Wissenschaftliches Rechnen

Aufgabenblatt 2 (Praxis)

wr@isg.cs.uni-magdeburg.de

SoSe 2019

Allgemeine Hinweise:

- Die Aufgaben sind von jeder/m Studierenden *einzeln* zu bearbeiten und abzugeben (Plagiate werden entsprechend der Studienordnung geahndet).
- Verwenden Sie den vorgegebenen Skelett-Code. Die zu implementierenden Funktionen befinden sich in der Datei **main.py**. Ihr Code ist an den mit `# TODO: ...` gekennzeichneten Stellen einzufügen. Die `NumPy`-Funktionen, welche Sie zur Lösung einer Aufgabe nicht verwenden dürfen, sind unter `Forbidden` in der Docstring Beschreibung der entsprechenden Funktion aufgelistet.
- Wir stellen *optional* einige rudimentäre Unit-Tests zur Verfügung, welche Sie verwenden können, um die Funktionalität ihres Codes zu testen (in der zur Verfügung gestellten Form sind diese nicht ausreichend, um die tatsächlich geforderte Funktionalität zu überprüfen). Sie sollten diese Tests während der Implementierung Ihrer Lösung vervollständigen (schlagen Sie ggf. die Funktionalität der Python `unittest` Klasse nach). Sie können die Tests mit dem Aufruf `python3 tests.py -v [Tests.test_<function>]` ausführen.
- Bitte reichen Sie `main.py` mit Ihrem Lösungen bis **Freitag, den 31.05.2019, um 18:00 Uhr** unter obiger Emailadresse ein.

Aufgabe 1: Eigenvektoren (3 Punkte)

Aufgabe 1.1: Potenzmethode (3 Punkte)

Implementieren Sie die Potenzmethode zur Berechnung des Eigenvektors, welcher mit dem größten Eigenwert assoziert ist, in der Funktion `power_iteration()`. Wählen Sie dazu ein geeignetes Konvergenzkriterium. Es soll angenommen werden, dass die übergebene Matrix die Voraussetzungen für die Methode erfüllt.

Aufgabe 2: Eigenfaces (7 Punkte)

Ziel der Aufgabe ist es, den Eigenface-Algorithmus zur Gesichtserkennung zu implementieren.

Aufgabe 2.1: Laden von Bildern (1 Punkt)

Implementieren Sie die Funktion `load_images()`, welche Bilder mit einer vorgegebenen Endung aus einem übergebenen Verzeichnis laden soll. Stellen Sie dabei sicher, dass die Bilder in der numerischen Reihenfolge der Dateinamen geladen werden (`01_02.png` kommt vor `01_03.png`; `01_05.png` kommt vor `02_01.png`; `03.png` kommt vor `05.png`).

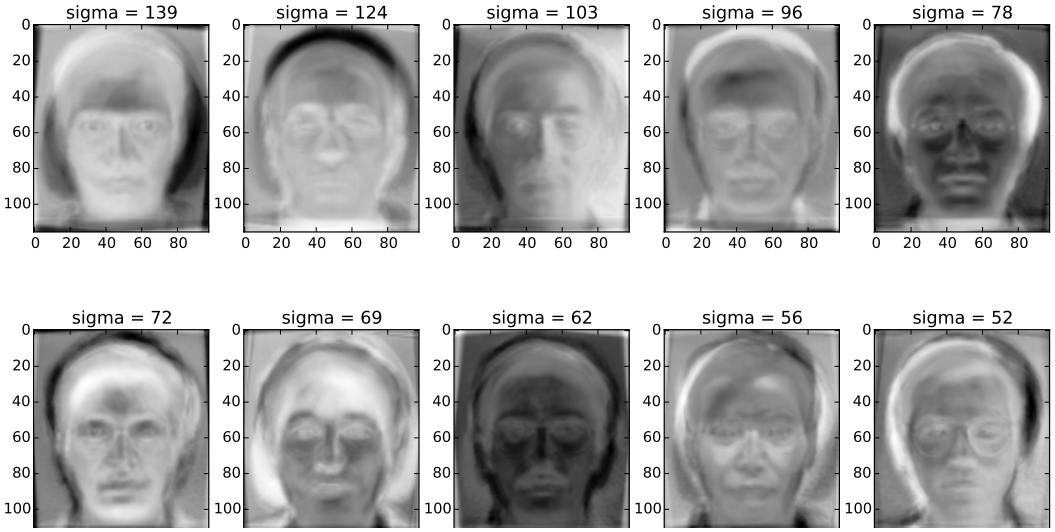
Aufgabe 2.2: Datenmatrix aufstellen (1 Punkt)

Erstellen Sie für die Liste von geladenen Bildern in der Funktion `setup_data_matrix()` die dazugehörige Datenmatrix. Die Bilder sollen dabei die Zeilen der Matrix bilden, mit den Bildern Zeile nach Zeile in den dazugehörigen Bildvektoren, und die Reihenfolge in den Zeilen soll der der Reihenfolge in der Liste entsprechen.

Aufgabe 2.3: Hauptkomponenten-Analyse (1 Punkt)

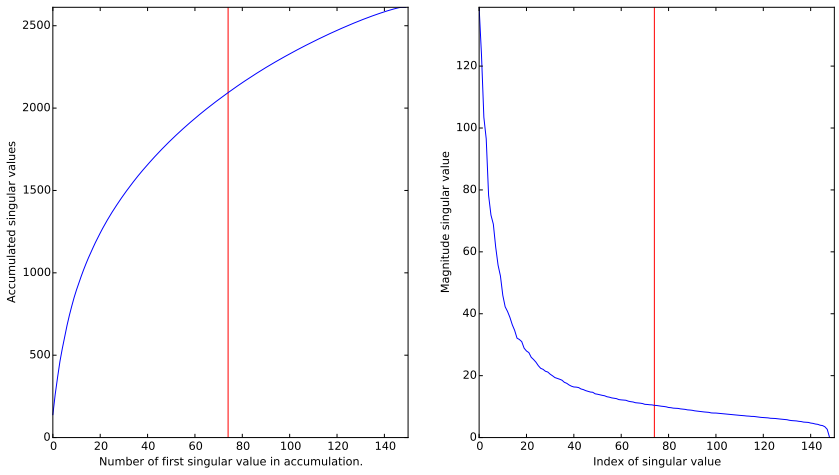
Implementieren Sie die Funktion `calculate_pca()` und berechnen Sie mithilfe der Singulärwert-Zerlegung der zuvor erstellten Datenmatrix die Eigenfaces für den Trainingsdatensatz.

Die ersten zehn Eigenfaces sollen mit Hilfe der Funktion `visualize_eigen_faces()`, welche von uns in `lib.py` zur Verfügung gestellt wird, dargestellt werden. Diese sind:



Aufgabe 2.4: Akkumulierte Magnitude berechnen (1 Punkt)

Berechnen Sie den Index k, so dass 80% der totalen akkumulierten Magnitude der Singulärwerte in den ersten k Hauptkomponenten enthalten ist. Implementieren Sie dazu `accumulated_energy()`. Stellen Sie das Ergebnis mithilfe der Funktion `plot_singular_values_and_energy()`, welche von uns in `lib.py` zur Verfügung gestellt wird, graphisch dar:



Aufgabe 2.5: Projektion in Eigenbasis (1 Punkt)

Implementieren Sie in der Funktion `project_faces()` die Projektion der übergebenen Bilder in den Raum, welcher durch die ersten k Eigenfaces aufgespannt wird, wobei k der von Ihnen in Aufgabe 2.4 berechnete Index ist. Die Koeffizienten der Bilder bezüglich der Eigenfaces sollen in Form einer Matrix zurückgegeben werden, wobei die Koeffizienten eines Bildes eine Zeile der Matrix formen und die Zeilen die Koeffizienten für die Hauptkomponenten absteigend entsprechend der Magnitude der Singulärwerte enthalten.

Aufgabe 2.6: Gesichtserkennung (2 Punkte)

In dieser Aufgabe sollen in der Funktion `identify_person()` Gesichter aus den Testdaten identifiziert werden.

i) Laden Sie die Testdaten aus dem Verzeichnis `'./data/test'` und projizieren Sie diese in die ersten k Eigenfaces. Verwenden Sie hierfür die von Ihnen bereits implementierten Funktionen.

ii) Bestimmen Sie für jedes der Testbilder die Ähnlichkeit zu den Trainingsdaten. Als Ähnlichkeitsmaß soll der Winkel zwischen den Bildern im Eigenface-Raum dienen. Berechnen Sie dafür für jedes Paar von Trainings- und Testbildern den Winkelabstand und geben sie die erhaltenen Werte in Form einer Matrix zurück, wobei die Zeilen den Trainings- und die Spalten den Testbildern entsprechen sollen. Stellen Sie nun mithilfe der Funktion `plot_identified_faces()`, welche von uns in `lib.py` zur Verfügung gestellt wird, neben jedem Testbild das ähnlichste Trainingsbild sowie die Rekonstruktion des Bildes dar.

iii) *Ohne Wertung:* Wie groß ist Ihre Erfolgsquote? Vergleichen Sie die Erfolgsquote, wenn alle und wenn nur die ersten k Eigenfaces verwendet werden.