Tema 2 – Metode Numerice

Popescu Tudor-Cristian 324CD

Task 1

În acest task ne dorim compresia unei imagini, folosind descompunerea redusa a valorilor singulare . Primim aceasta poza ca parametru sub forma unei matrice. Pentru ca „double” este tipul de date implicit pentru majoritatea calculelor matematice în MATLAB/Octave, vom aplica un „cast” la double pe imaginea/matricea „photo”.

Apoi, cu functia „svd” din Octave, vom realiza descompunerea valorilor singulare. În U vom salva o matrice ortogonala ce conține vectorii singulari stangi, în S vom reține o matrice diagonala cu valorile singulare ale matricei „photo”, iar în V o matrice ortogonala cu vectorii singulari drepti ai matricei originale.

Vom reduce toate aceste matrice prin selectarea primelor k elemente, astfel obținând o imagine aproximativa fata de cea originala, după înmulțire.

Task 2

În acest task vom calcula componenentele principale, utilizand metoda SVD. Următorii pași, deși tot în același scop precum Task-ul 1, ne ajuta sa normalizam datele înainte de a aplica SVD pe matrice, fiecare „feature” din matricea initiala contribuind, acum, în mod egal la SVD.

Astfel, se calculează întâi media fiecarei linii din matricea initiala și se scade din vectorii reprezentați de aceste din urma linii. Urmând formula din enunt calculez matricea Z în funcție de noile date normalizate (noua matrice). Pe ea aplic SVD, precum am explicat la task-ul 1.

Aproximarea matricei initiale va fi data de adunarea dintre vectorul medie al fiecarei linii intiale și o noua matrice determinata de câte componente alegem din matricea de vectori singulari drepti inmultita cu proiectia matricei initiale normalizate în spațiul componentelor principale.

Task 3

Aceste componente principale se pot calcula, de asemenea, folosind matricea de covarianta, data de formula din enunt. Înainte de aplicarea algoritmului, se normalizeaza matricea, ca la task-ul 2.

Utilizand valorile și vectorii proprii ai aceste matrice de covarianta (sortati descrescator), putem determina imaginea aproximativa. Implementarea sortarii pas cu pas este descrisa în comentariile din cod.

Pastrand doar primele coloane din noua matrice de vectori proprii sortati descr. V, obținem o compresie mai buna a datelor. Cu cat crestem numarul de componente principale claritatea imaginii creste, dar de la un numar incolo diferenta nu poate fi sesizata de ochiul uman asa ca pot fi eliminate.

Cu aceste componente principale selectate, ultimii pași (proiectia, formula finala) sunt asemenea task-ului precedent.

Task 4

În acest task vom implementa un program de detectie a cifrelor scrise de mana, utilizand algoritmul PCA (Principal Component Analysis).

Pentru a încarca datele din MNIST, folosim comanda load, iar accesarea se aseamana cu un struct din C, <nume_set_de_date>.<matrice_specifica>. Selectam apoi un numar anume de imagini de antrenament și etichete.

Apoi, pe datele rezultate, se aplica PCA. Primii pași (matricea de covarianta, vectori și valori proprii, sortarea, pastrarea de k elemente) sunt la fel ca la task-ul anterior. Apoi, schimb baza matricei initiale și aproximez matricea intiala. În cadrul acestui algoritm, voi returna, atât aceasta imagine aproximata, cât și vectorul medie, primele componente alese din matricea de vectori proprii sortati și proiectia în spațiul componentelor principale).

Inversez pixelii din imaginea alb-negru, scazandu-i din valoarea maxima 255, o transpun și o transform într-un sir cu functia reshape și dim. [1, 784].

Pentru a realiza predictia, avem nevoie de algoritmul KNN – k-nearest neighbours. Calculez, mai întâi, distanța euclidiana dintre matricea Y și vectorul de test primit ca argument:

distance = sqrt(sum((Y - repmat(test, m, 1)).^2, 2))

repmat(test, m, 1): Funcția repmat replică punctul 'test' de m ori pentru a crea o matrice cu aceleași dimensiuni ca și Y. Această operație se efectuează pentru a asigura că calculul poate fi realizat element-cu-element cu matricea Y.

(Y - repmat(test, m, 1)).^2: Operatorul de exponențiere (^2) ridică la pătrat fiecare element din matricea obținută în pasul anterior. Această etapă calculează diferențele la pătrat între fiecare punct din Y și punctul 'test'.

sqrt(sum((Y - repmat(test, m, 1)).^2, 2)): În final, este aplicată funcția radical (sqrt) asupra sumei diferențelor la pătrat. Rezultatul obținut este distanța euclidiană între fiecare punct din Y și punctul 'test'. Rezultatul este un vector de distanțe, în care fiecare element reprezintă distanța între punctul corespunzător din Y și punctul 'test'.

Predictia finala va fi mediana dintre cele mai apropiate k valori din vectorul sortat de distante.

Cei doi algoritmi (PCA și KNN) sunt legați și aplicați pe setul de date în functia „classifyImage”.

Observatii legate de calitatea imaginii

Calitatea imaginii reconstruite depinde de numărul de componente principale reținute. Un număr mai mare de componente principale va oferi o calitate mai bună a imaginii, cu o reprezentare mai precisă a detaliilor fine, texturilor și structurilor. Pe de altă parte, reducerea numărului de componente poate duce la pierderea detaliilor fine, rezultând o imagine mai puțin detaliată sau neclară.

De asemenea, numărul de componente principale selectate afectează direct rata de compresie.

Și complexitatea algoritmilor are de suferit, pe măsura ce crestem numărul de componente principale.