Technische Universität München
Institut für Informatik
Prof. Dr. Thomas Huckle
Michael Obersteiner, Michael Rippl

SoSe 2017

Numerisches Programmieren

2. Programmieraufgabe: Lineare Gleichungssysteme & PageRank

Einführung

Für Suchmaschinen ist es interessant, die Bedeutung einer Website zu kennen. Das von Google verwendete PageRank-Verfahren betrachtet dazu die Links zwischen den Seiten. Das Grundprinzip lautet: Je mehr Links auf eine Seite verweisen, umso bedeutender ist diese Seite. Je bedeutender die verweisenden Seiten sind, desto größer ist der Effekt.
Dieses Prinzip führt zu einem System linearer Gleichungen, das in dieser Aufgabe gelöst werden soll.

PageRank

Wie in der Einführung erwähnt soll der Rang einer Seite von der Anzahl der eingehenden Links und dem Rang der verlinkenden Seiten abhängen. Wir betrachten nun jemanden, der durch das Web surft, indem er zufällig Links folgt. Dieser Zufallssurfer wird sich desto häufiger auf einer bestimmten Webseite aufhalten, umso mehr Links auf sie verweisen, und umso bedeutender die vorherige Seite ist. Anhand der Aufenthaltswahrscheinlichkeit (=Rang $\bar{p}_j$) eines Zufallssurfers auf einer Seite j kann nun eine Rangliste der Seiten aufgestellt werden.

Um das zufällige Surfen auf insgesamt n Seiten zu simulieren, stellen wir zunächst eine Matrix $A \in \mathbb{R}^{n \times n}$ auf, in der das Element a_{ij} die Wahrscheinlichkeit angibt, von Seite j auf Seite i zu gelangen. Da es sich um Wahrscheinlichkeiten handelt, muss für **die Spaltensummen** $\sum_{i=1}^{n} a_{ij} = 1$ gelten. Außerdem soll jedem Link mit gleicher Wahrscheinlichkeit gefolgt werden, es gilt also

$$a_{ij} = \begin{cases} \frac{1}{\#\text{Links die von } j \text{ ausgehen}} & \text{falls } j \text{ auf } i \text{ linkt} \\ 0 & \text{sonst.} \end{cases}$$

Wir nehmen an, dass jede Seite auf sich selbst verweist; dies verhindert insbesondere eine Division durch 0.

Im bisherigen Modell könnte es passieren, dass einige Seiten eine „Insel“ bilden und nur untereinander, aber nicht zum Rest des Internets verlinken. Um zu verhindern, dass der Surfer auf einer solchen Insel gefangen bleibt, lassen wir ihn mit einer Wahrscheinlichkeit $\rho \in [0, 1]$ irgendeine Seite (z.B. aus seinen Bookmarks) besuchen, anstatt einem Link zu folgen. Dadurch ergibt sich eine modifizierte Wahrscheinlichkeits-Matrix $\tilde{A}$ mit

$$\tilde{a}_{ij} = (1 - \rho) * a_{ij} + \rho/n.$$

Der Aufenthaltsort des Surfers ist zunächst stark von der Ausgangsseite abhängig. Wir hoffen aber, dass sich die Aufenthaltswahrscheinlichkeiten nach hinreichend langer Zeit auf ein Gleichgewicht einpendeln, das von der Ausgangsseite unabhängig ist. [1] Falls wir die Wahrscheinlichkeiten p_j, sich auf der Seite j zu befinden, als Vektor p notieren, so sind die Wahrscheinlichkeiten nach einem weiteren Schritt zufälligen Surfens $\tilde{A}p$. Das gesuchte Gleichgewicht $\bar{p}$ verändert sich nicht mehr nach einem Surf-Schritt, also erfüllt es $\tilde{A}\bar{p} = \bar{p}$.

Gesucht ist also ein Eigenvektor $\bar{p} \neq 0$ von $\tilde{A}$ zum Eigenwert 1.[2] Es gilt $\tilde{A}\bar{p} = \bar{p} \Leftrightarrow \tilde{A}\bar{p} - I\bar{p} = 0 \Leftrightarrow (\tilde{A} - I)\bar{p} = 0$, d.h. $\bar{p}$ ist eine Lösung des linearen Gleichungssystems

$$(\tilde{A} - I)p = 0. \tag{1}$$

Da jedoch auch jedes Vielfache $\lambda\bar{p}$ ein Eigenvektor ist, ist die Lösung nicht eindeutig, d.h. $(\tilde{A} - I)$ ist nicht invertierbar und der normale Gauß-Algorithmus kann keine Lösung finden. Ein mögliches Lösungsverfahren eines solchen singulären Systems ist eine modifizierte Version des Gauß-Algorithmus, welcher im folgenden Abschnitt vorgestellt wird.

Sei $p \neq 0$ nun eine Lösung des Systems (1). Durch Normierung von p erhält man die gesuchten Aufenthaltswahrscheinlichkeiten[3]

$$\bar{p} = \lambda p \text{ mit } \lambda = 1/\sum_{j=1}^{n} p_j \quad (\bar{p} \text{ erfüllt nun } \sum_{j=1}^{n} \bar{p}_j = 1).$$

Basierend auf diesen Werten findet das PageRank-Verfahren eine sinnvolle Sortierung der Webbseiten.

---

[1] Bei dem Modell handelt es sich um eine Markov-Kette. Das gesuchte Gleichgewicht ist eine stationäre Verteilung.

[2] 1 ist immer Eigenwert von $\tilde{A}$, da per Konstruktion von $\tilde{A}$ (Spaltensumme 1) $(1, \cdots, 1)$ EV von $\tilde{A}^T$ zum EW 1 ist und $\tilde{A}$ und $\tilde{A}^T$ dasselbe char. Polynom haben.

[3] Die von $\tilde{A}$ beschriebene Markov-Kette ist ergodisch. Die stationäre Verteilung $\bar{p}$ ist daher eindeutig.

Verfahren zur Lösung von Linearen Gleichungssystemen

In diesem Abschnitt werden nun einige Löser linearer Gleichungssysteme vorgestellt: die Rückwärtssubstitution für obere Dreiecksmatrizen, die Gauß-Elimination für allgemeine invertierbare Systeme und ein modifizierter Gauß-Algorithmus für singuläre Systeme.

Rückwärtssubstitution

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & \cdots & a_{2n} \\ & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \tag{2}$$

Ein Gleichungssystem der Form (2) mit einer oberen Dreiecksmatrix lässt sich leicht zeilenweise von unten nach oben lösen. Beim Lösen der i-ten Zeile sind $x_{i+1}, \ldots, x_n$ bereits bekannt, wodurch die einzige verbleibende Unbekannte x_i sofort bestimmt werden kann.

Gauß-Elimination mit Spalten-Pivotsuche

Die Gauß-Elimination ist ein Verfahren um lineare Gleichungssysteme in die leicht lösbare obere Dreiecksform zu bringen. Im k-ten Schritt der Gauß-Elimination müssen alle Elemente der k-ten Spalte unterhalb der Hauptdiagonalen zu Null werden. Dazu wird von jeder Zeile $i > k$ die mit $\frac{a_{ik}}{a_{kk}}$ multiplizierte k-te Zeile abgezogen. Dies ist nur möglich, falls $a_{kk} \neq 0$. Zudem sollte $\left|\frac{1}{a_{kk}}\right|$ möglichst klein sein, um Rechenfehler nicht zu verstärken. Die Spalten-Pivotsuche ermittelt daher vor jedem Eliminationsschritt die Zeile $j \geq k$, die das betragsgrößte Element $a_{jk}, j \geq k$ enthält (bei Gleichstand die Zeile mit kleinstem Index). Falls $k \neq j$ werden die Zeilen k und j vertauscht.

Im folgenden Beispiel ist die erste Spalte schon fertig bearbeitet, d.h. es muss nun auf der zweiten Spalte die Pivot-Suche durchgeführt werden. Von den zu untersuchenden Matrixelementen ist das in der dritten Zeile das betragsmäßig größte, die zweite Zeile wird daher mit der dritten Zeile vertauscht:

$$\left(\begin{array}{cccc|c} 1 & 4 & 8 & 3 & 7 \\ 0 & 2 & 2 & 4 & 0 \\ 0 & -3 & -7 & 2 & 1 \\ 0 & 1 & 5 & 2 & 2 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 4 & 8 & 3 & 7 \\ 0 & -3 & -7 & 2 & 1 \\ 0 & 2 & 2 & 4 & 0 \\ 0 & 1 & 5 & 2 & 2 \end{array}\right)$$

Modifizierter Gauß-Algorithmus

Gegeben sei nun das System

$$Ap = 0 \tag{3}$$

mit einer singulären Matrix A. Wir verwenden eine Abwandlung des Gauß-Algorithmus mit Spalten-Pivotsuche, um ein $p \neq 0$ zu bestimmen, das (3) löst. Dazu werden zunächst die üblichen Schritte der Gauß-Elimination ausgeführt. Aufgrund der Nicht-Invertierbarkeit der Matrix A kann dieses Vorgehen nicht bis zum Ende fortgesetzt werden. Die Gauß-Schritte enden, wenn kein Pivotelement gefunden werden kann, d.h. wenn alle in Frage kommenden Elemente 0 sind (in unserem Fall $< 1E-10$). Das umgeformte LGS hat zu diesem Zeitpunkt also folgende Struktur:

$$\underbrace{\left(\begin{array}{ccc|c} & \mathrm{T} & & v & * \\ & & & | & \\ 0 & \cdots & 0 & 0 & \\ \hline & & & 0 & \\ & 0 & & \vdots & * \\ & & & 0 & \end{array}\right)}_{:=\widehat{A}} \left(\begin{array}{c} | \\ p \\ | \end{array}\right) = \left(\begin{array}{c} 0 \\ \vdots \\ 0 \end{array}\right), \tag{4}$$

wobei T eine invertierbare obere Dreiecksmatrix ist. Löst man $Tx = -v$ durch Rückwärtssubstitution nach x, erhält man mit

$$p := (x, 1, 0 \ldots 0)^T \,[4]$$

eine Lösung von (4), da gilt:

$$\begin{aligned} \widehat{A}p &= \begin{pmatrix} T & v & * \\ 0 & 0 & * \end{pmatrix} \begin{pmatrix} x \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} T \\ 0 \end{pmatrix} x + \begin{pmatrix} v \\ 0 \end{pmatrix} 1 + \begin{pmatrix} * \\ * \end{pmatrix} 0 \\ &= \begin{pmatrix} -v \\ 0 \end{pmatrix} + \begin{pmatrix} v \\ 0 \end{pmatrix} = 0. \end{aligned}$$

[4] x ist hier der obere Teil des Vektors p

Beschreibung des Programmgerüsts

Das auf der Vorlesungsseite zur Verfügung gestellte Programmgerüst enthält:

- `Gauss`:
  Diese Klasse soll um Funktionen zur Lösung von linearen Gleichungssystemen erweitert werden. Zum einfachen Testen dieser Funktionen enthält sie bereits eine Methode zur Multiplikation einer Matrix mit einem Vektor (`matrixVectorMult()`).
- `PageRank`:
  Die Klasse PageRank bietet Methoden zur Generierung der Matrix der Übergangswahrscheinlichkeiten $\tilde{A}$, zur Ermittlung des Gleichgewichts der Aufenthaltswahrscheinlichkeiten $\bar{p}$ und der daraus resultierenden Reihenfolge der Webseiten. In Form einer Link-Matrix L erhalten diese Methoden die Informationen zu den Links zwischen den Webseiten.
- `LinkMatrix`:
  Eine Link-Matrix beschreibt die Link-Verbindungen zwischen den Webseiten, d.h. $L_{ij} = \begin{cases} 1 & \text{falls } j \text{ auf } i \text{ linkt} \\ 0 & \text{sonst} \end{cases}$
- `webseiten`:
  Dieser Ordner enthält Beispiele für Link-Matrizen und Webseiten
- `Crawler`:
  Der Webcrawler durchforstet, ausgehend von einer Startadresse, das Internet über Hyperlinks und ermöglicht das Speichern von Link-Informationen in Form einer Link-Matrix.
- `GUI`:
  Mit der GUI lassen sich Link-Matrizen erstellen. Von einer gegebenen Menge an Addressen aus startend besuchen Crawler verlinkte Webseiten und speichern die gesammelten Informationen in einer Ausgabedatei.
- `Util`:
  Diese Klasse bietet Test- und Textausgabemethoden.

Aufgaben

Im Folgenden werden die zu implementierenden Methoden aufgelistet. Details finden Sie jeweils in den Kommentaren zu den einzelnen Methoden. Überlegen Sie sich zu allen Methoden Testbeispiele und testen Sie unbedingt jede einzelne Methode direkt nach der Erstellung. Sie können zusätzlich die zur Verfügung gestellte Testklasse `Test` verwenden. Es ersetzt aber nicht das Verwenden eigener Testbeispiele.

- Programmieren Sie in der Klasse `Gauss` die gegebenen Methodenrümpfe `backSubst`, `solve` und `solveSing`.
- Programmieren Sie in der Klasse `PageRank` die gegebenen Methodenrümpfe `buildProbabilityMatrix` und `rank`.

Formalien und Hinweise

- Das Programmgerüst erhalten Sie auf den Webseiten zur Vorlesung.
- Ergänzen Sie das Programmgerüst bitte **nur an den dafür vorgegebenen Stellen!** Falls Sie die Struktur der Programme an anderer Stelle verändern, können wir sie evtl. nicht mehr testen.
- Beseitigen Sie vor Abgabe Ihres Programms alle Ausgaben an die Konsole und reichen Sie bis zum **09. Juni 2017 12:00 Uhr** über Moodle ein.
- Reichen Sie nur die Dateien `Gauss.java` und `PageRank.java` ein.
- Bitte laden Sie Ihre java-Dateien als flaches tgz-Archiv hoch. Der Dateiname ist beliebig wählbar, bei der Erweiterung muss es sich jedoch um .tgz oder .tar.gz handeln.

  Ein solches Archiv können Sie beispielsweise mit dem Linux-Tool tar erstellen, indem Sie die laut Aufgabenstellung zu bearbeitenden java-Dateien in ein sonst leeres Verzeichnis legen und dort anschließend den Befehl

  ```
  > tar cvvzf numpro_aufg2.tgz *.java
  ```

  ausführen.
- Wir empfehlen Ihnen, die Programme unter Linux (Rechnerhalle) zu testen.

Bitte beachten Sie: *Alle Abgaben, die nicht den formalen Kriterien genügen, werden grundsätzlich mit 0 Punkten bewertet!*