Prof. Dr. Reinhard Klein
Elena Trunz

Universität Bonn
Institut für Informatik II
27.11.2020

Wintersemester 2020/2021

Übungen zu Angewandte Mathematik: Numerik - Blatt 4

Die Abgabe der Lösungen zu den Aufgaben 1 und 3 erfolgt als pdf-Datei und zu den Aufgabe 2 und 4 als .py-Dateien bis Freitag, den 04.12.2020, 10:15 über die eCampus-Seite der entprechenden Übungspruppe.

Aufgabe 1 (Sparversion der SVD, *3+1=4 Punkte*)

In dieser Aufgabe untersuchen wir, wie man die Singulärwertzerlegung einer $\mathbb{C}^{m\times n}$ Matrix mit $m \geq n$ schneller durchführen kann und warum. Gegeben sei die Matrix A:

$$\mathbf{A} := \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 1 \end{pmatrix}$$

a) Berechne die Singulärwertzerlegung $U\Sigma V^T = A$ von Hand. Gib alle wichtigen Zwischenschritte an. Nutze dazu den Algorithmus im Skript s. 34 mit folgenden Änderungen: Die Vektoren u_{n+1} bis u_m aus dem 5. Schritt werden nicht berechnet. Stattdessen werden in U einfach $*$ als Platzhalter eingesetzt.

b) Rekonstruiere aus der Zerlegung wieder die Martix $A = U\Sigma V^T$. Was fällt auf?

Aufgabe 2 (Lineare Gleichungssysteme mit SVD lösen, *4 Punkte*)

Implementiere in FrameworkSVD.py eine Funktion `LinearSolve(A,b)`, die zu einem gegebenen linearen Gleichungssystem $A \cdot x = b$ mit $A \in \mathbb{C}^{m\times n}$ und $b \in \mathbb{C}^m$ die Lösung $x \in \mathbb{C}^n$ (wir nehmen an, dass diese existiert, d.h. $m \geq n$) berechnet und zurückgibt. Dazu solltest du die folgenden Teilaufgaben lösen.

a) Berechne die SVD von A. Verwende dazu die Eigenwertzerlegung von A^TA und die Ideen und Folgerung aus der Aufgabe 1.

b) Schreibe eine Funktion `PseudoInverse(A)`, die die Pseudo-Inverse A^+ von A mit Hilfe ihrer SVD berechnet.

c) Nutze die Pseudo-Inverse und schreibe damit die Funktion `LinearSolve(A,b)`. Dazu kann das lineare Gleichungssystem als lineares Ausgleichsproblem $\|A \cdot x - b\|_2 \to \min$ angesehen werden.

Hinweis: Um eine Eigenwertzerlegung durchzuführen, kann in Python `numpy.linalg.eig` verwendet werden.

Aufgabe 3 (Gewichtetes Fitten von Funktionen, *3 Punkte*)

Wenn man sich nur für das Verhalten der Messungen in einem kleinen Bereich interessiert, bietet es sich an nur Messungen aus diesem Bereich stark zu gewichten. Gegeben sei eine beliebige Menge D, Funktionen $f_1, \ldots, f_n : D \to \mathbb{R}$, Messpunkte $u_1, \ldots, u_m \in D$ und zugehörige Messwerte $v_1, \ldots, v_m \in \mathbb{R}$.

Ferner seien Gewichte für die Messpunkte $w_1, \ldots, w_m \in \mathbb{R}$ gegeben, d.h. $w_i \geq 0$ gibt an wie stark die Messung (u_i, w_i) berücksichtigt werden soll.

Wir suchen $x \in \mathbb{R}^n$ so, dass die Funktion $f := \sum_{j=1}^{n} x_j \cdot f_j$ eine optimale Approximation der Messungen im Sinne der kleinsten, gewichteten Quadrate ist, d.h.

$$E(x) := \sum_{i=1}^{m} w_i \cdot \left| v_i - \sum_{j=1}^{n} x_j \cdot f_j(u_i) \right|^2$$

soll minimiert werden. Zeige, dass für

$$A := \begin{pmatrix} \sqrt{w_1} \cdot f_1(u_1) & \cdots & \sqrt{w_1} \cdot f_n(u_1) \\ \vdots & \ddots & \vdots \\ \sqrt{w_m} \cdot f_1(u_m) & \cdots & \sqrt{w_m} \cdot f_n(u_m) \end{pmatrix} \in \mathbb{R}^{m \times n}, \qquad b := \begin{pmatrix} \sqrt{w_1} \cdot v_1 \\ \vdots \\ \sqrt{w_m} \cdot v_m \end{pmatrix} \in \mathbb{R}^m$$

gilt $E(x) = \|b - A \cdot x\|_2^2$. Wie kann man nun f berechnen?

Aufgabe 4 (Bildkompression mit SVD, *3+1=4 Punkte*)

In dieser Aufgabe soll die Singulärwertzerlegung verwendet werden, um Bilder zu komprimieren. Wir fassen ein quadratisches Schwarzweißbild mit $m \cdot m$ Pixeln als Matrix $A \in \mathbb{R}^{m \times m}$ auf.

a) Schreibe in `FrameworkImage.py` zwei Funktionen `Compress()` und `Decompress()`. Dabei bekommt `Compress()` ein Bild A und die Anzahl p der zu behaltenden Singulärwerte übergeben. Die Kompression erfolgt nun dadurch, dass die Singulärwertzerlegung von A berechnet wird und, in Abhängigkeit von p, Singulärwerte und Spalten von U und V verworfen werden. Als komprimierte Darstellung zurückgegeben werden dann die übriggebliebenen Teile von U und V sowie die übriggebliebenen Singulärwerte. Zusätzlich soll `Compress()` noch das Kompressionsverhältnis zurückgeben, also den Quotient des Speicherbedarfs vor und nach der Kompression. Die Funktion `Decompress()` soll nun die verkleinerten Matrizen entgegen nehmen und das approximierte Bild wieder berechnen.

   Hinweis: Zur Berechnung der Singulärwertzerlegung kannst du die Funktion numpy.linalg.svd nutzen.

b) Das Framework testet die Methode anhand dreier Bilder jeweils mit $1, 4, 8, 32$ und 64 beibehaltenen Singulärwerten. Wie kommen die unterschiedlichen Ergebnisse zustande? Erläutere insbesondere weshalb "Stoff" besser komprimiert wird als "Stoff2".

   Hinweis: Überlege dir zunächst, wie das Bild für $p = 1$ in `Decompress()` zustandekommt und dann was sich bei größerem p ändert.