Universitatea POLITEHNICA din București
Facultatea de Automatică și Calculatoare

Utilizarea PCA în recunoașterea facială

Mihalcea Laurențiu
323CB

CUPRINS

Descrierea formală a algoritmului

1. Reprezentarea sub formă matriceală a unei imagini

Întrucât dorim să aplicăm diverse transformări și grupări ale imaginilor ne vom folosi de reprezentarea sub formă matriceală a acestora. Pentru o imagine i codificată folosind schema grayscale, reprezentarea sub formă matriceală va avea următorul format:

$$\boldsymbol{A_i} = \begin{pmatrix} \boldsymbol{a_{1,1}} & \cdots & \boldsymbol{a_{1,k}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a_{l,1}} & \cdots & \boldsymbol{a_{l,k}} \end{pmatrix} \in \boldsymbol{R^{l,k}}$$

Imaginile utilizate în cadrul aplicației vor avea următoarele proprietăți:

1) $a_{i,j} \in [0, 255],\ 1 \leq i \leq l,\ 1 \leq j \leq k.$

2) l și k sunt la fel pentru toate imaginile.

Proprietatea **1)** apare deoarece imaginile sunt codificate folosind schema grayscale și nu sunt normate.
Proprietatea **2)** apare deoarece toate imaginile vor fi puse într-o matrice și deci trebuie să aibă aceeași dimensiune.

2. Reprezentarea sub formă de vector coloană a unei imagini

Deoarece dorim să grupăm imaginile folosite pentru antrenarea aplicației într-o singură matrice pentru a putea calcula și extrage diverse informații, trebuie să transformăm matricele corespunzătoare acestora în vectori coloană. Acest lucru se face folosind următoarea formulă:

$$\eta_i = \begin{pmatrix} a_{1,1} \\ \vdots \\ \vdots \\ a_{1,k} \\ \vdots \\ \vdots \\ a_{l,k} \end{pmatrix} \in R^n, n = l * k$$

3. Pregătirea setului de date

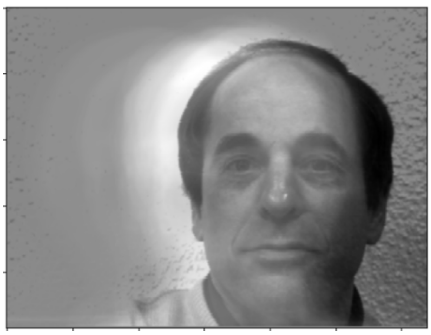
Pentru a ne putea folosi de setul de imagini de antrenament, acestea trebuie centrate în jurul mediei și grupate într-o matrice astfel:

$$I = (\phi_1 \quad \phi_2 \quad \cdots \quad \phi_M), \qquad unde:$$

$$\phi_i = \eta_i - \psi, \qquad \phi_i = \text{imaginea i centrată în } jurul\ mediei$$

$$\psi = \frac{1}{M} * \sum_{i=1}^{M} \eta_i, \qquad \psi = valoarea\ medie\ a\ imaginilor$$

$$M = dimensiunea\ setului\ de\ imagini$$



Exemplu imagine centrată în jurul mediei

Exemplu valoare medie a imaginilor

4. Construirea spațiului fețelor

Acesta este cel mai important pas din cadrul aplicației, întrucât vectorii ce formează acest spațiu se vor folosi în procesul de clasificare a imaginilor necunoscute pentru aplicație.

a) **Descompunerea în valori singulare**

Scopul acestui pas este de a găsi un set de vectori $u_i,\ i = 1, k$ care să reprezinte o bază ortonormată pentru setul de imagini de test. Spațiul span-uit de vectorii din această bază se va numi spațiul fețelor.

În mod normal, pentru a putea găsi un astfel de set, s-ar aplica algoritmul de descompunere în valori proprii, însă, din cauza faptului că matricea I poate să nu fie pătratică, trebuie folosită o altă abordare. Din fericire, algoritmul de descompunere în valori singulare (în engleză, Singular Value Decomposition) ne oferă posibilitatea de a găsi un astfel de set de vectori. O matrice descompusă folosind SVD are următoarea formă:

$$A = U * \Sigma * V^T \; SAU \; A = \sum_{i=1}^{r} \sigma_i * u_i * v_i^T \; , unde:$$

$$i) \; A * A^T = U * \Sigma * \Sigma^T * U^T$$
$$ii) \; A^T * A = V * \Sigma^T * \Sigma * V^T$$

Din formula i) se poate deduce faptul că U conține vectorii proprii ai matricei $A * A^T$. Deoarece această matrice este simetrică și pătratică, conform teoremei spectrale, vectorii din U vor reprezenta o bază ortonormată pentru R^n.

În mod normal, am încerca să-l calculăm pe U folosind descompunerea în valori proprii a matricei $A * A^T \in R^{n \, x \, n}$. Deoarece n este foarte mare raportat la valoarea lui M (n = 77760), îl vom calcula mai întâi pe V^T și vom deduce u_i conform ecuației:

$$u_i = \frac{A * v_i}{\sigma_i}$$

(Notă) Produsul dintre A și v_i se împarte la σ_i pentru a putea obține un u_i normat în cazul în care v_i nu a fost deja normat. În caz contrar, produsul nu mai trebuie normat.

b) **Selecția celor k cele mai importante componente**

În cadrul acestui pas se vor selecta k vectori din U care descriu „cel mai bine” setul de imagini. Selecția acestor valori se face în funcție de coeficienții σ_i corespunzători vectorilor u_i. Conform Eckart -Young (1), dacă alegem primii k u_i care au cele mai mari valori pentru σ_i vom obține cea mai bună aproximare de rang k a matricei I. Acest pas are rolul de a reduce dimensiunea datelor în momentul în care se va face clasificarea.

$$(1) \; Dacă \; B \; are \; rangul \; k, atunci \; \|A \; - \; B\| \; \geq \; \|A - A_k\|$$

5. Clasificarea imaginilor

Pentru a putea clasifica o nouă imagine i (prin imagine nouă ne referim la o imagine care nu a fost adăugată la matricea de imagini I, adică o imagine de test), trebuie să proiectăm varianta centrată în medie a imaginii pe spațiul fețelor și să calculăm distanța dintre vectorul ce rezultă din această proiecție și proiecția vectorului medie al fiecărei clase (1). Pe lângă asta, trebuie să calculăm distanța dintre varianta centrată în medie a imaginii și proiecția acesteia pe spațiul fețelor înmulțită cu matricea U(2). Această ultimă distanță trebuie calculată pentru a putea vedea dacă în imagine se află o față sau nu.

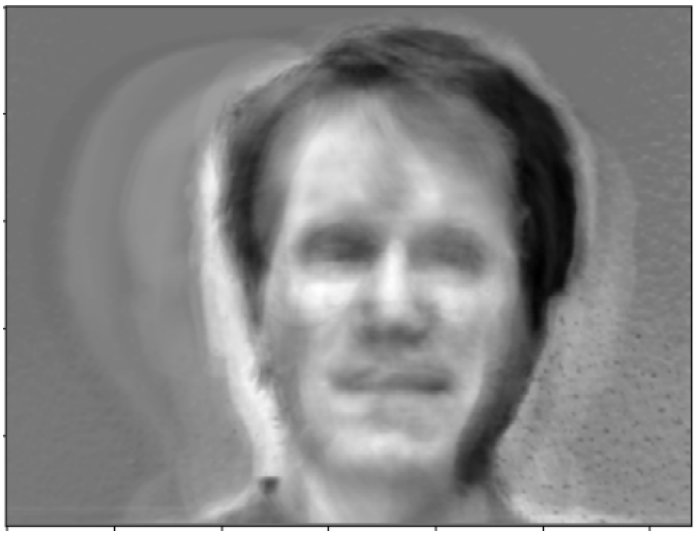
$$(1)\ \Omega = U^T * (\eta - \psi), \qquad \Omega = proiecția\ imaginii\ pe\ spațiul\ fețelor$$

$$(1)\epsilon_i = \|\Omega - \Omega_i\|,$$
$$\Omega_i = proiecția\ imaginii\ medie\ a\ claseii\ pe\ spațiul\ fețelor$$

$$(2)\ \theta = \eta - U * U^T * \eta$$

Clasificare: 1) $\epsilon_i < \mu_{clasă}$ și $\theta < \mu_{proiecție}$ atunci imaginea aparține clasei i.
2) $\epsilon_i > \mu_{clasă}$ și $\theta < \mu_{proiecție}$ atunci nu se știe clasificarea imaginii.
3) $\epsilon_i < \mu_{clasă}$ și $\theta > \mu_{proiecție}$ atunci nu se știe clasificarea imaginii.
4) $\epsilon_i > \mu_{clasă}$ și $\theta > \mu_{proiecție}$ atunci imaginea nu conține o față.

(Notă) Prin clasă ne referim la persoană.



Exemplu imagine proiectată pe spațiul fețelor

Efectul modificării parametrilor în performanța algoritmului

Clasificarea imaginilor se face folosind 3 parametrii: numărul de componente principale, threshold-ul pentru proiecție și threshold-ul pentru clasă. Modul în care aceștia sunt aleși poate influența rata de succes a algoritmului. În continuare vom arăta câteva valori ale parametrilor și ratele de succes aferente acestora :

1) k = 10, threshold proiecție = 20000, threshold clasă = 20000
   Rată de succes obținută : 93.33%

2) k = 1, threshold proiecție = 20000, threshold clasă = 20000
   Rată de succes obținută : 40.00%

3) k = 10, threshold proiecție = 10000, threshold clasă = 20000
   Rată de succes obținută: 86.67%

4) k = 10, threshold proiecție = 20000, threshold clasă = 10000
   Rată de succes obținută : 86.67%

5) k = 4, threshold proiecție = 10000, threshold clasă = 20000
   Rată de succes obținută: 33.33%

Utilizare aplicație

1. Utilitare și module necesare

- python 3.8
- matplotlib 3.4.1
- numpy 1.20.2

2. Rulare aplicație

Pentru a putea rula aplicația trebuie modificate valorile variabilelor **train_path** și **test_path** din modulul **main.py**. Variabila **train_path** trebuie să aibă ca valoare calea absolută către fișierul **train_samples** găsit în arhivă, iar variabila **test_path** trebuie să aibă ca valoare calea absolută către fișierul **test_samples** găsit în arhivă

Pentru a rula aplicația se va tasta într-un terminal deschis în fișierul cu **main.py** comanda:

$$python3\ main.py$$

(Notă) Aplicația va merge garantat pe Linux sau macOS. Este posibil să meargă și pe Windows dar acest lucru nu a fost testat.

3. Output aplicație

Pentru setul de date inclus în arhivă și parametrii: k = 10, threshold proiecție = 20000, threshold clasă = 20000, output-ul aplicației va fi următorul:

```
Pred: 12, Actual: 12
Pred: 10, Actual: 10
Pred: 09, Actual: 09
Pred: 14, Actual: 14
Pred: 11, Actual: 11
Pred: 08, Actual: 08
Pred: 13, Actual: 13
Pred: 07, Actual: 15
Pred: 02, Actual: 02
Pred: 06, Actual: 06
Pred: 04, Actual: 04
Pred: 03, Actual: 03
Pred: 01, Actual: 01
Pred: 05, Actual: 05
Pred: 07, Actual: 07
The success rate of the algorithm is 93.33%
```

Pred: X reprezintă clasa din care algoritmul consideră că face parte imaginea curentă (**X** reprezintă numele clasei; de exemplu, în cazul de față 12, 10, 09, 14 etc reprezintă numele unor oameni)

Actual: X reprezintă clasa din care face parte imaginea curentă (**X** reprezintă numele clasei). Dacă clasa ce apare după **Pred:** este la fel cu cea care apare după **Actual:**, atunci algoritm a clasat corect imaginea.

Descriere module aplicație

1. main.py

Acest modul este folosit pentru a putea rula aplicația.

2. ImageHandler.py

Acest modul se ocupă în principiu de prelucrarea și afișarea imaginilor.

3. PCHandler.py

Acest modul se ocupă de calcularea componentelor principale și a proiecțiilor pe spațiul fețelor.

4. Predictor.py

Acest modul se folosește de modulele **2** și **3** pentru a putea clasifica imaginile pe care le primește.

Concluzii

1) Deși este un algoritm destul de simplu, acesta oferă rezultate destul de bune în cazul în care se folosesc imagini clare cu persoane ce respectă un anumit format. Din păcate, în momentul în care apar obiecte care să blocheze fețele sau acestea nu sunt destul de clare, algoritmul nu va mai funcționa la fel de bine.

2) Alegerea parametrilor optimi pentru algoritm trebuie făcută prin încercări repetate și prin analizarea ratei de succes a acestuia.

3) Deoarece se efectuează calcule cu valori mici, algoritmul poate fi destul de instabil din punct de vedere numeric.

Bibliografie

[1] : https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf

[2] : http://ijarcet.org/wp-content/uploads/IJARCET-VOL-1-ISSUE-9-135-139.pdf

[3] : "Linear Algebra and Learning from Data" – Gilbert Strang

[4] : https://www.face-rec.org/databases/ (sursă imagini folosite)