Technische Universität München
Institut für Informatik
Prof. Dr. Thomas Huckle
Michael Lieb, M. Sc.
Dipl.-Tech. Math. Stefanie Schraufstetter

SS 2010

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.

Rückwärtssubstitution

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

Ein Gleichungssystem der Form (1) 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 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 k-te Zeile multipliziert mit $\frac{a_{ik}}{a_{kk}}$ 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 Zeilevertauscht:

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

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, d.h umso bedeutender sie ist. Anhand der Aufenthaltswahrscheinlichkeit kann also 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 $\sum_{i=1}^{n} = 1$ 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 Division durch 0.

Der Aufenthaltsort des Surfers ist zunächst stark von der Ausgangsseite abhängig, wir hoffen aber, dass sie sich nach hinreichend langer Zeit die Aufenthaltswahrscheinlichkeiten 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 Ap. Für das Gleichgewicht $\bar{p}$, das wir suchen gilt $A\bar{p} = \bar{p}$, d.h. $\bar{p}$ ist Eigenvektor von A zum Eigenwert 1. [2]

Im bisherigen Modell könnte es passieren, dass einige Seiten eine „Insel“ bilden und nur untereinander, aber nicht zum Rest des Internets verlinken. Um zu

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

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

verhindern, dass der Surfer auf einer solchen Insel gefangen bleibt, lassen wir ihn mit einer Wahrscheinlichkeit ρ irgendeine Seite (z.B. aus seinen Bookmarks) besuchen, anstatt einem Link zu folgen. [3] Dadurch ergibt sich eine modifizierte Matrix $\tilde{A}$ mit

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

Modifizierter Gauß-Algorithmus

Gesucht ist also der Eigenvektor $\bar{p} \neq 0$ von $\tilde{A}$ zum Eigenwert 1. 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

$$(\tilde{A} - I)\bar{p} = 0 \tag{2}$$

Da jedoch auch jedes Vielfache $\lambda\bar{p}$ Eigenvektor ist, ist die Lösung nicht eindeutig, d.h. $(\tilde{A} - I)$ ist nicht invertierbar und der normale Gauß-Algorithmus kann keine Lösung finden.

Wir verwenden daher eine Abwandlung des Gauß-Algorithmus mit Spaltenpivotsuche um ein $\bar{p} \neq 0$ zu bestimmen, das (2) löst. Dazu werden zunächst solange wie möglich die üblichen Schritte des Gauß-Elimination ausgeführt. Die Elimination kann genau dann nicht mehr fortgesetzt werden, wenn kein Pivotelement gefunden werden kann, d.h. wenn alle in Frage kommenden Elemente 0 sind. Das umgeformte LGS hat zu diesem Zeitpunkt also folgende Struktur:

$$\underbrace{\left(\begin{array}{ccc|c|c} & \mathrm{T} & & | & \\ & & & v & * \\ & & & | & \\ 0 & \cdots & 0 & 0 & \\ \hline & & & 0 & \\ & 0 & & \vdots & * \\ & & & 0 & \end{array}\right)}_{:=\widehat{A}} \left(\begin{array}{c} | \\ p \\ | \end{array}\right) = \left(\begin{array}{c} 0 \\ \vdots \\ 0 \end{array}\right) \tag{3}$$

wobei T eine invertierbare obere Dreiecksmatrix ist. Löst man $Tx = -v$ durch Rückwärtssubstitution nach x erhält man für beliebige $\lambda \in \mathbb{R}$ mit

$$\lambda p := \lambda\,(x, 1, 0 \ldots 0)^T$$

Lösungen von (3) (und somit auch von (2)), da $\widehat{A}\lambda\,(x, 1, 0 \ldots 0)^T = \lambda(Tx, 0 \ldots 0)^T + \lambda(v * 1, 0 \ldots 0)^T = \lambda(-v + v, 0 \ldots 0)^T = 0$. Wählt man λ so, dass $\sum_{j=1}^{n} \lambda p_j = 1$ erhält man die gesuchten Aufenthaltswahrscheinlichkeiten $\bar{p} = \lambda p$.

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

Aufgaben

- Implementieren Sie in der Klasse `Gauss` die Methoden `backSubst`, `solve` und `solveSing`.
- Sie erhalten für Webseiten $1, \ldots, n$ eine Matrix L, die angibt welche Links zwischen diesen Seiten bestehen, d.h. $L_{ij} = \begin{cases} 1 & \text{falls } j \text{ auf } i \text{ linkt} \\ 0 & \text{sonst} \end{cases}$ und den Parameter $\rho \in [0, 1]$. Implementieren Sie die Methode `buildMatrix` der Klasse `PageRank`, die daraus die Matrix $\tilde{A}$ aufstellt.
- Implementieren Sie das Verfahren zur Bestimmung der Rangliste. Die Methode `rank` der Klasse `PageRank` erhält eine Matrix L (siehe voriger Punkt) und die zu den Indizes gehörenden URLs. Die Ausgabe soll eine Rangliste der URLs nach absteigender Aufenthaltswahrscheinlichkeit sein.

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 **04. Juni 2010, 12:00 Uhr** über das Web-Portal (siehe Homepage) ein.
- Wir empfehlen, die Programme unter Linux (Rechnerhalle) zu testen.