Metode Numerice - Tema 2

Student: Andronache Madalina-Georgiana Grupa: 312CC

Urmatorul fisier contine informatii despre rezolvarea problemelor propuse in tema 2 de la MN. Punctajul obtinut la testarea locala este de 100 de puncte.

Cea mai mare provocare intalnita a fost rezolvarea corecta si cat mai eficienta a problemele propuse intr-un timp cat mai scurt. Aceasta tema a fost rezolvata pe parcursul a 4 zile: in total am lucrat la aceasta tema 7 h, dintre care 2 h fiind necesare pentru scrierea fisierului README, iar restul pentru codare si testarea problemelor. Totusi, am observat un progres de la prima tema la aceasta, deoarece cunosteam mai bine functionalitatile Octave acum si faptul ca am avut de la inceput checkerul la dispozitie a fost de ajutor.

Cuprins:

1. Task 1

Pentru rezolvarea acestei cerinte am implementat urmatoarea functie:

- function new_X = task1(photo, k) - primind imaginea sub forma unei matrici si un numar k reprezentand numarul de valori singurale luate in considerare, se realizeaza compresia imaginii photo cu ajutorul algoritmului SVD in urmatorul mod: Se aplica SVD pentru matricea photo initiala si se obtin matricele U, S, V. Ulterior, pornind de la aceste matrice, se calculeaza matricele reduse, adica continand doar primele k valori singulare. Folosind matricele reduse, se calculeaza new_X, care este aproximatia matricei initiale photo.

2. Task 2

Pentru rezolvarea acestei cerinte am implementat urmatoarea functie:

- function new_X = task2(photo, pcs) - primind imaginea sub forma unei matrici si un numar pcs reprezentand numarul de componente principale luate in considerare, se realizeaza compresia imaginii photo folosind analiza componentelor principale. Acest algoritm are urmatorii pasi: se calculeaza media pentru fiecare rand din matricea photo. Ulterior, se normalizeaza matricea initiala, scazand din ea media fiecarui rand. Se construieste matricea Z care este egala cu transpusa lui photo impartit la sqrt(n - 1). Se calculeaza descompunerea SVD pentru aceasta noua matrice si se obtine matricea V. Pornind de la aceasta matrice V, se construieste matricea W din primele pcs coloane ale lui V. Calculez matricea Y ca transpusa lui W de inmultit cu matricea initiala photo. Se calculeaza aproximatia matricei initiale new_X, care este returnata ca rezultat al functiei.

3. Task 3

Pentru rezolvarea acestei cerinte am implementat urmatoarea functie:

- function new_X = task3(photo, pcs) - primind imaginea sub forma unei matrici si un numar pcs reprezentand numarul de componente principale luate in considerare, se realizeaza compresia imaginii photo folosind analiza componentelor principale. Acest algoritm seamana foarte mult cu cel de la task-ul 2, diferentele incepand cu calcularea matricei Z, intrucat, in acest caz, matricea Z este matricea de covarianta. Ulterior, calculam valorile si vectorii proprii ai matricei Z si ordonam descrescator in functie de valorile proprii si matricea formata din vectorii proprii, astfel incat, prima coloana sa fie vectorul propriu corespunzator celei mai mari valori proprii si asa mai departe. Pornind de la aceasta matrice V, se construieste matricea W din primele pcs coloane ale lui V. Calculez matricea Y ca transpusa lui W de inmultit cu matricea initiala photo. Se calculeaza aproximatia matricei initiale new_X, care este returnata ca rezultat al functiei.

4. Task 4

Pentru rezolvarea acestei cerinte am implementat urmatoarele functii:

- function im = visualise_image(train_mat, number) - cu ajutorul acestei functii se poate vizualiza imaginea cu numarul number din setul de date. Acest lucru se realizeaza in urmatorul mod: se citeste din train_mat linia cu numarul number si se transforma intr-o matrice cu dimensiunile 28, 28. Transpunem matricea obtinuta si o transformam folosind uint8 astfel incat sa reprezinte o matrice valida.
- function [train_mat, train_val] = prepare_data(name, no_train_images) - aceasta functie primeste ca parametru name, reprezentand un fisier .mat si no_train_images, reprezentand numarul de poze de antrenament. Initial, se incarca datele din fisierul name, folosind functia load. Se salveaza datele images si labels. Folosindu-ne de ele, extragem in matricea train_mat primele no_train_images linii din tabelul de imagini de antrenament si in vectorul train_val primele no_train_images valori ale vectorului de etichete.
- function [train, miu, Y, Vk] = magic_with_pca(train_mat, pcs) - implementarea acestei functii este asemanatoare cu ce a task-ului 3. Se calculeaza matricea de covarianta cov_matrix. Ulterior, calculam valorile si vectorii proprii ai matricei cov_matrix si ordonam descrescator in functie de valorile proprii si matricea formata din vectorii proprii, astfel incat, prima coloana sa fie vectorul propriu corespunzator celei mai mari valori proprii si asa mai departe.
- function sir = prepare_photo(im) - aceasta functie primeste imaginea de test pe care o modifica si o transforma intr-un sir pentru a se putea face mai usor predictia. Deoarece imaginile de antrenament au fundalul negru si cifra alba, iar cele de test au fundalul alb si cifra neagra, trebuie sa inversam culorile imaginii primite ca argument.
- function prediction = KNN(labels, Y, test, k) - aceasta functie pune in aplicare algoritmul k-nearest neighbours care calculeaza cele mai „apropiate” k imagini din setul de antrenament. Algoritmul este urmatorul: Se foloseste matricea Y care reprezinta proiectia matricii cu datele de intrare in spatiul componentelor principale si pentru fiecare rand se calculeaza distanta Euclidiana dintre acesta si vectorul de test primit ca argument. Se ordoneaza crescator distantele obtinute si se memoreaza primele k valori ale imaginilor

cele mai apropriate. Pe baza acestor k valori se calculeaza predictia ca mediana lor.

- function prediction = classifyImage(im, train_mat, train_val, pcs) - pune in aplicare functiile de mai sus pentru a obtine predictia.