Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 - MMIS Fall Semester 2017
Lessons 5 29 November 2017

EigenFaces -
Face Detection and Recognition with Principal Components Analysis

Outline:

Principal Components Analysis of Face Imagettes.

Introduction

Principal Components Analysis (PCA) is a popular method to reduce the number of dimensions in high dimensional feature vectors. In some cases this can provide an important reduction in computing time with little or no impact on recognition rates. It can also be used to determine an orthogonal basis set for highly redundant features, such as the raw pixels in small windows extracted from images.

While PCA is primarily a data compression method for encoding, it has been successfully used as a method for generating features for detection and recognition. An important example occurred in 1991, with the thesis of Mathew Turk at MIT Media Lab (Directed by A. Pentland). (Turk and Pentland - CVPR 1991). This paper won "best paper" at CVPR and marked the beginning of the use of Appearance based techniques in Computer Vision.

To use the method we require a training set of M face windows (imagettes) $\{W_m\}$ of size R rows and C columns. We will use the set $\{\vec{W}_m\}$ to learn a set of D orthogonal basis images $\vec{\varphi}_d$. These can serve as feature detectors for detection, and recognition.

We can then project an imagette onto basis vectors to obtain a feature vector.

$$\vec{X}_m = \langle W_m, \vec{\Phi} \rangle \text{ where each component is } x_{dm} = \langle W_m, \varphi_d \rangle = \sum_{i=1}^{C} \sum_{j-1}^{R} W_m(i,j)\, \varphi_d(i,j)$$

We can reconstruct the imagette by as a weighted sum of basis images.

$$\hat{W}(i,j) = \mu(i,j) + \sum_{d=1}^{D} x_d \varphi_d(i.j)$$

where $\mu(i,j)$ is the "average" imagette from the training data.

The energy of the difference between the original imagette and reconstructed imagette is a measure of similarity of the imagette to a face.

$$\varepsilon_R^2 = \sum_{i=1}^{C} \sum_{j-1}^{R} E(i,j)^2 \quad \text{where } E(i,j) = W(i,j) - \hat{W}(i,j)$$

We can also train a Bayesian classifier for subsets of face images, for example all images of a particular person. Other recognition techniques are also possible.

Principle Components Analysis of Face Windows

For notation reasons, it is often convenient to map the 2D *W(i,j)* imagettes onto a 1-D vector *W(n)*. This allows us to express using classical vector algebra.

Assume that the imagette is of size C columns by R rows.
allocate a vector W(n) with N=R x C coefficients
For any pixel (i,j), compute $n = j*C + i$
Then W(n) = W(i, j).

Principal components analysis is a method to determine a linear subspace that is optimal for reconstructing a set of vectors.

Assume a set of M training vectors (imagettes): $\{W_m\}$
We are going to use the training data to determine an orthogonal basis set of K vectors to represent $W_m(n)$.

$$\vec{\varphi}(n) = \begin{pmatrix} \varphi_1(n) \\ \varphi_2(n) \\ \vdots \\ \varphi_D(n) \end{pmatrix}$$

To do this we compute

Average vector: $$\mu(n) = \frac{1}{M}\sum_{m=1}^{M} W_m(n)$$

Zero Mean vectors: $V_m(n) = W_m(n) - \mu(n)$

The orthogonal bases are constructed as the Principal Components of the covariance of $\{V_n(m)\}$.

Compose the matrix **V** as

$$\boldsymbol{V} = \left(V_1(n), V_2(n), ..., V_m(N)\right) = \begin{pmatrix} V_1(1) & V_2(1) & ... & V_M(1) \\ V_1(2) & V_2(2) & ... & V_M(2) \\ \vdots & \vdots & \ddots & \vdots \\ V_1(n) & V_2(n) & ... & V_M(n) \end{pmatrix}$$

V has N lines and M columns.

This covariance matrix has N x N = N^2 coefficients.

$$\Sigma_n = E\{V \cdot V^T\}$$

Each column is a training image, $W_m(n)$. Each row is a pixel n← (i,j).
The outer product is a covariance matrix, Σ_n:

$$\Sigma_n = \boldsymbol{V}\boldsymbol{V}^T = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

For an N=RxC imagette there are N^2 coefficients in $\boldsymbol{\Sigma}_n$.
For example, in a 32 x 32 imagette, $\boldsymbol{\Sigma}_n$ is of size 1024x1024.

The coefficients of $\boldsymbol{\Sigma}_n$ are the covariances for the pixels, n.

$$\sigma_{ij}^2 = \frac{1}{M}\sum_{m=1}^{M} V_m(i)V_m(j)$$

The principal components of $\boldsymbol{\Sigma}_n$ are the direction vectors for a rotation that diagonalizes $\mathbf{C}_n$.

$$\varphi^T \Sigma_n \varphi = \varphi^T \boldsymbol{V}\boldsymbol{V}^T \varphi = \Lambda_n$$

such that the Λ is a diagonal matrix composed of N principal values : λ_n.
In English these are called the Eigenvectors $\varphi_n(n)$ and the Eigenvalues, λ_n

We can obtain these vectors from an algorithm for diagonalizing large matrices, such as Householder's method. (See numerical recipes in C or any other numerical methods toolkit).

$(\varphi_n(n),\ \Lambda_n) \leftarrow \mathrm{PCA}(\boldsymbol{\Sigma}_n)$.

The eigenvectors provide an orthogonal basis for the training data. We can any imagette $W(n)$ onto this basis with :

$$\vec{X} = \langle W(n), \vec{\varphi}(n) \rangle = \sum_{n=1}^{N} W(n) \cdot \vec{\varphi}(n)$$

where : $x_d = \langle W(n), \varphi_d(n) \rangle = \sum_{n=1}^{N} W(n) \cdot \varphi_d(n)$ for $d=1,\ldots,D$ with $D \leq N$

We can reconstruct the imagette from the coefficients as a weight sum of the bases using the coefficients of $\vec{X}$.

Reconstruction: $\hat{W}(n) = \mu(n) + \sum_{d=1}^{D} x_d \varphi_d(n)$

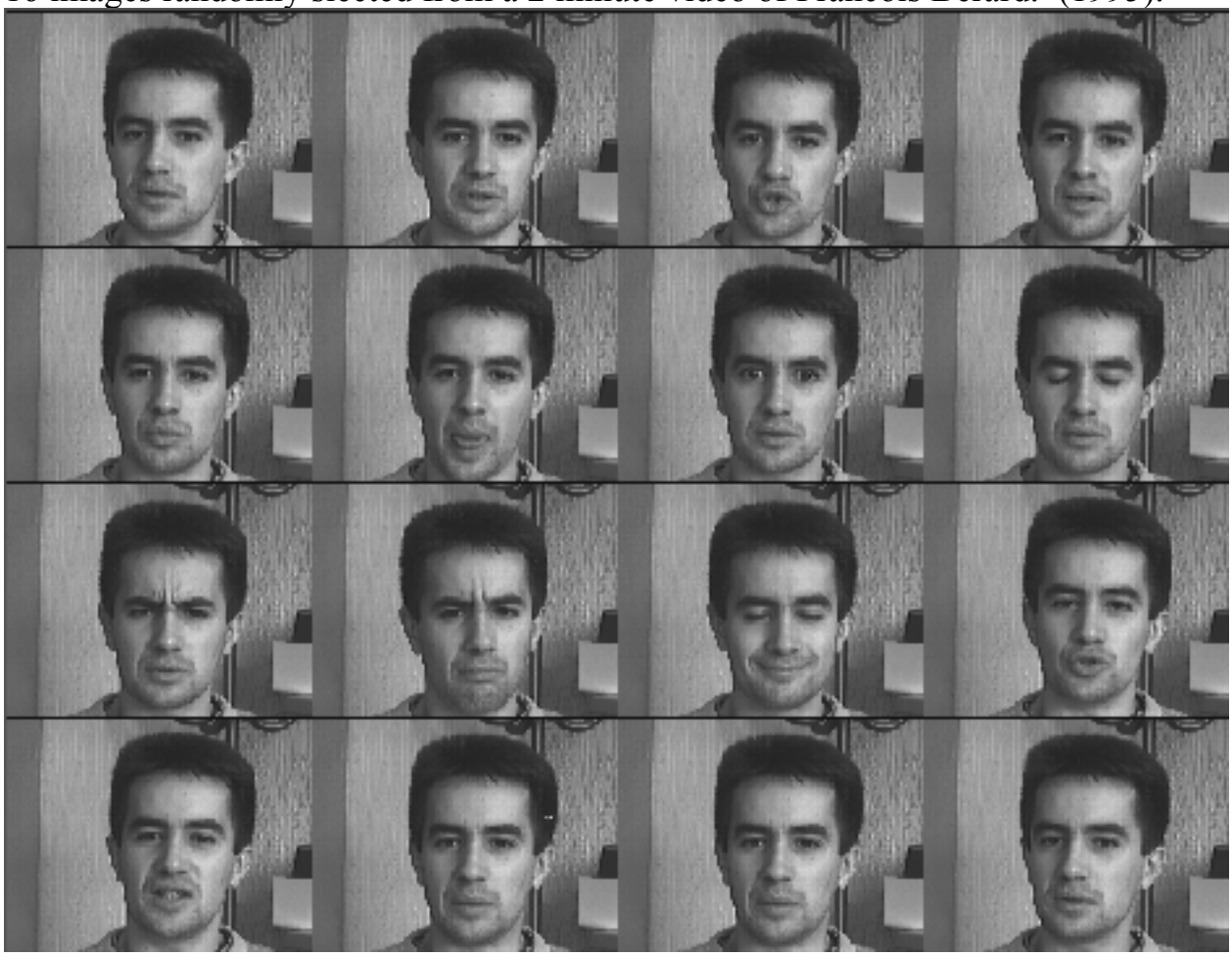
We can determine an error image:

Error image: $R(n) = W(n) - \hat{W}(n)$

Error Energy: $\varepsilon_R^2 = \sum_{n=1}^{N} R(n)^2$

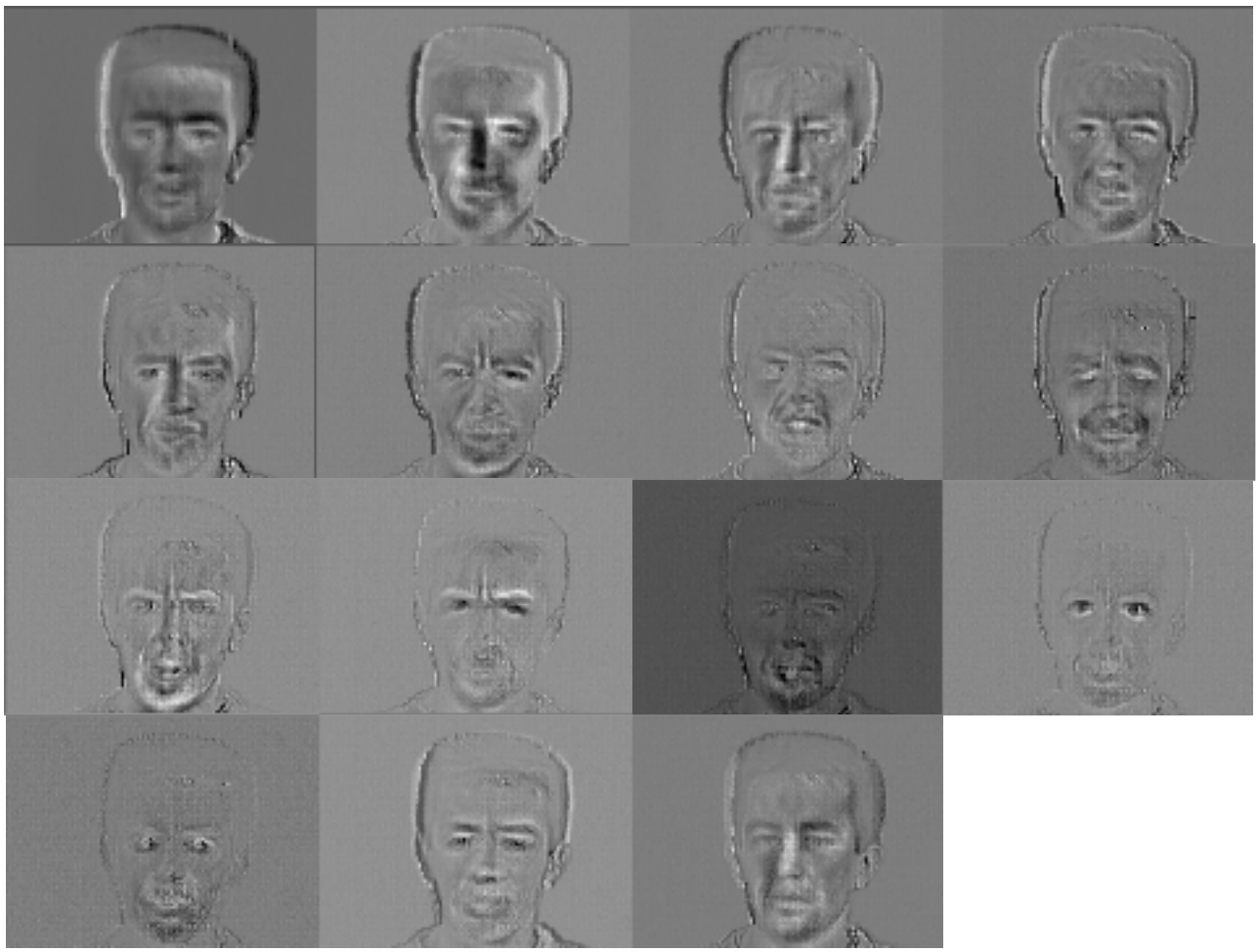
Example

16 images randomly slected from a 2 minute video of Francois Berard. (1995).

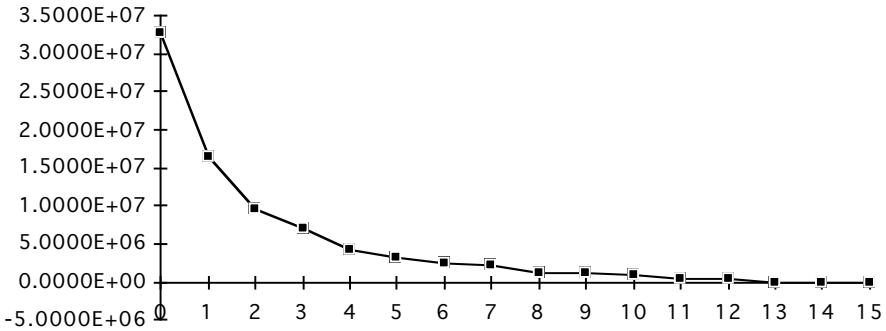


Average Image

Principals Components images



Eigen Values.



Typically only the first 5 or 6 images are needed for 95% of the energy.

Reconstruction

Reconstruction: $\hat{W}(n) = \mu(n) + \sum_{d=1}^{D} x_d \varphi_d(n)$

Image Reconstructed image (120 bytes) Error Image.

Reconstruction (120 bytes)

Image Error

Face Detection using Distance from Face Space

The residue image can be used to determine if a new face imagette, W(n), is "similar" to the eigen-space (linear subspace). In this case, the residue is called the "Distance from Face Space" (DFS)

The distance from Face Space can be used as a face detector!

We scan the image with different size windows, texture map each window to a standard size, then computer the residue (DFS). If the DFS is small, the window contains a face similar to the Face space.

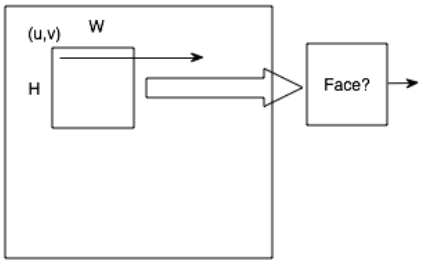
If all N bases used, then any imagette from $\{W_m\}$ can be perfectly reconstructed (except for round off error). For any subset D < N of $\varphi_n(n)$, the residue error will be smaller than with any other basis of D vectors.

Reconstruction: $\hat{W}(n) = \mu(n) + \sum_{d=1}^{D} x_d \varphi_d(n)$

Residue image: $E(n) = W(n) - \hat{W}(n)$

Residue Energy: $\varepsilon_R^2 = \sum_{n=1}^{N} E(n)^2$

Test if an imagette is a face : if (ε_R^2 < Threshold) THEN Face ELSE NOT Face

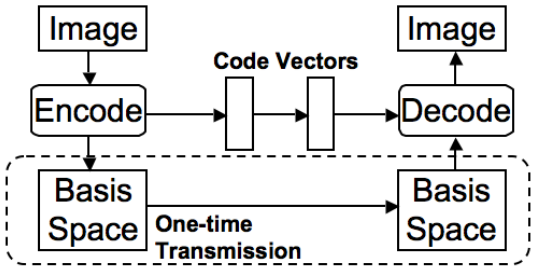


In practice, this method is less effective and more expensive than the cascade classifiers seen last lecture.

Distance from Face space is very effective for normaliseing the position for transmission for video conferencing.
Scan a range of windows and select the position with the least residue.

$$W^*(i,j) = \arg\min_{W(i,j)\in P(i,j)} \{\| W(i,j) - \hat{W}(i,j) \|\}$$

Eigenspace coding for transmission.
Eigenspace coding is a very effective method for signal compression!



In 1996 we were able to demonstrate video telephony in real time (video rates) over a 9600 baud serial line! We ran a video conf with MIT over the internet of the period. In this demo, we used 32 coefficients per image! (32 x 4 - 128 bytes/image).

Eigenspace Coding for Face Recognition.

We can use the coefficients $\vec{X} = \langle W(n), \vec{\varphi}(n) \rangle$ as a feature vector for recognition.

This technique typically used with Gaussian Mixture models,
Let us define the parameters for an I component model as $\nu = (\alpha_i, \vec{\mu}_i, \Sigma_i)$
then

$$p(\vec{X};\nu) = \sum_{i=1}^{I} \alpha_i \mathcal{N}(\vec{X};\vec{\mu}_i,\Sigma_i) \quad \text{where : } \mathcal{N}(\vec{X};\vec{\mu},\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}-\vec{\mu})^T \Sigma^{-1} (\vec{X}-\vec{\mu})}$$

We use an algorithm such as EM to learn a model ν_k for M_k samples of images $\{W_m^k(n)\}$ of the face for each individual, k, from a population of K individuals.

Given an unknown face, W(n): $\vec{X} = \langle W(n), \vec{\varphi}(n) \rangle = \sum_{n=1}^{N} W(n) \cdot \vec{\varphi}(n)$

from Bayes rule we can determine the most probable individual, k as :

$$p(k \mid \vec{X}) = \frac{p(\vec{X} \mid \nu_k)}{\sum_{k=1}^{k} p(\vec{X} \mid \nu_k)}$$

In 1991, Eigen-space coding was a revolutionary technique for face recognition, because it was the first technique to actually seem to work!

However, testing soon revealed that it could only work with:
1) Controlled lighting
2) Pre-defined face orientation (i.e. Cooperating Subject)
3) Normalized image position and size.
4) No occlusions
5) Limited size population
In other conditions the results are unreliable.

In practice, if there are variations in illumination, these will dominate the first eigenvectors. In this case the corresponding eigenvectors are not useful for recognition and can be omitted.

In particular, reliable recognition of a 2D face image with unconstrained face orientation remains an unsolved problem.

Eigenvectors for Large Vectors

For an imagette of size 32 x 32, the vector *W(n)* has 1024 coefficients and the matrix $\mathbf{\Sigma}_n = V\,V^T$ is of size $2^{10} \times 2^{10} = 2^{20}$ coefs.

Most diagonalization methods are capable of handling matrices of up to N = 1024. Diagonalizing usually converges. For larger size vectors, standard techniques tend to fail. There is a clever trick that can be used to get around this.

For larger vectors, diagonalize a covariance matrix based on a subset of the imagettes, M < N.

Instead of computing $\mathbf{\Sigma}_n = VV^T$ we can computer $\mathbf{\Sigma}_m = V^TV$

$\mathbf{\Sigma}_m$ is an MxM covariance between image pairs.
Each term is the covariance of two images, *k, l.*

$$\sigma_{kl}^2 = \frac{1}{N}\sum_{n=1}^{N} V_k(n)V_l(n)$$

We then look for a rotation matrix R such that

$$R^T\,(V^TV)\;R = \Lambda_m$$

Because the same information is used, the first M principal components $\mathbf{\Sigma}_m$ are the same as those of $\mathbf{\Sigma}_n$

$$\Lambda_n = \left(\begin{array}{c|ccc} \Lambda_m & 0 & 0 & 0 \\ \hline 0 & \lambda_{m+1} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{array}\right)$$

and we can note that

$$R\,R^T\,V^T\,V\,R = R\Lambda_m$$

R is a square matrix such that $R^TR = RR^T = I$.

Thus

$$V^T V\ R = R\ \Lambda_m$$

Now multiple by V

$$V\ (V^T V)\ R = (V\ V^T)\ V\ R = V\ R\ \Lambda_m$$

$$= (VV^T)(V\ R) = (V\ R)\ \Lambda_m$$

and Λ_m is an MxM submatrix of Λ_N

As a result, if we substitute $\varphi_m = (V\ R)$
Where φ_m is the first m terms of φ

We see that for the first m terms,

$$= (VV^T)\ \varphi_m = \varphi_m\ \Lambda_m$$

or

$$\varphi_m^T\ (VV^T)\ \varphi_m = \Lambda_m$$

Thus we can use $\varphi_m = (V\ R)$
as the principal components of $\mathbf{\Sigma}_n$

Thus

$$(VV^T)\ \varphi = \varphi\ \Lambda_n = (V\ R)\ \Lambda_m$$

$$\varphi_m(n) = \sum_{l=1}^{M} V_l(n) R(l,m)$$

Reason: The same information was used to construct $\mathbf{\Sigma}$ and $\mathbf{\Sigma}_m$.

For $x_d = \langle W(n), \varphi_d(n) \rangle = \sum_{n=1}^{N} W(n) \cdot \varphi_d(n)$

The coefficients x_d are a code that represents V(n).

This technique can provide principal components of any size image using a sample of M=1024 images.