Analisi tempo-frequenza e multiscala

Giulio Nenna - 292399

Homework 1 - Riconoscimento facciale attraverso "Eigenfaces"

Lo scopo dell'Homework è quello di implementare e testare l'algoritmo "*Eigenfaces*" per il riconoscimento facciale attraverso uno script Python. Nell'elaborato verranno presentati gli strumenti teorici a supporto dell'algoritmo, frammenti di codice salienti utilizzati e risultati computazionali ottenuti.

Il dataset presenta 10 ritratti facciali per ciascuno dei 40 soggetti presenti al suo interno. Vengono utilizzate immagini in scala di grigio di dimensione $m \times n$ pixels con $m = 112$ e $n = 92$. Alcuni esempi di immagini utilizzate sono mostrati in Figura 1

(a) Soggetto 1 (b) Soggetto 2 (c) Soggetto 31

Figure 1: Alcuni esempi di immagini presenti nel dataset

L'algoritmo si distingue per la fase di **Training** e quella di **Testing**.

1 Training phase

Le immagini vengono inizialmente importate e vengono generati i dataset di *Training* e *testing*. In particolare nel dataset di training sono presenti le prime 6 immagini per ciascun soggetto mentre in quello di testing le restanti 4. Le immagini sono importate sotto forma di vettori $\{f_1, f_2, \ldots, f_N\}$ in $\mathbb{R}^{mn}$. Il dataset di training contiene $L = Np$ immagini, dove p è il rapporto tra dimensione del training set e dimensione del test set, nel nostro caso $p = 0.6$.

```
faces_all = np.zeros([num_subjects*num_faces_per_subject, size])
faces_train = np.zeros([train_set_size, size])
faces_test = np.zeros([test_set_size, size])
print('Importing data...')
for i in range(num_subjects): #for each subject
    subject_faces = np.zeros([num_faces_per_subject, size]) #array containing all
    faces of the subject
    for j in range(num_faces_per_subject): #for each face
        f=path+'/s'+str(i+1)+'/'+str(j+1)+'.pgm' #compose filepath
        img = pgm.read_pgm(f) #read the file (2D array of shape m by n)
        img = img.reshape(size) #reshape the img into a 1D array
        subject_faces[j,:] = img

    faces_all[i*num_faces_per_subject:(i+1)*num_faces_per_subject,:] =
    subject_faces
    faces_train[i*int(num_faces_per_subject*train_test_ratio):(i+1)*int(
    num_faces_per_subject*train_test_ratio),:] = \
```

```
        subject_faces[0:int(num_faces_per_subject*train_test_ratio),:]
    faces_test[i*int(num_faces_per_subject*(1-train_test_ratio)):(i+1)*int(
    num_faces_per_subject*(1-train_test_ratio)),:] = \
            subject_faces[int(num_faces_per_subject*train_test_ratio):
    num_faces_per_subject,:]
```

Listing 1: Data import

A questo punto viene calcolata la faccia media $\tilde{f} = \frac{1}{L}\sum_{l=1}^{L} f_l$ e il dataset di training viene centrato rispetto alla faccia media, ottenendo nuovi vettori $\phi_l = f_l - \tilde{f}$ con $l \in \{1, \dots, L\}$.

```
L =  faces_train.shape[0]
mean_face = faces_train.mean(axis = 0) #compute mean face
faces_train_center = faces_train-mean_face #centering the dataset
```

Listing 2: Centering data

A questo punto l'idea è quella di trovare una particolare base ortonormale $\{u_1, \dots, u_{mn}\}$ con $u_i \in \mathbb{R}^{mn}, i = 1, \dots mn$ tale per cui:

$$\phi_l = \tilde{f} + \sum_{n \geq 1} \alpha_n u_n$$
$$|\alpha_1| \geq |\alpha_2| \geq, ..., \geq |\alpha_{mn}|$$

La seconda condizione afferma che la base ortonormale che si sta cercando è tale per cui le componenti sono ordinate per importanza. Questo concetto è di fondamentale importanza per la pratica della riduzione dimensionale: se si riesce a trovare una base di rappresentazione delle immagini per cui le prime componenti sono più importanti delle ultime, allora sarà possibile rappresentare le immagini utilizzando il sottoinsieme delle prime componenti perdendo la minor quantità di informazioni possibile.
Sia α_1 il coefficiente rispetto alla prima componente della base u_1 per la generica immagine ϕ_l.

$$\alpha_1 = \langle \phi_l, u_1 \rangle$$

Sia Φ^T la matrice contenente contenente per ciascuna riga le immagini di training:

$$\Phi^T = \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \vdots \\ \phi_L^T \end{bmatrix}$$

Allora il problema della ricerca della prima componente della base u_1 che massimizza in media il modulo del relativo coefficiente per ogni immagine di training si formalizza come:

$$\max_{\|x\|=1} \frac{1}{L} \sum_{l=1}^{L} |\langle \phi_l, x \rangle|^2 = \max_{\|x\|=1} \frac{1}{L} \|\Phi^T x\|_2^2 = \max_{\|x\|=1} \left\langle \frac{1}{L} \Phi\Phi^T x, x \right\rangle = \lambda_1. \tag{1.1}$$

In particolare $\frac{1}{L}\Phi\Phi^T$ è la matrice di varianza-covarianza campionaria di $\{\phi_1, ...\phi_L\}$ e λ_1 è il suo più grande autovalore in modulo. u_1 sarà pertanto l'autovettore relativo a λ_1. Per la ricerca della seconda componente il procedimento è lo stesso, imponendo l'ortogonalità rispetto alla prima componente:

$$\max_{\|x\|=1,\, x \perp u_1} \left\langle \frac{1}{L} \Phi\Phi^T x, x \right\rangle = \lambda_2.$$

La ricerca della particolare base che stiamo cercando si riduce pertanto al problema del calcolo di autovalori e autovettori della matrice $\frac{1}{L}\Phi\Phi^T$.

Un'importante dettaglio implementativo è dato dalla dimensione della matrice $\Phi\Phi^T \in \mathbb{R}^{mn \times mn}$. La quantità mn è potenzialmente molto grande (nel nostro caso $mn \sim 10^3$), rendendo il calcolo

degli autovalori e autovettori un'operazione computazionalmente molto onerosa. Questo problema può tuttavia essere aggirato se si considera che $\text{Rank}(\Phi\Phi^T) \leq L$ e che gli autovalori della matrice $\Phi^T\Phi$ sono gli stessi della matrice $\Phi\Phi^T$. Inoltre:

$$\Phi\Phi^T x = \lambda x \iff \Phi^T\Phi\Phi^T x = \lambda\Phi^T x$$

Pertanto se $y = \Phi^T x$ è un autovettore di $\Phi^T\Phi$, allora $x = \Phi y$ sarà il corrispondente autovettore di $\Phi\Phi^T$. Il calcolo degli autovettori può pertanto essere eseguito sulla matrice $\Phi^T\Phi \in \mathbb{R}^{L\times L}$ con $L \ll mn$ alleggerendo di molto i costi computazionali.

La particolare base che stiamo cercando sarà pertanto data da $\{v_1, \ldots, v_L\}$ dove

$$v_i = \Phi u_i \quad i = \{1, \ldots, L\},$$

$$u_i \text{ autovettore di } \Phi^T\Phi \text{ relativo all'autovalore } \lambda_i,$$

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L.$$