TU Berlin – FG Angewandte Mathematik
Prof. Dr. Gabriele Steidl
Dr. Michael Quellmalz, M.Sc. Johannes von Lindheim

Numerische Mathematik I

Wintersemester 2021/22

Programmieraufgabe 3

In dieser Programmieraufgabe werden wir die Hauptkomponentenanalyse und den K-Means-Algorithmus verwenden, um handgeschriebene Ziffern maschinell zu lernen. Als Grundlage verwenden wir einen Ausschnitt des MNIST-Datensatzes, welcher aus Trainings- und Testdaten sowie der Zuordnung zu den richtigen Ziffern besteht.

Die Bilder der Trainingsdaten können mit den Funktionen

- `imgs = np.fromfile('train-images-idx3-ubyte', dtype=np.uint8)`
- `imgs = np.reshape(imgs[16:], [-1, 28, 28])`

geladen werden. Die zugehörigen Labels mit

- `labs = np.fromfile('train-labels-idx1-ubyte', dtype=np.uint8)`
- `labs = labs[8:]`

Analog können die Testdaten eingelesen werden.

1. Schreibe zwei Funktionen, welche für eine gegebene Menge an Bildern $\{x_n : n = 0, \dots, N-1\}$ pixelweise den empirischen Mittelwert und die empirische Varianz,

$$b = \frac{1}{N}\sum_{n=0}^{N-1} x_n \qquad \text{und} \qquad V = \frac{1}{N}\sum_{n=0}^{N-1}(x_n - b)^2,$$

   berechnen, wobei das Quadrat komponentenweise angewendet wird. Benutze diese Funktionen, um den Mittelwert und die Varianz aus den ersten 100 Trainingsbildern je Ziffer zu bestimmen. Stelle die zehn berechneten Mittelwerte und Varianzen grafisch dar.

 Abgabe am: 16. Dezember 2021

2. Für die Hauptkomponentenanalyse benötigen wir eine Eigenwertzerlegung der skalierten empirischen Kovarianzmatrix $S = YY^T$ mit $Y := [x_0 - b, \ldots, x_{N-1} - b]$, wobei die Daten x_n als Spaltenvektoren dargestellt wurden. Berechne die Eigenwerte der Matrix S und die Singulärwerte von Y mittels entsprechender `numpy`-Funktionen für die Kovarianzmatrix der ersten 1000 Trainingsbilder und stelle die ersten 50 Eigenwerte und quadrierten Singulärwerte grafisch dar.

3. Schreibe eine Funktion, welche für einen gegebenen Datensatz den bestmöglichen d-dimensionalen affinen Unterraum $H_d = \{At + b : t \in \mathbb{R}^d\}$, sodass

$$(A, b) := \operatorname*{argmin}_{A,b} \sum_{n=0}^{N-1} ||\mathrm{proj}_{H_d}(x_n) - x_n||_2^2,$$

   bestimmt. Benutze diese Funktion, um den Mittelwert und die ersten fünf Hauptkomponenten für die ersten 1000 Trainingsbilder zu bestimmen. Stelle den Mittelwert und die Hauptkomponenten grafisch dar. Schreibe eine Funktion, welche ein Test-Datum auf den bestimmten Unterraum H_d projiziert. Wähle vier beliebige Testdaten, projiziere auf die ersten fünf Hauptkomponenten und stelle Testbild und Projektion grafisch dar.

4. Implementiere den K-Means-Algorithmus. Erzeuge einen Datensatz bestehend aus den ersten 1000 Trainingsbildern für zwei unterschiedliche Ziffern. Projiziere die Daten auf die ersten beiden Hauptkomponenten der gewählten Daten mittels

$$A^T(x_n - b).$$

   Verwende den K-Means-Algorithmus, um die reduzierten Daten in zwei Klassen einzuteilen, wobei die ersten Klassenmittelpunkte den Mittelwerten der projizierten zwei Ziffern entsprechen. Stelle die erhaltene Klassifizierung mit

`matplotlib.pyplot.scatter`

   grafisch dar. Wähle nun für jede der beiden Ziffern 100 Testbilder aus und klassifiziere diese. Stelle in einer Tabelle dar, wie viele Testbilder je Ziffer richtig und falsch klassifiziert wurden.

Verwende außer `skimage`, `numpy`, `scipy` und `matplotlib` keine weiteren Pakete. Kommentiere den Quellcode. Wenn du Jupyter Notebooks verwendest, soll der “Run all”-Befehl für alle Zellen einwandfrei durchlaufen. Ansonsten füge deiner Abgabe ein Hauptprogramm (Python-Skript) bei, welches eigenständig alle Ausgaben erzeugt. Beschrifte die Ausgaben direkt oder erläutere in einer Readme-Datei in welcher Reihenfolge die Ausgaben erfolgen.

Vermeide, so gut es geht, duplizierten Code (code duplication)! Insbesondere sollten Funktionen, die in vorigen Teilaufgaben implementiert wurden, einfach aufgerufen werden, anstatt den Code zu copy-pasten. Generell ist davon abzuraten, Code zu copy-pasten und nur an ein paar Stellen zu modifizieren, weil man eine Änderung (z.B. Bugfix) an mehreren Stellen vornehmen muss. Das erzeugt unnötigen Aufwand und ggf. Bugs, wenn man eine Stelle vergisst. Besser ist allgemein gehaltener Code, der durch Verwendung von Variablen/Parametern unterschiedliche Szenarien abdecken kann.