mathbf{a}_k - \mathbf{q}_1 r_{1,k} - \mathbf{q}_2 r_{2,k} - \cdots - \mathbf{q}_{k-1} r_{k-1,k} \\ &= \mathbf{a}_k - \mathbf{q}_1(\mathbf{e}_1^T Q_{k-1}^T\mathbf{a}_k) - \mathbf{q}_2(\mathbf{e}_2^T Q_{k-1}^T\mathbf{a}_k) - \cdots - \mathbf{q}_{k-1}(\mathbf{e}_{k-1}^T Q_{k-1}^T\mathbf{a}_k) \\ &= \mathbf{a}_k - (\mathbf{q}_1\mathbf{e}_1^T)(Q_{k-1}^T\mathbf{a}_k) - (\mathbf{q}_2\mathbf{e}_2^T)(Q_{k-1}^T\mathbf{a}_k) - \cdots - (\mathbf{q}_{k-1}\mathbf{e}_{k-1}^T)(Q_{k-1}^T\mathbf{a}_k) \\ &= \mathbf{a}_k - \left((\mathbf{q}_1\mathbf{e}_1^T) + (\mathbf{q}_2\mathbf{e}_2^T) + \cdots + (\mathbf{q}_{k-1}\mathbf{e}_{k-1}^T)\right)(Q_{k-1}^T\mathbf{a}_k) \\ &= \mathbf{a}_k - Q_{k-1}Q_{k-1}^T\mathbf{a}_k, \end{aligned}$$

waarin $\mathbf{e}_j$ de j^{de} standaardbasisvector is. Zo komen we tot een matrixgebaseerde implementatie van het Gram–Schmidt orthogonalisatieproces dat beschreven staat in de cursus; het wordt weergegeven in Algoritme 1. Dit proces resulteert in de ontbinding $A = QR$, waarbij $Q \in \mathbb{R}^{m\times n}$ orthogonaal is en $R \in \mathbb{R}^{n\times n}$ een bovendriehoeksmatrix is.

Algoritme 1: Gram–Schmidt orthogonalisatie

$\mathbf{q}_1 \leftarrow \mathbf{a}_1/\|\mathbf{a}_1\|$;
$\mathbf{r}_1 \leftarrow \begin{bmatrix} \|\mathbf{a}_1\| & 0 \end{bmatrix}^T$;
for $i \leftarrow 2$ *to* n **do**
  $\mathbf{r}_i \leftarrow 0$;
  $\mathbf{s}_i \leftarrow Q_{i-1}^T\mathbf{a}_i$;
  $\mathbf{a}_i \leftarrow \mathbf{a}_i - Q_{i-1}\mathbf{s}_i$;
  $\mathbf{r}_i \leftarrow \mathbf{r}_i^T + \begin{bmatrix} \mathbf{s}_i^T & \|\mathbf{a}_i\| & 0 \end{bmatrix}^T$;
  $\mathbf{q}_i \leftarrow \mathbf{a}_i/\|\mathbf{a}_i\|$;
end

Opgave 3. Implementeer bovenstaande versie van het Gram–Schmidt orthogonalisatieproces.

Opgave 4. Genereer een willekeurige 500×500 matrix en bereken diens QR-ontbinding met het algoritme uit Opgave 1, Opgave 3 en met de Matlabfunctie `qr`. Wat zijn de uitvoeringstijden voor deze algoritmen? Hoeveel bedraagt de fout $\|Q^TQ - I\|_F$ en $\|QR - A\|_F$ voor de drie algoritmen?

Een triviale aanpassing van het Gram–Schmidt orthogonalisatieproces zorgt voor numerieke stabiliteit. De idee bestaat eruit om de geprojecteerde vector $\mathbf{a}_i' := (I - Q_{i-1}Q_{i-1}^T)\mathbf{a}_i$ iteratief te projecteren op het orthogonale complement van Q_{i-1}. We berekenen dus in de tweede stap $\mathbf{a}_i'' = (I - Q_{i-1}Q_{i-1}^T)\mathbf{a}_i'$, in de derde stap $\mathbf{a}_i''' = (I - Q_{i-1}Q_{i-1}^T)\mathbf{a}_i''$ enzoverder. Het volstaat om één extra lus te plaatsen rond stappen 5 tot 7 in Algoritme 1. Deze lus wordt dan een opgegeven aantal keer uitgevoerd. Men noemt dit het geïtereerde Gram–Schmidt orthogonalisatieproces.

Opgave 5. Beschouw opnieuw een willekeurige 500×500 matrix en bereken diens QR-ontbinding met het geïtereerde Gram–Schmidt algoritme, waarbij je $k = 1, 2, \ldots, 5$ keer itereert in de binnenste lus. Voor elk van deze waarden van k beschouw je de fout $\|Q^TQ - I\|_F$ en $\|QR - A\|_F$. Wat merk je?

2 Een toepassing: Information retrieval

Beschouw de volgende eenvoudige gegevensbank met vijf documenten dewelke beschreven worden door zes termen. De $t = 6$ termen zijn:

T1:	bake
T2:	recipe
T3:	bread
T4:	cake
T5:	pastry
T6:	pie

De $d = 5$ documenttitels zijn:

D1:	How to <u>Bake</u> <u>Bread</u> without <u>Recipes</u>
D2:	The Classic Art of Viennese <u>Pastry</u>
D3:	Numerical <u>Recipes</u>: The Art of Scientific Computing
D4:	<u>Breads</u>, <u>Pastries</u>, <u>Pies</u> and <u>Cakes</u>: Quantity <u>Baking</u> <u>Recipes</u>
D5:	<u>Pastry</u>: A Book of Best French <u>Recipes</u>

waarbij de woorden die overeenstemmen met termen in onze gegevensbank onderlijnd werden. Merk op dat we hierbij steeds het enkelvoud en de stam van het woord gebruiken om te bepalen of het woord in de verzameling van termen voorkomt. Zo komen "bake," "baking," "baked" en "bakes" allen overeen met de term "bake." We kunnen deze gegevensbank op natuurlijke wijze voorstellen door een vectorruimtemodel. In dit model wordt elk document voorgesteld door een *documentvector*; deze vector bevat de waarde 1 op positie i wanneer term Ti uit de gegevensbank voorkomt in het document. De gegevensbank bestaande uit alle documenten kan dan beschreven worden door de 6×5 term-documentmatrix $\hat{A}$, waarbij $\hat{a}_{i,j}$ het aantal keer dat term Ti optreedt in document Dj registreert. De kolommen van deze matrix zijn de documentvectoren. De matrix wordt dus als volgt gegeven:

$$\hat{A} = \begin{array}{r} \\ \text{bake} \\ \text{recipe} \\ \text{bread} \\ \text{cake} \\ \text{pastry} \\ \text{pie} \end{array} \begin{array}{c} \begin{array}{ccccc} D1 & D2 & D3 & D4 & D5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Opgave 6. Pas de elementen van de matrix $\hat{A}$ aan zodat de Euclidische norm van elke kolom gelijk is aan 1 en noem deze nieuwe matrix A. Kun je een oplossing bedenken die gebruik maakt van `arrayfun`?

We kunnen het bovenstaande vectorruimtemodel gebruiken om relevante documenten in onze gegevensbank te zoeken. Hiervoor moet eerst elke zoekopdracht omgezet worden in een *zoekvector*. Deze vector bevat een “1” op elke positie die overeenstemt met een zoekterm in de zoekopdracht. De overige waarden van de zoekvector zijn gelijk aan nul. De zoekopdracht “cake pie” wordt bijvoorbeeld omgezet in de vector

$$\mathbf{q} = \begin{array}{r} \text{bake} \\ \text{recipe} \\ \text{bread} \\ \text{cake} \\ \text{pastry} \\ \text{pie} \end{array} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

In de gegevensbank wordt daarna nagegaan welke documenten het meest overeenkomen met deze zoekopdracht. We kunnen verschillende definities hanteren om de afstand tussen een zoekvector en een documentvector te bepalen. We zouden bijvoorbeeld de norm van het verschil kunnen gebruiken. In deze opgave zullen we echter de overeenkomst tussen twee vectoren meten aan de hand van de principale hoek tussen deze vectoren. De principale hoek tussen de zoekvector $\mathbf{q}$ en de j^{de} documentvector $\mathbf{a}_j$ wordt gegeven door:

$$\theta_j = \operatorname{acos}\left(\frac{|\mathbf{q}^T\mathbf{a}_j|}{\|\mathbf{a}_j\|\|\mathbf{q}\|}\right). \tag{1}$$

Een document wordt als relevant beoordeeld wanneer de principale hoek θ_j “voldoende klein” is; in deze sectie gebruiken we de drempelwaarde $\frac{\pi}{4}$ radialen, oftewel 45 graden.

Opgave 7. Implementeer een zoekfunctie op basis van formule (1) die alle relevante documenten oplevert. Hint: zoek uit wat de functie `subspace` doet.

Opgave 8. Stel nu dat een gebruiker op zoek is naar informatie over brood bakken: de zoekopdracht is “baking bread.” Hoe wordt deze opdracht omgezet in een zoekvector? Welke documenten worden als relevant weergegeven voor deze opdracht? Wat zijn de resultaten als we enkel “baking” als zoekterm invoeren? Wat stel je vast?

2.1 De QR-ontbinding

We merken op dat de term-documentmatrix $\hat{A}$ niet van volle rang is. De kolomruimte van een matrix $A \in \mathbb{R}^{m\times n}$ kan voorgesteld worden door een basis van r vectoren, waarbij $r \leq \min\{m, n\}$ de rang van A is. Met behulp van een QR-ontbinding met goed gekozen kolompivotering kunnen we dergelijke basis in de praktijk als volgt bepalen:

$$A' := AP = QR, \tag{2}$$

waarbij $R \in \mathbb{R}^{m\times n}$ een bovendriehoeksmatrix is, $Q \in \mathbb{R}^{m\times m}$ een orthogonale matrix en $P \in \mathbb{R}^{n\times n}$ een *permutatiematrix*. Een permutatiematrix is een orthogonale matrix wiens kolommen een permutatie vormen van de eenheidsmatrix van orde n. Indien de rang van A (alsook deze van A') gelijk is aan $r \leq \min\{m, n\}$, dan zal R een additionele structuur bezitten: de laatste $(m-r)$ rijen zullen enkel nullen bevatten. Wanneer $A \in \mathbb{R}^{m\times n}$ met $m \geq n$ en A van rang r, dan kunnen we de QR-ontbinding met geschikte kolompermutatie uit vergelijking (2) steeds schrijven als

$$A' = AP = QR = \begin{bmatrix} Q_r & Q_r^{\perp} \end{bmatrix} \begin{bmatrix} R_r \\ 0 \end{bmatrix} \tag{3}$$

waarbij $Q_r \in \mathbb{R}^{m\times r}$ de eerste r kolommen van Q bevat, $Q_r^{\perp}$ de laatste $m-r$ kolommen van Q bevat, $R_r \in \mathbb{R}^{r\times n}$ een bovendriehoeksmatrix die de eerste r rijen van R bevat en 0 de $(m-r)\times n$ nulmatrix. Uit bovenstaande uitdrukking volgt dat de kolomruimte van A' en A gegenereerd wordt door de kolomruimte van Q_r.

Opgave 9. Bewijs deze laatste bewering: het volstaat aan te tonen dat de j^{de} kolom $\mathbf{a}'_j$ van A' kan geschreven worden als een lineaire combinatie van de eerste r kolomvectoren uit Q. Hint: rechts vermenigvuldigen met de j^{de} standaardbasisvector $\mathbf{e}_j \in \mathbb{R}^n$ selecteert de j^{de} kolom uit een matrix $X \in \mathbb{R}^{m\times n}$.

Opgave 10. Bereken handmatig de rang van de genormaliseerde term-documentmatrix A uit Opgave 6.

Opgave 11. Bereken in Matlab een basis van de kolomruimte van A aan de hand van (3). Hint: inspecteer `help qr`.

2.2 Lagerangbenaderingen

Wanneer het QR-ontbindingsalgoritme met kolompivotering uit de cursus wordt toegepast op een willekeurige matrix $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ om een QR-ontbinding $AP = QR$ te berekenen, heeft het de neiging om grotere waarden in de richting van de linkerbovenhoek van de bovendriehoeksmatrix R te duwen en kleinere waarden in de richting van de rechteronderhoek. We kunnen dit neerschrijven als

$$A' = AP = QR = \begin{bmatrix} Q_k & Q_k^\perp \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} \\ 0 & R_{2,2} \end{bmatrix}, \tag{4}$$

waarbij $Q_k \in \mathbb{R}^{m\times k}$ de eerste $k \leq n$ kolomvectoren van Q bevat, $Q_k^\perp$ de laatste $(m-k)$ kolommen van Q bevat, $R_{1,1} \in \mathbb{R}^{k\times k}$ en $R_{2,2} \in \mathbb{R}^{(m-k)\times(n-k)}$ bovendriehoeksmatrices en $R_{1,2} \in \mathbb{R}^{k\times(n-k)}$. Een lagerangbenadering van A kan eenvoudig via een afgeknotte QR-ontbinding opgesteld worden. Hiertoe volstaat het om in te zien dat wanneer we $R_{2,2}$ gelijkstellen aan de nulmatrix we opnieuw de ontbinding in vergelijking (3) bekomen. Een rang-k benadering van A kunnen we dus bekomen als volgt:

$$A' = AP = QR = \begin{bmatrix} Q_k & Q_k^\perp \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} \\ 0 & R_{2,2} \end{bmatrix} \approx \begin{bmatrix} Q_k & Q_k^\perp \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} \\ 0 & 0 \end{bmatrix} = Q_k \begin{bmatrix} R_{1,1} & R_{1,2} \end{bmatrix} = Q_k R_k$$

met alle matrices zoals in vergelijking (4) en $R_k = \begin{bmatrix} R_{1,1} & R_{1,2} \end{bmatrix}$.

Opgave 12. Definieer nu $E = QR - Q_kR_k$. Stel een algemene uitdrukking op voor E en $\|E\|_F$ in functie van Q_k, $Q_k^\perp$, $R_{1,1}$, $R_{1,2}$ en $R_{2,1}$.

De QR-ontbinding kan ons helpen om met onzekerheden in de gegevensbank te werken. In dit voorbeeld zouden we kunnen zeggen dat het vijfde document relevant is voor bakken omdat het over "pastry recipes" gaat en dat zijn eigenlijk instructies om "pastries" te bakken. In de niet-genormaliseerde matrix $\hat{A}$ zou dan het element $\hat{A}_{15} = 1$ opgenomen moeten worden. A wordt dus misschien beter voorgesteld door $\widehat{A} = A + E$, waarbij de onzekerheidsmatrix E ontbrekende informatie bevat. We proberen vervolgens onze gegevensbank voor te stellen door middel van lagerangbenaderingen, dewelke ons enkele van zulke geperturbeerde matrices $\widehat{A}$ zal opleveren.

Opgave 13. Zij gegeven een matrix $A \in \mathbb{R}^{n\times n}$ en zijn rang-k lagerangbenadering $AP \approx Q_kR_k$. Wat is de geheugencomplexiteit voor het opslaan van A? Wat is de geheugencomplexiteit voor het opslaan van de lagerangbenadering van A? Voor welke waarden van k is het efficiënter om de lagerangbenadering op te slaan?

Voor de volgende opgaven beschouwen we opnieuw de genormaliseerde term-documentmatrix uit Opgave 6. Als norm gebruiken we steeds de Frobeniusnorm.

Opgave 14. Beschouw $k = 1, 2, \ldots, 5$ en partitioneer de QR-ontbinding met kolompivotering zoals in vergelijking (4). Hoe groot is R_{22} relatief ten opzichte van R voor de verschillende waarden van k?

Opgave 15. Implementeer de lagerangbenadering van A en voer opnieuw de zoekopdrachten "baking bread" en "baking". Wat stel je vast?