Abschlussprojekt

Ida Hönigmann Fabian Dopf

January 4, 2022

Aufgabe 1: Aufwandsordnung numerischer Verfahren

Wir betrachten ein abstraktes numerisches Verfahren, das für $N \in \mathbb{N}$ Eingabedaten eine Laufzeit von $y_N \in \mathbb{R}_+$ hat. Man sagt, das Verfahren habe Aufwandsordnung $p > 0$, falls eine Konstante $C > 0$ existiert, sodass $y_N \leq CN^p$ für alle $N \in \mathbb{N}$.

Teilaufgabe 1a:

Die Aufwandsordnung lässt sich über die Folge $\{p_N\}_{N\in\mathbb{N}}$ mit

$$p_N = \frac{\log(y_{2N}) - \log(y_n)}{\log(2)} \text{ für } N \in \mathbb{N} \tag{1}$$

quantifizieren. Beachten Sie, dass die Bestimmung von p_N die Verfügbarkeit von zwei aufeinanderfolgenden Folgengliedern y_N und y_{2N} erfordert. Verwenden Sie den Ansatz $y_N = CN^p$ und leiten Sie die Formel in 1 her!

Beweis. Annahme: $\forall N \in \mathbb{N}$ ist p_N, sodass $y_N \leq CN^{p_N}$ für ein $C > 0$.

Für ein beliebiges $N \in \mathbb{N}$ gilt $\exists C_{1N}, C_{2N} > 0$ und $p_{1N}, p_{2N} > 0$ mit $y_N \leq C_{1N}N^{p_{1N}}$ und $y_{2N} \leq C_{2N}(2N)^{p_{2N}}$.

Für $C := max\{C_{1N}, C_{2N}\}$ und $p_N := max\{p_{1N}, p_{2N}\}$ gilt $y_N \leq C_{1N}N^{p_{1N}} \leq CN^{p_N}$ und $y_{2N} \leq C_{2N}(2N)^{p_{2N}} \leq C(2N)^{p_N}$.

$$\log(y_{2N}) - \log(y_N) = \log\left(\frac{y_{2N}}{y_N}\right) = \log\left(\frac{C(2N)^{p_N}}{C \cdot N^{p_N}}\right) = \log(2^{p_N}) = p_N \log(2) \tag{2}$$

$$\implies p_N = \frac{\log(y_{2N}) - \log(y_N)}{\log(2)} \tag{3}$$

TODO ob das so alles passt...

□

Teilaufgabe 1b:

Sei $\{\delta_N\}_{N\in\mathbb{N}} \subseteq \mathbb{R}$ eine Nullfolge, d.h. es gilt $\delta_N \to 0$ für $N \to \infty$. Weiters verhalte sich die Laufzeit wie $y_N = (C + \delta_N)N^p$ mit $C > 0$. Zeigen Sie, dass die Folge $\{p_N\}_{N\in\mathbb{N}}$ gegen p konvergiert, d.h. es gilt $p_N \to p$ für $N \to \infty$.

Beweis. Zuerst berechnen wir einen Grenzwert, den wir in späterer Folge verwenden werden. Die Gleichungen stimmen, da lim stetig ist und da laut Voraussetzung δ_N und somit auch δ_{2N} als Teilfolge, gegen 0 konvergieren.

$$\lim_{n\to\infty} \log\left(\frac{C+\delta_{2N}}{C+\delta_N}\right) = \log\left(\lim_{n\to\infty}\frac{C+\delta_{2N}}{C+\delta_N}\right) = \log\left(\frac{\lim\limits_{n\to\infty} C+\delta_{2N}}{\lim\limits_{n\to\infty} C+\delta_N}\right) = \log\left(\frac{C}{C}\right) = \log(1) = 0 \quad (4)$$

Wir berechnen $\lim\limits_{n\to\infty} p_n$ indem wir die Gleichung 1 verwenden. Durch Einsetzen von $y_N = (C+\delta_N)N^p$ und den Rechenregeln von Limiten und dem Logarithmus erhalten wir folgendes:

$$\implies p_N = \frac{\log(y_{2N}) - \log(y_N)}{\log(2)} = \frac{\log((C+\delta_{2N})(2N)^p) - \log((C+\delta_N)N^p)}{\log(2)} = \frac{\log\left(\frac{(C+\delta_{2N})(2N)^p}{(C+\delta_N)N^p}\right)}{\log(2)} \quad (5)$$

$$= \frac{\log\left(\frac{(C+\delta_{2N})2^p}{(C+\delta_N)}\right)}{\log(2)} = \frac{p\log(2) + \log\left(\frac{C+\delta_{2N}}{C+\delta_N}\right)}{\log(2)} = p + \frac{\log\left(\frac{C+\delta_{2N}}{C+\delta_N}\right)}{\log(2)} \xrightarrow{n\to\infty} p + 0 = p \quad (6)$$

Zusammenfassend gilt nun $\lim\limits_{n\to\infty} p_n = p$, was zu zeigen war.

□

Teilaufgabe 1c:

In sogenannter doppelt logarithmischer Darstellung (log-log Plots) wird für beide Koordinatenachsen eine logarithmische Skalierung verwendet, d.h. sowohl die waagrechte als auch die senkrechte Koordinatenachse wird logarithmisch unterteilt. Wie werden Potenzfunktionen der Form $y = cx^p$ in einem log-log Plot dargestellt? Wie können Sie die Ordnung p und die Konstante $c > 0$ aus einem log-log Plot von $y = cx^p$ direkt auslesen?

Darstellung ist Gerade. $c = f(1)$ und p ist Steigung, wenn beide Achsen "gleich" skaliert.

Aufgabe 2: Cholesky-Verfahren und Skyline-Matrizen

Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt Skyline-Matrix, falls es für $l = 1, ..., n$ Zahlen $p_l, q_l \in \mathbb{N}_0$ gibt, sodass für die i-te Zeile und j-te Spalte von A gilt:

- $A_{i,k} = 0$ für $k < i - p_i$,
- $A_{k,j} = 0$ für $k < j - q_j$,

Folgendes Beispiel illustriert diese Aussage:

$$A = \begin{pmatrix} 1 & & & & 1 \\ & 1 & & 2 & 2 \\ & & 1 & 3 & 3 \\ 1 & 2 & 3 & 14 & 18 \\ & 4 & 5 & 29 & 48 \end{pmatrix}.$$

Teilaufgabe 2a:

Beweisen Sie, dass das Cholesky-Verfahren genau dann wohldefiniert ist (d.h. es wird nicht durch Null dividiert oder die Wurzel aus einer negativen Zahl gezogen), wenn die Matrix $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit ist.

Beweis. TODO Beweis 2a aufschreiben □

Teilaufgabe 2b:

Beweisen Sie, dass die Besetzungsstruktur der Cholesky-Zerlegung der Skyline-Matrix A erhalten bleibt, d.h. dass auch die untere Dreiecksmatrix L eine geeignete Bandstruktur aufweist.

Beweis. TODO Beweis 2b $\square$

Aufgabe 3: Pseudocode für Cholesky-Zerlegung von Skyline-Matrizen

Verwenden Sie den Cholesky-Algorithmus aus der Vorlesung. Entwerfen Sie jeweils einen Pseudocode, der für eine Skyline-Matrix:

0.1 Teilaufgabe 3a:

möglichst effizient die Struktur erkennt.
TODO Pseudocode

Teilaufgabe 3b:

die Cholesky-Zerlegung berechnet.
TODO Pseudocode aufschreiben

Aufgabe 4: Aufwand des Algorithmus und Verhalten in Spezialfällen

Teilaufgabe 4a:

Sei $A \in \mathbb{R}^{n \times n}$ eine Skyline-Matrix. Welchen Aufwand haben Ihre Algorithmen aus Aufgabe 3 in Abhängigkeit von der Größe n der Eingabedaten und Skyline-Indices $p_l = q_l$?
TODO Aufwand bestimmen

Teilaufgabe 4b:

Betrachten Sie Matrizen mit den Besetzungsstrukturen

$$\begin{pmatrix} * & * & * & * & * \\ * & * & & & \\ * & & * & & \\ * & & & * & \\ * & & & & * \end{pmatrix} \text{ bzw. } \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \\ * & * & * & * & * \end{pmatrix}$$

Welche Besetzungsstruktur hat die Cholesky-Zerlegung für beide Matrizen? Was könnte man machen, um für Matrizen mit der "linken" Besetzungsstruktur die Cholesky-Zerlegung effizienter zu berechnen?

linke Matrix ist quasi vollbesetzt als Skyline-Matrix (mit Nulleinträgen!)

rechte Matrix hat p=q=0 für alle außer letzte Zeile dort p=q=n

effizientere berechnung der cholesky-Zerlegung durch "spiegeln". Also aus $A \in \mathbb{R}^{n \times n}$ mache $B \in \mathbb{R}^{n \times n}$ durch $B_{i,j} = A_{(n+1-i),(n+1-j)}$. Umkehrabbildung ist gleich (auch $(i,j) \mapsto (n+1-i, n+1-j)$). Dann erhält man eine Matrix mit der rechten Form, kann die cholesky-zerlegung $LL^T = B$ berechnen und erhält dann, dass L gespiegelt (auch wieder gleich) eine untere Dreiecksmatrix mit $\hat{L}\hat{L}^T = A$.

Aufgabe 5: Implementierung des Algorithmus und empirische Aufwandsschätzung

Implementieren Sie Ihren modifizierten Cholesky-Algorithmus in Python und weisen Sie empirisch nach, dass der Aufwand linear in n wächst. Vergleichen Sie die Performance Ihrer Implementierung mit der Python-Funktion `scipy.linalg.cholesky`, wobei die Skyline-Matrix A als vollbesetzte Matrix gespeichert ist.

TODO Anhang Python-Code (+ Grafik Performance?)