

# **Applicazione di PCA e LDA per il riconoscimento facciale**

Marco Calamai

21 aprile 2020

# Indice

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Riconoscimento facciale</b>   | <b>3</b>  |
| <b>2</b> | <b>Eigenfaces</b>  | <b>4</b>  |
| 2.1      | Principal component analysis . . . . .                                 | 5         |
| 2.2      | Algoritmo Eigenfaces . . . . .   | 6         |
| 2.3      | Ricostruzione dalla rappresentazione a bassa dimensionalità . . . . .  | 9         |
| <b>3</b> | <b>Fisherfaces</b>   | <b>10</b> |
| 3.1      | Linear discriminant analysis . . . . .                                 | 11        |
| 3.2      | Algoritmo Fisherfaces . . . . .  | 12        |
| 3.3      | Ricostruzione dalla rappresentazione a bassa dimensionalità . . . . .  | 14        |
| <b>4</b> | <b>Scatterplots</b>  | <b>14</b> |
| <b>5</b> | <b>Classificazione</b>   | <b>15</b> |
| 5.1      | Test usando un training set rappresentativo per classi . . . . .       | 17        |
| 5.2      | Test usando un training set limitato e poco rappresentativo per classi | 19        |
| 5.3      | Test con variazioni di illuminazione . . . . .                         | 20        |
| <b>6</b> | <b>Conclusioni</b>   | <b>23</b> |

# Listings

|   |                       |    |
|---|-----------------------|----|
| 1 | PCA . . . . .         | 7  |
| 2 | LDA . . . . .         | 12 |
| 3 | Fisherfaces . . . . . | 12 |

# Elenco delle figure

|   |   |    |
|---|---|----|
| 1 | Eigenfaces images . . . . .                                     | 9  |
| 2 | Original Image to reconstruct . . . . .                         | 10 |
| 3 | PCA Face Reconstruction . . . . .                               | 10 |
| 4 | Fisherfaces images . . . . .                                    | 13 |
| 5 | LDA Face Reconstruction . . . . .                               | 14 |
| 6 | 2-D Scatterplots PCA and LDA . . . . .                          | 15 |
| 7 | PCA and LDA score . . . . .                                     | 17 |
| 8 | PCA and LDA to increase of num components . . . . .             | 18 |
| 9 | Yale Facedatabase A, PCA and LDA with small training sample . . | 20 |

|    |   |    |
|----|---|----|
| 10 | PCA LDA and PCA without largest 3,4,5 and 7 components on<br>Yale Facedatabase B with heavy lighting variations . . . . . | 22 |
|----|---|----|

# 1 Riconoscimento facciale

Dati in input un’insieme di immagini di volti associati a soggetti, il riconoscimento facciale consiste nel classificare nuovi volti.

Mediante il riconoscimento facciale, data una nuova immagine di un volto, quello che dobbiamo fare è indicare il nome della persona raffigurante. Tale compito risulta semplice per gli umani, ma può risultare molto difficile per un computer.

L’essere umano vede ripetutamente il volto di tante persone nella sua vita. Ogni volta che incontra qualcuno, ricorda le caratteristiche facciali peculiari di quella persona mediante un processo di estrazione delle caratteristiche (features) piuttosto che dell’intero viso, riconoscendo l’immagine del viso in modo naturale. Naturalmente, questo processo di estrazione delle caratteristiche è un’attività inconscia ed è un processo sconosciuto.

Nei profili del volto umano, la forma e le dimensioni di occhi, naso, bocca e la loro relazione sono state comunemente utilizzate come features.

Con le caratteristiche estratte correttamente possiamo facilmente riconoscere un volto umano. Tuttavia, i capelli, gli occhiali, il rumore nell’immagine o la rotazione di un volto, possono distorcere la forma del viso.

Riconoscere un volto è piuttosto complesso poiché esso è ricco di informazioni e lavorare con ognuna di esse richiede tempo. Risulta più vantaggioso ottenere informazioni uniche e salienti scartandone altre inutili al fine di rendere efficiente il sistema.

Gli algoritmi per il riconoscimento facciale vengono suddivisi da alcuni in due grandi categorie: modelli olistici e feature-based.

Mentre il primo tenta di riconoscere il viso nella sua interezza, il secondo lo suddivide in componenti in base alle features ed analizza ognuna di queste, compresa la sua posizione spaziale rispetto alle altre features.

Algoritmi comuni per il riconoscimento facciale includono: principal component analysis (PCA) usando eigenfaces, linear discriminant analysis (LDA), elastic bunch graph matching, hidden Markov model, multilinear subspace learning, e neuronal motivated dynamic link matching [1].

I due metodi che sono stati trattati in questa tesina sono Eigenfaces e Fischerfaces. Sia Eigenfaces che Fischerfaces, impiegano un’approccio olistico per il riconoscimento facciale. L’idea alla base di tali metodi è quella di pensare ad un’immagine come ad un punto in uno spazio ad alta dimensionalità. Fatta tale assunzione, viene trovata una sua rappresentazione di dimensionalità inferiore tale che faciliti la classificazione. Per il metodo basato sugli Eigenfaces, il sottospazio a più bassa dimensionalità viene trovato utilizzando PCA che identifica gli assi con varianza massima. Sebbene questo metodo come vedremo risulta ottimo dal punto di vista della ricostruzione dei volti, il principale inconveniente è che non tiene conto delle etichette di classe. Tale inconveniente risulta determinante in situazioni dove la

varianza è generata da sorgenti esterne come ad esempio luci ed ombre. In tali casi gli assi con massima varianza non contengono alcuna informazione discriminatoria e quindi la classificazione diventa totalmente errata.

In questi casi risulta più efficace utilizzare una proiezione specifica per classe con un'analisi discriminante lineare (LDA) per il riconoscimento dei volti.

L'idea alla base di LDA è quella di minimizzare la varianza intra classe massimizzando allo stesso tempo la varianza tra classi diverse.

## 2 Eigenfaces

Eigenface è il nome dato ad un insieme di autovettori quando sono usati in problemi di computer vision nel riconoscimento di volti.

Gli eigenfaces cercano di catturare la variazione in una raccolta di immagini facciali ed usano queste informazioni per codificare e confrontare le immagini di singoli volti in modo olistico (al contrario di metodologie feature-based).

In particolare, gli eigenfaces sono le componenti principali di una distribuzione di volti, oppure equivalentemente sono gli autovettori derivati dalla matrice di covarianza sullo spazio vettoriale ad alta dimensione delle immagini facciali.

Gli stessi autovettori formano un insieme di basi di tutte le immagini utilizzate per costruire la matrice di covarianza. Ciò produce una riduzione dimensionale, consentendo un insieme più piccolo di immagini di base per rappresentare quelle originali.

La classificazione può essere fatta confrontando come i volti sono rappresentati dall'insieme di basi.

In tale ottica ogni immagine di  $N$  pixel, viene considerata come un punto (o vettore) nello spazio  $N$ -dimensionale.

L'idea di utilizzare le componenti principali per rappresentare volti umani è stata sviluppata da Sirovich e Kirby (Sirovich e Kirby 1987) e usata da Turk e Pentland (Turk e Pentland 1991) per il rilevamento e riconoscimento facciale.

Sebbene questo approccio sia considerato come una delle prime tecnologie per il riconoscimento facciale, è ancora oggi considerato come riferimento di base per dimostrare le prestazioni minime attese da un sistema.

Le principali motivazioni nell'utilizzo degli Eigenfaces sono due [2]:

- Estrarre le informazioni facciali rilevanti, che possono o meno essere direttamente correlate all'intuizione umana delle caratteristiche del viso come occhi, naso e labbra. Un modo per farlo è catturare la variazione statistica tra le immagini dei volti.

- Rappresentare le immagini dei volti in modo efficiente. Per ridurre la complessità computazionale e spaziale, ogni immagine del viso può essere rappresentata utilizzando un piccolo numero di parametri.

Come vedremo, ogni Eigenface può essere considerato come un insieme di caratteristiche che individua la variazione globale delle immagini facciali. Ogni volto viene quindi approssimato usando un sottoinsieme di eigenfaces associati agli autovalori più grandi. Queste "features" rappresentano la varianza maggiore nell'insieme dei volti di training.

## 2.1 Principal component analysis

Consideriamo un insieme di  $n$  campioni di immagini  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  ognuno in uno spazio  $d$ -dimensionale e assumiamo che ogni immagine appartenga ad una classe tra  $\{X_1, X_2, \dots, X_c\}$ . Quello a cui siamo interessati è una trasformazione lineare che mappi lo spazio originale  $d$ -dimensionale in uno di dimensione  $m$ , con  $m < d$ .

A tale scopo possiamo utilizzare PCA, che mira a trovare un insieme di assi ortogonali linearmente incorrelati, noti anche come componenti principali (PCs) nello spazio di dimensione  $m$  in modo tale da proiettare i punti di partenza su queste PCs.

L'idea alla base di PCA è che un dataset ad alta dimensionalità sia spesso descritto da un insieme di variabili correlate e che quindi soltanto alcune dimensioni significative rappresentano la maggior parte delle informazioni. PCA trova dunque le direzioni con maggiore varianza nei dati.

Formalmente, dopo aver calcolato l'immagine media da tutti i campioni

$$\mu = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \text{ calcoliamo la matrice di covarianza } S \text{ come:}$$

$$S = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$$

Dove i vari  $\mathbf{x}_i$  sono vettori colonna.

Centrando preventivamente la media del dataset in zero, può essere scritta come:

$$S = \frac{\mathbf{X}\mathbf{X}^T}{n-1}.$$

Si calcolano quindi gli autovalori  $\lambda_i$  e autovettori  $\mathbf{v}_i$  della matrice di covarianza  $S$

$$S\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad i = 1, 2, \dots, n$$

Le  $m$  componenti principali sono ottenute ordinando gli autovettori in ordine decrescente rispetto ai loro autovalori e selezionando i primi  $m$ .

Sia quindi  $W \in \mathbb{R}^{d \times m} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m)$  una matrice con colonne ortonormali, in PCA la proiezione  $W_{opt}$  è scelta come  $W_{opt} = \arg \max_{\mathbf{W}} |W^T S W| = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m]$

dove gli elementi  $\mathbf{v}_i$  con  $i = 1, 2, \dots, m$  sono gli autovettori  $d$ -dimensionali della matrice  $S$ , corrispondenti agli  $m$  più grandi autovalori.

Le  $m$  componenti principali dei vettori osservati  $\mathbf{x}$  sono quindi dati da:

$$\mathbf{y} = W^T(\mathbf{x} - \mu)$$

## 2.2 Algoritmo Eigenfaces

Il metodo Eigenfaces esegue riconoscimento facciale mediante i seguenti step:

- Proietta tutti i campioni di training nel sottospazio PCA
- Proietta l'immagine query nello stesso sottospazio PCA
- Utilizza l'algoritmo k-Nearest Neighbor (K-NN) per trovare il volto più vicino a quello della query tra quelli di training

Quando trattiamo vettori ad alta dimensionalità come nel caso delle immagini, si presenta un ulteriore problema da risolvere.

Prendiamo come esempio uno dei dataset utilizzato per questo progetto (AT&T Facedatabase) contenente 400 immagini di dimensione pari a circa  $100 \times 100$  pixel. PCA trova una matrice di covarianza  $S = XX^T$  (con  $\text{size}(X) = 10000 \times 400$ ) di dimensioni pari a  $10000 \times 10000$  rendendo la risoluzione del problema di complessità molto elevata. Infatti i tipici algoritmi per trovare gli autovettori di una matrice di dimensione  $D \times D$  hanno un costo computazionale dell'ordine di  $O(D^3)$  rendendo di fatto l'applicazione di PCA a problemi di immagini computazionalmente infattibile.

Esistono due principali metodi per risolvere questo problema:

1. *PCA per dati ad alta dimensionalità* [4]: Sia  $X$  una matrice  $d \times n$ , se  $d > n$  allora può avere soltanto  $n-1$  autovalori non nulli. È quindi possibile operare una decomposizione scrivendo la matrice di covarianza come  $S = X^T X$  la quale avrà dimensione  $n \times n$ .

La corrispondente equazione degli autovettori diventa quindi:

$$X^T X \mathbf{v}_i = \lambda_i \mathbf{v}_i$$

Moltiplicando da entrambi i lati la matrice  $X$  dei dati otteniamo quindi gli autovettori originali della matrice di covarianza  $S = XX^T$  come segue:

$$XX^T(X\mathbf{v}_i) = \lambda_i(X\mathbf{v}_i)$$

Gli autovettori risultanti sono ortogonali, per renderli ortonormali dobbiamo operare una normalizzazione per ottenere lunghezza di ogni autovettore uno, dividendo per la loro norma.

L'implementazione in Python dell'algoritmo PCA è riportato nel codice 1.

```

1  def pca(X):
2      #The original matrix must have samples on column and features on row
3      d, n = X.shape
4      X_mean = X.mean(axis = 1).reshape(-1,1)
5      X = X - X_mean
6      if d < n:
7          Cov = np.dot(X, X.T)
8          eigenvalues, eigenvectors = np.linalg.eig(Cov)
9      else:
10         Cov = np.dot(X.T, X)
11         eigenvalues, eigenvectors = np.linalg.eig(Cov)
12         eigenvectors = np.dot(X, eigenvectors)
13         for i in range(n):
14             eigenvectors[:,i] = eigenvectors[:,i]/np.linalg.norm(eigenvectors[:,i])
15     #eigenvectors sorting based on biggest eigenvalues
16     index_sort = np.argsort(-eigenvalues.real)
17     eigenvalues = eigenvalues[index_sort]
18     eigenvectors = np.array(eigenvectors)[:,index_sort]
19     return(eigenvalues, eigenvectors, X_mean)
20

```

Listing 1: PCA

2. *Singular Value Decomposition (SVD)* [5]: SVD è un altro metodo di decomposizione applicabile a matrici reali e complesse. Esso decomponete la matrice originale nel prodotto di due matrici unitarie ( $U, V^*$ ) ed una matrice diagonale rettangolare ( $\Sigma$ ) (non quadrata ma possiede elementi non nulli solo quando gli indici di riga e colonna coincidono).

Formalmente, sia quindi  $X$  una matrice  $d \times n$  allora esiste una fattorizzazione:

$$X = U\Sigma V^*$$

Dove  $U$  è una matrice unitaria di dimensioni  $d \times d$ ,  $\Sigma$  diagonale rettangolare di dimensioni  $d \times n$  e  $V^*$  la trasposta coniugata di una matrice unitaria di dimensioni  $n \times n$ . (Nota che in caso di valori reali la trasposta coniugata  $V^*$  coincide con la trasposta  $V^T$ ).

Nella versione comunemente utilizzata denominata *forma SVD ridotta*  $U$  ha dimensioni  $d \times n$  mentre  $\Sigma$   $n \times n$ . Gli elementi della diagonale di  $\Sigma$  sono i valori singolari di  $X$  con le seguenti proprietà:

$$s_i \geq 0 \quad \forall i \quad s_1 \geq s_2 \geq \dots \geq s_n$$

Il rango della matrice  $X$  è uguale a quello della matrice  $\Sigma$ . In particolare il rango di  $\Sigma$  è uguale al numero di valori singolari diversi da zero.

Si supponga di avere una matrice  $X$  con rango pari ad  $r$ , allora si ha che  $s_1 \geq s_2 \geq \dots \geq s_r > s_{r+1} = \dots = s_n = 0$ .

Le  $r$  colonne della matrice  $U$  e le  $r$  righe della matrice  $V^*$  rappresentano gli autovettori associati agli  $r$  autovalori rispettivamente di  $XX^*$  e  $X^*X$ .

Sulla diagonale della matrice  $\Sigma$  si trovano i valori singolari non nulli di  $X$  che sono le radici quadrate degli autovalori non nulli di  $XX^*$  e  $X^*X$ .

PCA e SVD sono approcci strettamente correlati e possono essere entrambi applicati per scomporre matrici rettangolari. Vediamo la loro relazione.

La decomposizione in PCA può essere scritta, data la matrice di covarianza  $S = \frac{X^T X}{n-1}$ , come la decomposizione di  $S$  nei rispettivi autovalori e autovettori:

$$S = W \Lambda W^{-1}$$

Dove  $W$  e  $\Lambda$  sono rispettivamente la matrice degli autovettori e matrice diagonale degli autovalori di  $S$ .

Eseguendo SVD sulla matrice di covarianza, otteniamo:

$$S = \frac{X^T X}{n-1} = \frac{V \Sigma U^T U \Sigma V^T}{n-1} = V \frac{\Sigma^2}{n-1} V^{-1}$$

Dalla derivazione precedente notiamo che il risultato è nella stessa forma della decomposizione vista poco sopra in PCA e quindi possiamo facilmente vedere la relazione tra valori singolari  $\Sigma$  ed autovalori  $\Lambda$ :

$$\Lambda = \frac{\Sigma^2}{n-1}$$

Ogni componente principale ha la stessa lunghezza delle immagini originali e quindi può essere visualizzata come tale. Chiaramente gli autovalori calcolati potranno avere valori negativi, mentre quelli nella matrice delle immagini dovranno essere positivi, compresi tra 0 e 255, rendendo necessaria quindi una normalizzazione.  
Un esempio di Eigenfaces è mostrato in figura 1.

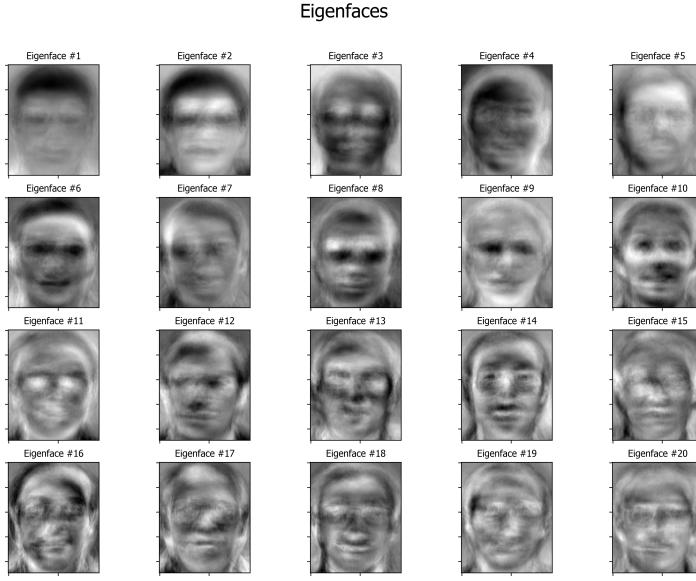


Figura 1: Eigenfaces images

### 2.3 Ricostruzione dalla rappresentazione a bassa dimensionalità

Siano  $\mathbf{y}$  le  $m$  componenti principali dei vettori osservati  $\mathbf{x}$ , è possibile eseguire una ricostruzione dalle basi di PCA come segue:

$$\mathbf{x} = \mathbf{W}\mathbf{y} + \mu$$

Chiaramente la ricostruzione dell'immagine originale migliorerà all'aumentare del numero  $m$  delle componenti principali utilizzate.

L'immagine 3 mostra di quanti Eigenfaces abbiamo bisogno per una buona ricostruzione del volto originale 2.

Come si può vedere dall'immagine 3, già con 70 autovettori, sebbene non si riesca a ricostruire in modo definito e corretto il volto originale, possono essere sufficienti per codificare le principali e determinanti caratteristiche facciali dell'individuo. Intorno ai 200 autovettori invece, iniziamo ad avere una ricostruzione abbastanza accurata, vicina a quella originale di figura 2.

Il numero di Eigenfaces per la buona riuscita della ricostruzione facciale tuttavia è difficile da stabilire a priori ed è molto dipendente dal dataset utilizzato.

Original image



Figura 2: Original Image to reconstruct

Face Reconstruction



Figura 3: PCA Face Reconstruction

### 3 Fisherfaces

PCA altro non fa che trovare una combinazione lineare di features che massimizzi la varianza totale nei dati. Sebbene questo sia un potente metodo per rappresentare i dati, esso non considera alcuna classe e di conseguenza una grande parte di informazione "discriminante" potrebbe venire persa.

Poiché il training set è etichettato, ha senso utilizzare queste informazioni per costruire un algoritmo supervisionato che sia più affidabile per la riduzione della

dimensionalità [3].

Al fine di trovare una combinazione di features che separi al meglio le classi, la Linear Discriminant Analysis (LDA), massimizza il rapporto tra covarianza tra classi e all'interno delle classi. L'idea, suggerita da R. A. Fisher nel 1936 è semplice: le stesse classi dovrebbero raggrupparsi strettamente insieme mentre classi diverse il più lontano possibile da ogni altra.

### 3.1 Linear discriminant analysis

Mantenendo la stessa notazione, possiamo definire le matrici di covarianza "tra classi"  $S_B$  ed "intra classe"  $S_W$  nel seguente modo:

$$S_B = \sum_{i=1}^c N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^c \sum_{\mathbf{x}_k \in X_i} (\mathbf{x}_k - \mu_i)(\mathbf{x}_k - \mu_i)^T$$

Dove  $\mu_i$  è la media delle immagini di classe  $X_i$ ,  $N_i$  è il numero di campioni di immagini della classe  $X_i$  e  $\mu$  è la media totale di tutte le immagini per ogni feature. La proiezione ottimale  $\mathbf{W}_{opt}$  è scelta come la matrice che massimizza il criterio di separabilità tra classi, ovvero la matrice con colonne ortonormali che massimizza il rapporto tra determinante della matrice di covarianza tra le diverse classi dei campioni proiettati e il determinante della matrice di covarianza all'interno delle classi dei campioni proiettati cioè:

$$\mathbf{W}_{opt} = \arg \max_{\mathbf{W}} \frac{|W^T S_B W|}{|W^T S_W W|} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m]$$

Dove i vettori  $\mathbf{v}_i$  sono l'insieme degli autovettori generalizzati di  $S_B$  ed  $S_W$  corrispondenti agli  $m$  più grandi autovalori generalizzati ottenuti da:

$$\begin{aligned} S_B \mathbf{v}_i &= \lambda_i S_W \mathbf{v}_i \\ S_W^{-1} S_B \mathbf{v}_i &= \lambda_i \mathbf{v}_i \end{aligned}$$

Tale equazione prende il nome di Discriminante Lineare di Fischer (FLD dall'inglese Fischer's Linear Discriminant).

Dato che ci sono al massimo  $c - 1$  autovalori generalizzati diversi da zero, questo rappresenta un upper bound di  $m$  [6].

Nel caso specifico del riconoscimento facciale, ci si deve scontrare con la problematica legata al fatto che la matrice  $S_W \in \mathbb{R}^{d \times d}$  è sempre singolare. Questo è conseguenza del fatto che il rango di  $S_W$  è al massimo  $n - c$  ed in generale, il numero di immagini  $n$  nell'insieme di apprendimento è decisamente più piccolo del

numero di pixel  $d$  in ogni immagine [6].

Esistono diversi metodi per superare questo inconveniente. In questo progetto è stato adottato il metodo mostrato in [3]. Tale metodo, evita il problema proiettando le immagini in uno spazio a più bassa dimensionalità in modo tale che in tale spazio la matrice  $S_W$  sia non singolare. Nello specifico, è stato usato PCA per ridurre la dimensionalità ad  $n - c$  e quindi applicato FLD riducendo la dimensionalità a  $c - 1$  come visto sopra.

Il problema di ottimizzazione può dunque essere riscritto come:

$$W_{pca} = \arg \max_{\mathbf{v}} |W^T S_W|$$

$$W_{fld} = \arg \max_{\mathbf{v}} \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

La matrice  $W_{opt}$  è quindi data da:

$$W_{opt} = W_{fld}^T W_{pca}^T$$

*Nota:*  $W_{fld}$  ha dimensione  $(n - c) \times (c - 1)$  e  $W_{pca}$  ha dimensione  $(d) \times (n - c)$ . Quindi il prodotto  $W_{fld}^T W_{pca}^T$  è necessario in quanto, avendo precedentemente ridotto la dimensionalità a  $n - c$  per calcolare LDA, viene riportata alla dimensione originaria  $d$  con tale prodotto.

### 3.2 Algoritmo Fisherfaces

L'algoritmo Fisherfaces esegue riconoscimento facciale mediante gli stessi passaggi visti precedentemente per l'algoritmo Eigenfaces. L'implementazione in Python di LDA e quindi dell'algoritmo Fisherfaces sono mostrati in 2 e 3.

```

1 def lda(X, y):
2     y = np.asarray(y)
3     d,n = X.shape
4     #classes
5     c, Nc = np.unique(y, return_counts=True)
6     meanTotal = X.mean(axis=1).reshape(-1,1)
7     #Def scatter matrix Sw and Sb
8     Sw = np.zeros((d, d), dtype=np.float32)
9     Sb = np.zeros((d, d), dtype=np.float32)
10    for i in c:
11        Xi = X[:,np.where(y==i)[0]]
12        #Mean of i-th class
13        meanClass = Xi.mean(axis=1).reshape(-1,1)
14        Sw = Sw + np.dot((Xi-meanClass), (Xi-meanClass).T)
15        Sb = Sb + np.size(Xi,1) * np.dot((meanClass - meanTotal), (meanClass - meanTotal).T)
16        eigenvalues, eigenvectors = np.linalg.eig(np.linalg.inv(Sw)*Sb)
17        idx = np.argsort(-eigenvalues.real)
18        eigenvalues, eigenvectors = eigenvalues[idx], eigenvectors[:,idx]
19        #Return all eigenvalues and all eigenvectors
20        eigenvalues = np.array(eigenvalues[0:len(c)-1].real, dtype=np.float32)
21        eigenvectors = np.array(eigenvectors[:,0:len(c)-1].real, dtype=np.float32)
22    return [eigenvalues , eigenvectors]
```

Listing 2: LDA

```

1 def fisherfaces(X, y):
2     #The original matrix must have samples on column and features on row
3     y = np.asarray(y)
4     d,n = X.shape
5     c = len(np.unique(y))
6     #Compute PCA
7     eigenval, eigenvec, X_mean = pca(X)
8     eigenvectors_pca = selectPcaComponents(X, eigenvec, num_components = (n-c))
9     X_pca_comp = pcaProject(X,eigenvectors_pca, X_mean)
10    eigenvalues_lda , eigenvectors_lda = lda(X_pca_comp, y)
```

```

11     eigenvectors = np.dot(eigenvectors_pca ,eigenvectors_lda)
12     return [eigenvalues_lda , eigenvectors]

```

Listing 3: Fisherfaces

Anche i Fisherfaces, come visto precedentemente per gli Eigenfaces, possono essere visualizzati come immagini. Utilizzando lo stesso Dataset, i Fisherfaces ottenuti sono mostrati in figura 4 .

LDA, a differenza degli Eigenfaces, cerca le caratteristiche facciali per discriminare tra le persone. Le prestazioni di questo metodo (come anche per gli Eigenfaces) dipendono fortemente dai dati di input. Ad esempio, se si utilizza un training set con volti ben illuminati e si prova successivamente a riconoscere un volto con una scarsa illuminazione è probabile che il metodo trovi le componenti sbagliate (semplicemente perché quelle caratteristiche potrebbero non essere predominanti in immagini scarsamente illuminate ed esserlo in quelle con una buona luce). Questo è anche abbastanza logico dato che l'algoritmo non ha avuto modo di apprendere l'illuminazione.

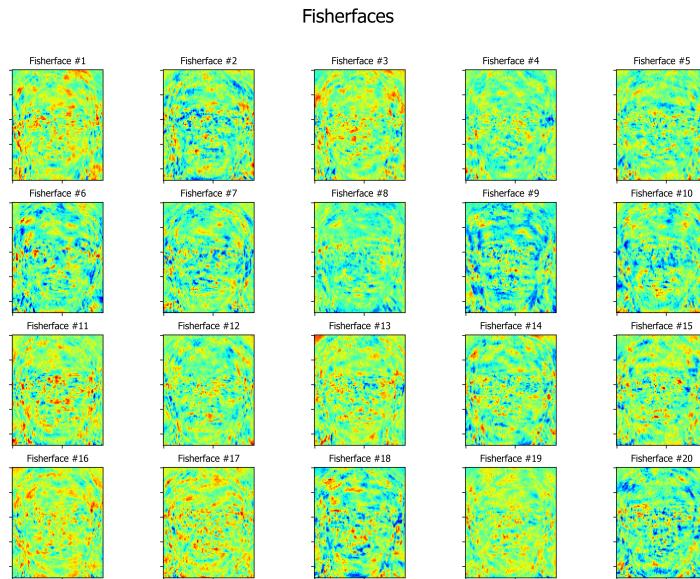


Figura 4: Fisherfaces images

### 3.3 Ricostruzione dalla rappresentazione a bassa dimensionalità

Possiamo provare a ricostruire l'immagine originale da quella a più bassa dimensionalità ottenuta con il metodo Fisherfaces come visto con gli Eigenfaces. Purtroppo però quello che otteniamo non è un buon risultato di approssimazione dell'immagine originale dato che abbiamo identificato con i Fisherfaces soltanto le caratteristiche per distinguere tra soggetti.

Un esempio di ricostruzione dell' immagine 2 vista precedentemente è mostrata di seguito in figura 5.



Figura 5: LDA Face Reconstruction

## 4 Scatterplots

Prima di procedere con la classificazione dei volti, sono stati generati gli scatterplots ottenuti con PCA ed LDA, proiettando tutti i punti dei Datasets AT&T Facedatabase e Yale Facedatabase A in uno spazio a due dimensioni, mostrando graficamente le differenze ottenute dai due algoritmi (vedi sezione 5 per una descrizione dettagliata dei datasets).

Dagli scatterplots risultanti in figura 6 si può notare, come ci si aspetterebbe, che LDA, essendo un algoritmo supervisionato, clusterizza più accuratamente le immagini dei volti quando si dispone di un numero di campioni rappresentativo per classe (in questo caso si ha a disposizione l'intero dataset), riuscendo a raggruppare per classe molto bene.

Sebbene PCA non sia un algoritmo supervisionato, il risultato ottenuto con i due Datasets testati risulta buono, riuscendo comunque a clusterizzare i volti abbastanza bene.

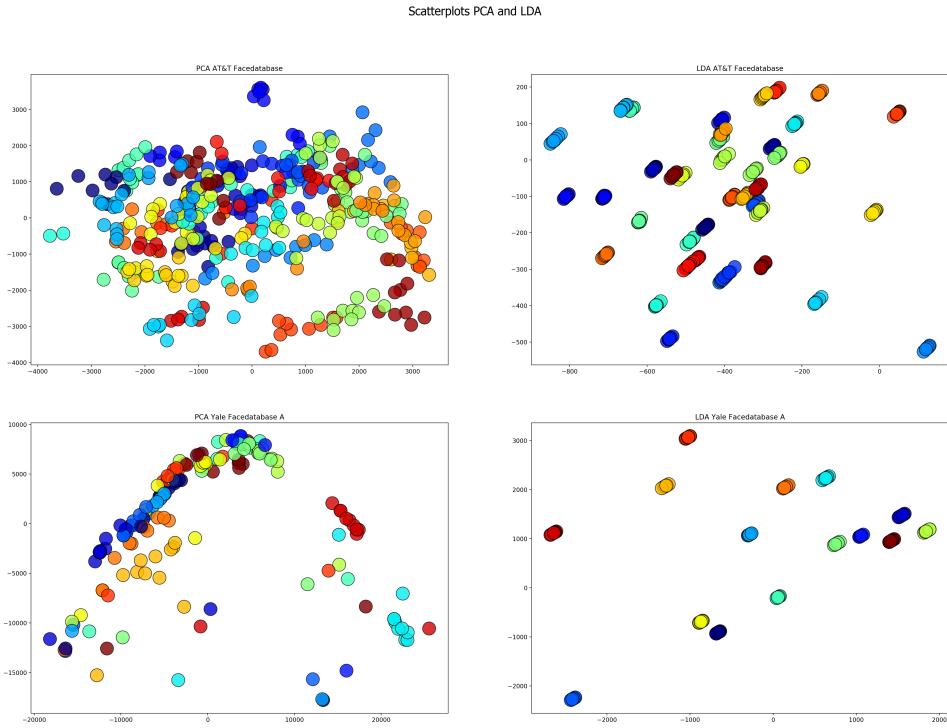


Figura 6: 2-D Scatterplots PCA and LDA

## 5 Classificazione

In questa fase finale quello che è stato fatto è testare i due algoritmi visti provando a classificare volti di alcuni datasets noti.

Le immagini usate sono tutte riportate in scala di grigi, in quanto questo semplifica l'algoritmo e riduce i requisiti computazionali. L'informazione aggiuntiva data dal colore infatti ha spesso un limitato beneficio e l'introduzione di informazioni non necessarie potrebbe aumentare la quantità di dati di training richiesti per ottenere buone prestazioni [7].

I dataset utilizzati sono i seguenti:

- AT&T Facedatabase: Contenente una serie di volti catturati tra aprile 1992 ed Aprile 1994 presso l'Olivetti Research Laboratory di Cambridge, Regno Unito. Il Dataset contiene 10 differenti immagini di 40 soggetti distinti. Per alcuni dei soggetti, le immagini sono state scattate in momenti diversi, variando leggermente l'illuminazione, espressioni facciali (occhi aperti / chiusi,

sorridenti / non sorridenti) e dettagli del viso (occhiali/senza occhiali). Tutte le immagini sono state scattate su uno sfondo scuro, omogeneo e i soggetti sono posizionati frontalmente dall'alto verso il basso.

- Yale Facedatabase A: Contenente 165 immagini in formato GIF di 15 soggetti. Vi sono 11 diverse immagini per soggetto, una per ciascuna delle seguenti espressioni o configurazioni facciali: luce centrale, con occhiali, felice, luce da sinistra, senza occhiali, normale, luce da destra, triste, assonnato (occhi chiusi), sorpreso e con occhialino.
- Yale Facedatabase B: Il più grande dei Datasets utilizzati, presenta 2414 immagini ritagliate di 38 differenti persone. Il Dataset si focalizza sulla variazione di illuminazione dei volti. Ogni volto infatti è ripreso 64 volte, con leggere variazioni dell'espressione facciale ma con pesanti cambiamenti di illuminazione.

Lo schema di classificazione che vedremo presenta alcuni principali inconvenienti: [3]

1. Nel caso in cui le immagini di training e test siano raccolte sotto condizioni di illuminazioni molto differenti allora i corrispondenti punti nello spazio delle immagini non verranno clusterizzati correttamente. Quindi affichè questo metodo funzioni in modo affidabile in base alle variazioni di illuminazione, avremo bisogno di un trainig set che campioni densamente in base alle possibili condizioni di illuminazione.  
Il tema della variazione di illuminazione nelle immagini facciali sarà trattato con alcuni test in seguito.
2. La correlazione tra volti è computazionalmente costosa. Per il riconoscimento di un volto, dobbiamo confrontare l'immagine facciale di test con ogni immagine di training.
3. È richiesta una grande quantità di spazio di archiviazione in quanto il traning set deve contenere numerose immagini di ogni persona per ottenere buoni risultati di classificazione.

Per la classificazione, una volta proiettati i campioni dell'insieme di trainign nel sottospazio (PCA o LDA), ogni immagine di query è stata proiettata nello stesso sottospazio generato e successivamente utilizzato l'algoritmo K-NN per trovare l'immagine del volto più vicina a quella della query tra quelle presenti nel training utilizzando come metrica la distanza euclidea.

## 5.1 Test usando un training set rappresentativo per classi

Per testare le potenzialità dei due algoritmi in condizioni ottimali con un grande training set, quello che è stato fatto è valutare l'accuratezza cercando prima mediante una fase di validazione il miglior paramento (numero di componenti) per entrambi i metodi e successivamente provando a classificare l'insieme di test con il parametro trovato. Nello specifico è stato diviso l'intero Dataset in 75% training set, ottenendo quindi un campionamento abbastanza rappresentativo delle cassi e il rimanente 25% test set.

L'insieme di training è stato utilizzato prima per la ricerca del miglior numero di componenti mediante una 5-Fold Cross Validation e successivamente per generare il sottospazio dei due metodi con il numero di componenti trovato.

Una volta fatto ciò, è stata valutata l'accuratezza sul test set.

I risultati in termini di accuratezza così ottenuti sono riassunti nella tabella 7: Come ci si potrebbe aspettare, LDA supera sempre PCA in accuratezza quando

|                     | PCA            | LDA         |
|---------------------|----------------|-------------|
| AT&T Facedatabase   | Accuracy       | <b>0,93</b> |
|                     | Num components | 11          |
| Yale Facedatabase A | Accuracy       | <b>0,76</b> |
|                     | Num components | 20          |
| Yale Facedatabase B | Accuracy       | <b>0,35</b> |
|                     | Num components | 19          |

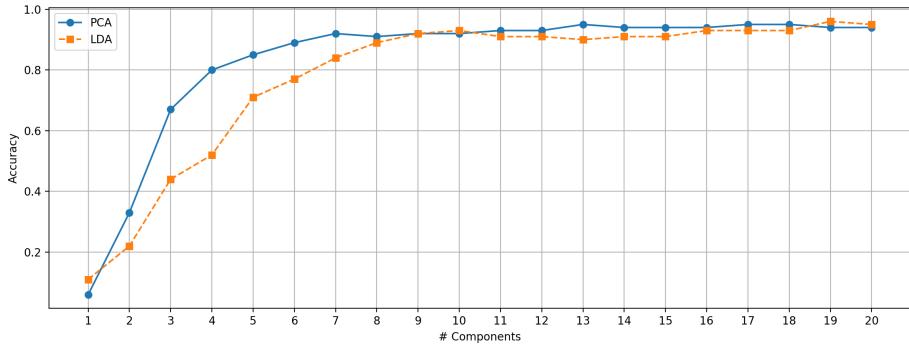
Figura 7: PCA and LDA score

si cerca di classificare con lo schema visto, con un buon numero di elementi di training. LDA infatti trae un enorme vantaggio dal fatto di essere un algoritmo di apprendimento supervisionato, che utilizza le etichette di classe per costruire un metodo più affidabile per la riduzione della dimensionalità che massimizzi la separabilità tra classi.

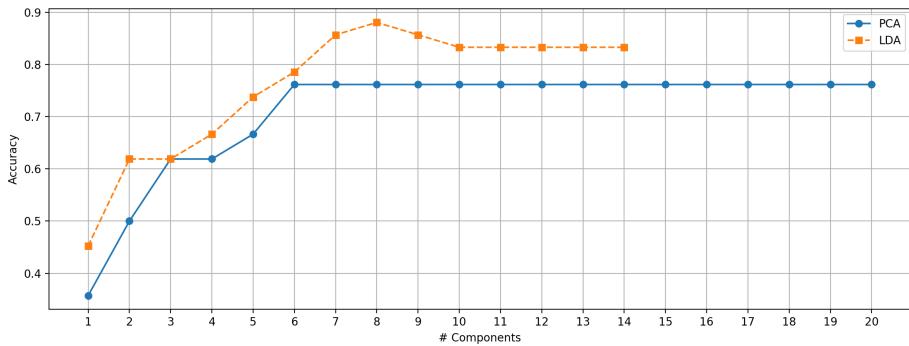
Dalla tabella inoltre si può notare come PCA in generale tragga maggior beneficio da un numero di componenti maggiore, le quali forniscono quasi sempre una migliore accuratezza sui test.

Nei plot sottostanti in figura 8 invece, sono rappresentati i livelli di accuratezza ottenuti quando il numero di componenti utilizzati per i due metodi è lo stesso. Anche in questo caso, si può vedere come la curva di PCA sia quasi del tutto monotona crescente all'aumentare del numero di componenti, cosa che non è sempre vera per LDA.

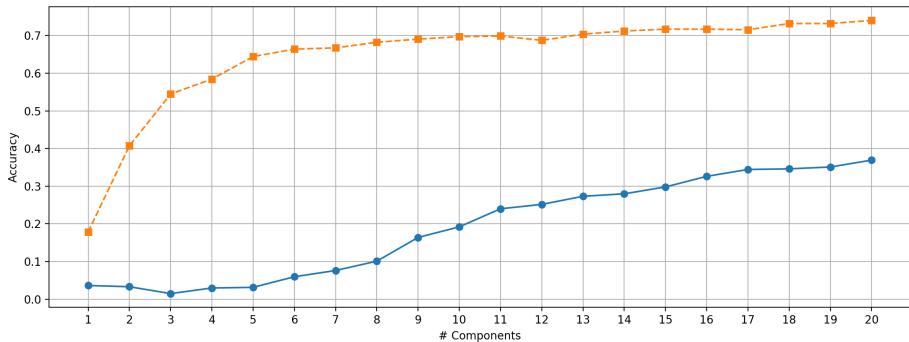
Per quanto riguarda i livelli di accuratezza, fatta eccezione per il datasets AT&T



(a) AT&T Facedatabase PCA and LDA.



(b) Yale Facedatabase A PCA and LDA.



(c) Yale Facedatabase B PCA and LDA.

Figura 8: PCA and LDA to increase of num components

Facedatabase, il quale risulta essere un dataset molto semplice, LDA si comporta ancora generalmente meglio, confermando la sua superiorità su PCA.

## 5.2 Test usando un training set limitato e poco rappresentativo per classi

I test precedenti mostrano come LDA restituisca generalmente migliori risultati di classificazione rispetto a PCA. I test sono però stati fatti con un training set molto grande, che comprendeva il 75% dei volti del Dataset. Vediamo adesso cosa succede nel caso in cui il training set è più piccolo e poco rappresentativo delle classi.

Per eseguire i test è stato creato un insieme di training con pochi (o nessuno) campioni per classe, con lo scopo di capire se i risultati variano rispetto a quelli ottenuti precedentemente.

Per effettuare questi test, è stato usato il Dataset Yalefaces A, in quanto presenta immagini dei volti che hanno poca variazione di illuminazione.

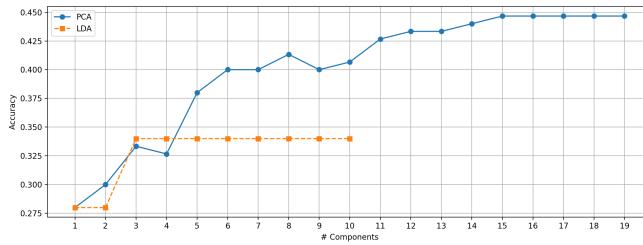
Sono stati eseguiti test con numerosità dell'insieme di training che va da 15 a 25 campioni sui 165 immagini totali.

I risultati ottenuti, al variare del numero di componenti, sono riportati in figura 9, le quali mostrano i risultati rispettivamente con 15, 20 e 25 campioni usati per l'addestramento.

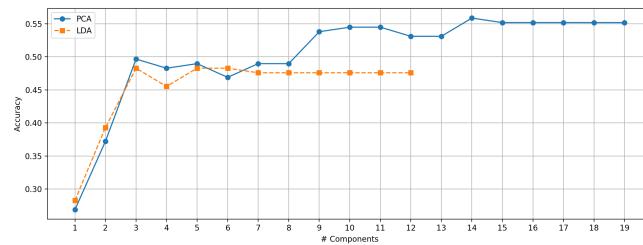
In questo caso PCA migliora molto le prestazioni comparato con LDA, superandolo spesso in accuratezza al variare del numero di componenti utilizzate.

Da ciò si evince che se l'insieme di training è limitato, oppure nel caso in cui ci siano pochi campioni per classe (o anche nessuno per alcune classi) PCA potrebbe superare LDA in accuratezza.

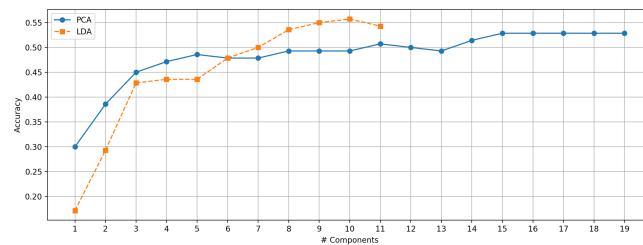
Tale fenomeno potrebbe risultare possibile in molte applicazioni pratiche in cui risulta difficile capire se l'insieme di training è adeguato o meno.



(a) Yale Facedatabase A, PCA and LDA with 15 training samples.



(b) Yale Facedatabase A, PCA and LDA with 20 training samples.



(c) Yale Facedatabase A, PCA and LDA with 25 training samples.

Figura 9: Yale Facedatabase A, PCA and LDA with small training sample

### 5.3 Test con variazioni di illuminazione

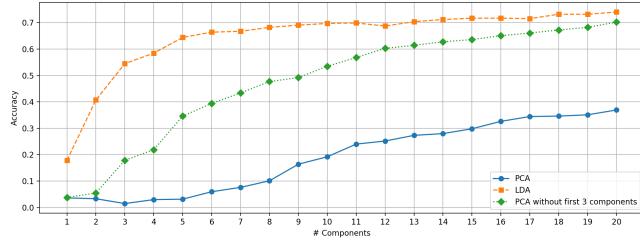
Come ultimo test, è stato preso in esame il Dataset Yalefaces B, nel quale ci sono forti variazioni di illuminazione.

Quello che si è voluto cercare di capire è quanto, in casi del genere, sia efficace la rimozione delle prime componenti di PCA, ipotizzando che queste codifichino l'illuminazione (avendo un'elevata varianza).

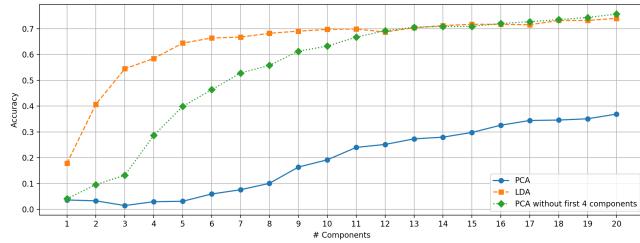
I risultati ottenuti al variare del numero di componenti sono mostrati in figura 10. Dai risultati ottenuti si può vedere come rimuovendo le prime (più grandi) componenti principali di PCA, si ottenga un ottimo miglioramento delle performance nel caso di variazioni di illuminazione come quelle presenti nel Dataset Yale Facedata-

base B, superando anche LDA all'aumentare delle componenti utilizzate, quando si scartano almeno le 4 più grandi.

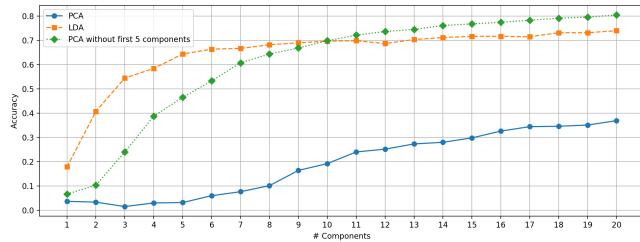
Ovviamente, continuando a scartare componenti l'accuratezza ad un certo punto inizierà a peggiorare.



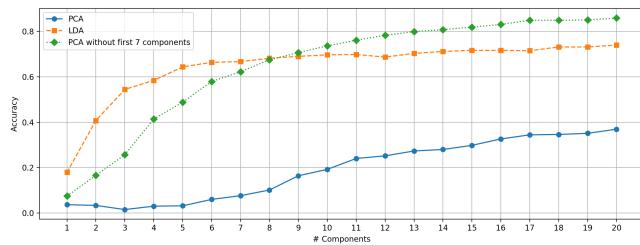
(a) Yale Facedatabase B PCA LDA and PCA without first 3 components.



(b) Yale Facedatabase B PCA LDA and PCA without first 4 components.



(c) Yale Facedatabase B PCA LDA and PCA without first 5 components.



(d) Yale Facedatabase B PCA LDA and PCA without first 7 components.

Figura 10: PCA LDA and PCA without largest 3,4,5 and 7 components on Yale Facedatabase B with heavy lighting variations

## 6 Conclusioni

In conclusione entrambi i metodi sono validi e le loro performance dipendono molto dal contesto e dal Dataset utilizzato come visto in questi brevi esperimenti.

Dai vari test è emerso che:

- Nei casi in cui si ha disposizione un campione di addestramento abbastanza grande e rappresentativo delle classi, il metodo LDA Fisherfaces restituisce generalmente risultati migliori.
- Sebbene si potrebbe pensare che LDA sia sempre migliore di PCA (essendo LDA un algoritmo supervisionato che fa una discriminazione per classi), i test empirici svolti hanno mostrato che esistono casi in cui questo non è vero.
- Generalmente PCA, come la sua versione senza le componenti più grandi, trae maggior beneficio o varia di poco l'accuratezza nei test all'aumentare delle componenti e quindi all'aumentare della dimensionalità dello spazio, a differenza di LDA per il quale ciò non è sempre detto.
- Nei casi in cui ci siano fattori esterni che influenzano le immagini dei volti come variazioni di illuminazione, può essere un'ottima idea scartare le prime componenti di PCA ed eseguire la proiezione senza di esse per cercare di avere un boost prestazionale.
- Il metodo Fisherfaces come visto, perde la capacità di ricostruire volti dal sottospazio a più bassa dimensionalità, cosa che potrebbe essere utile in alcuni contesti. Sarebbe dunque una buona idea mantenere anche il metodo PCA (utilizzato per calcolare i Fisherfaces) in modo tale da utilizzare le sue capacità di ricostruzione.

Chiaramente i test e risultati ottenuti non vogliono essere un'analisi completa dei due algoritmi ma una piccola e semplice comparazione delle loro performance nel riconoscimento di volti in alcuni specifici casi.

## Riferimenti bibliografici

- [1] [https://en.wikipedia.org/wiki/Facial\\_recognition\\_system](https://en.wikipedia.org/wiki/Facial_recognition_system)
- [2] <http://www.scholarpedia.org/article/Eigenfaces>
- [3] Belhumeur, P. N., Hespanha, J., and Kriegman, D. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 7 (1997), 711–720
- [4] Christopher M. Bishop: *Pattern Recognition and Machine Learning* (2006), 569-570
- [5] [https://it.wikipedia.org/wiki/Decomposizione\\_ai\\_valori\\_singolari](https://it.wikipedia.org/wiki/Decomposizione_ai_valori_singolari)
- [6] Xiaofei He , Shuicheng Yan , Yuxiao Hu, Partha Niyogi , Hong-Jiang Zhang: Face Recognition Using Laplacianfaces
- [7] Color-to-Grayscale: Does the Method Matter in Image Recognition: Christopher Kanan , Garrison W. Cottrell