Linear Algebra and Face Recognition

Chen Yu
Indiana University

Face Detection and Recognition

- One solution: detection of individual features, such as eyes, nose, and mouth.
- Difficult to extend to multiple views.

Face Recognition

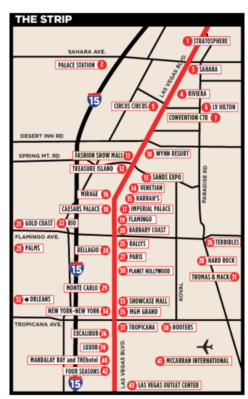
- Another approach: converting an image (N by N) array as a vector of dimension N^2
- E.g. 256 x 256 => a point in 65,536-dimensional space
- A set of images then maps to a collection of points in this huge space.

Face Recognition

- Images of faces, being similar in overall configuration, will not be randomly distributed in this huge image space.
- Patterns in data can be hard to find in a high dimensional space.
- They can be described by a relatively low dimensional subspace.

Dimension reduction

- Feature selection: removing some dimensions so that we can work on a subspace.



Vectors

- Vector Addition and Subtraction

$$c = a + b \Leftrightarrow c_i = a_i + b_i, i = 1,\dots,n$$
$$c = a - b \Leftrightarrow c_i = a_i - b_i, i = 1,\dots,n$$

- Multiplication by a Scalar

$$b = \varepsilon a \Leftrightarrow b_i = \varepsilon a_i, i = 1,\dots,n$$

- Vector Transpose

Linear Combinations

$$\alpha \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_m \end{bmatrix} + \beta \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_m \end{bmatrix} = \begin{bmatrix} \alpha u_1 + \beta v_1 \\ \alpha u_2 + \beta v_2 \\ \dots\dots \\ \alpha u_m + \beta v_m \end{bmatrix}$$

Vector Inner Product

$$\sigma = x \cdot y = \sum_{i=1}^{n} x_i y_i$$
$$\sigma = u^T v = \sum_{i=1}^{n} u_i v_i$$
$$u^T v = v^T u$$

The L_2 Norm

$$\|x\|_2 = (x_1{}^2 + x_2{}^2 + \dots + x_n{}^2)^{1/2} = (\sum_{i=1}^{n} x_i{}^2)^{1/2}$$
$$\|x\|_2 = \sqrt{x \cdot x} = \sqrt{x^T x}$$

Matrix

- Addition and subtraction
- Multiplication by a Scalar
- Matrix Transpose

Matrix: column vectors

$$A = \begin{bmatrix} a_{(1)} & a_{(2)} & \dots & a_{(n)} \end{bmatrix}$$

$$b = Ax; b_i = \sum_{j=1}^{n} a_{ij} x_j$$

$$\begin{bmatrix} a_{(1)} & a_{(2)} & \dots & a_{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b \end{bmatrix}$$

$$[m \times n] \quad [n \times 1] = [m \times 1]$$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}$$

Linear Independence

- Two vectors are not independent if they lie along the same line.

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; y = 2x = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

Matrix Multiplication

- Multiplying a matrix by a vector is a special case of matrix multiplication where

  y = Ax

- This can be written as:

$$y_i = \sum_{k=1}^{N} a_{kj} x_{j,} i = 1,...,M$$

Alternatively we can see the transformation as linear combination of the columns of A

$$y_i = a_1 x_1 + a_2 x_2 + ... + a_N x_N$$

Coordinate Systems

- The vectors a_i have a special interpretation as a coordinate system or basis for a multidimensional space. For example, in the traditional basis in three dimensions,

$$a_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, a_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, a_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- This basis is orthogonal, since

$$a_i \cdot a_j = 0$$

for all i and j such that $i \neq j$

- The basis vector allow y to be written as

$$y = a_1 y_1 + a_2 y_2 + a_3 y_3$$

Other Bases

- A non-orthogonal basis would also work.

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

would still allow y to be represented (although the coefficients would of course be different). However the matrix

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

would not work because there is no way of representing the third component.

Linear Independence

- Consider two linearly independent vectors, u and v, if a third vector, w, cannot be expressed as a linear combination of u and v, then the set {u,v,w} is linearly independent.

Linear Independence

- To represent n-dimensional vectors, the basis must span the space. A general condition for this is that the columns of A must be linearly independent. Formally this means that the only way you could write

$$a_1 x_{(1)} + a_2 x_{(2)} + ... + a_n x_{(n)} = 0$$

would be the case that

$$a_1 = a_2 = ... = a_n = 0$$

Linear Independence

$$\begin{bmatrix} x_{(1)} \,\big|\, x_{(2)} \,\big|\, \ldots \,\big|\, x_{(n)} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- The number of linearly independent vectors is called the rank of the matrix.

- When the rank r is less than the dimension N, the vectors are said to span an r-dimensional subspace.

Change of Bases

- Change of basis, or coordinate transform: If we have data that is defined relative to some basis, we are free to re-map that data into a new basis

$$x^* = Ax$$

- Here A defines our new basis. We can always convert back to the original basis via:

$$x = A^{-1}x^*$$

Eigenvectors

- For any matrix W there are special vectors v such that:

$$Wv = \lambda v$$

v is rescaled by a constant λ. The direction of v is not changed.

- The vectors v are known as eigenvectors, and the associated scalars λ are known as eigenvalues.

Example

$$\begin{bmatrix} 3 & 1 \\ 2 & 2 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 4 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Finding Eigenvectors

- The linear equation Ax = 0 only has a solution (non-trivial) if the columns of A are linearly dependent.
- The columns of A are linearly dependent iff the determinant of A is equal to zero, |A|=0.
- Reminder: the determinant of a 2 x 2 matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is given by |A| = ad - bc

Finding Eigenvectors

For a two-dimensional case:

$$\begin{bmatrix} 3 & 1 \\ 2 & 2 \end{bmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \lambda \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

Or

$$\begin{bmatrix} 3-\lambda & 1 \\ 2 & 2-\lambda \end{bmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

For this equation to have a solution, the columns of the matrix must be linearly dependent, and thus |W|=0. Thus,

$$(3-\lambda)(2-\lambda) - 2 = 0$$

Finding Eigenvectors

- Substituting $\lambda_1 = 4$ into the equation results in

$$\begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- Only one useful equation in two unknowns. Pick $V_1=1$. Then $V_2=1$. Thus the eigenvector associated with $\lambda_1 = 4$ is

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Similar Matrices

- Suppose that a coordinate transformation is given by

$$x^* = Ax$$
$$y^* = Ay$$

- For any matrix W such that,

$$y = Wx$$

- There is a corresponding matrix W*, such that

$$y^* = W^* x^*$$

- What is the relation between W and W*? Given W and A how can we find W*?

W and W* are similar

$$x = A^{-1}x^*$$
$$y = Wx$$
$$y^* = Ay$$

$$y^* = AWA^{-1}x^*$$

$$W^* = AWA^{-1}$$

Diagonalization

- Let's choose the eigenvectors of W as the basis set. For a given eigenvector,

$$Wy_i = y_i\lambda_i$$

- If Y is a matrix whose columns are the eigenvectors, then

$$WY = Y\Lambda$$

- Here Λ is a matrix whose only nonzero components are the diagonal elements λ_i.

$$Y^{-1}WY = \Lambda$$

Covariance

- Variance for a scalar-valued random variable X

$$\mathrm{var}(X) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(X_i - \overline{X})}{(n-1)}$$

- Covariance is a measure on how much the dimensions vary from the mean with respect to each other.

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(Y_i - \overline{Y})(X_i - \overline{X})}{(n-1)}$$

Covariance

- Covariance can be negative? Zero?
- cov(X,Y) == cov(Y,X) ?
- How about >2 dimensions?

Covariance Matrix

$$\Sigma = \begin{bmatrix} \mathrm{cov}(X_1, X_1) & \mathrm{cov}(X_2, X_1) & \dots & \mathrm{cov}(X_n, X_1) \\ \mathrm{cov}(X_1, X_2) & \mathrm{cov}(X_2, X_2) & \dots & \mathrm{cov}(X_n, X_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{cov}(X_1, X_n) & \mathrm{cov}(X_2, X_n) & \dots & \mathrm{cov}(X_n, X_n) \end{bmatrix}$$

M. Turk and A. Pentland (1991).

``Eigenfaces for recognition''.
Journal of Cognitive
Neuroscience, 3(1).

Dana H. Ballard (1999).
``An Introduction to Natural Computation
(Complex Adaptive Systems)'',
Chapter 4, pp 70-94, MIT Press.

Face Recognition

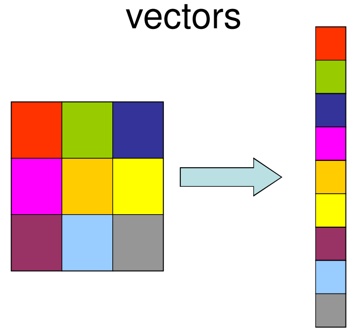
- Considering each image of a face to be a point in a very high dimensional space
- When given an unknown face, compute its distance to all of the existing points in a database of known faces.
- High dimension is bad.
  - Distance metric
  - Sparse

Eigenfaces

PCA

- The eigenvectors of the covariance matrix corresponding to the directions of maximum variance in the data
- The corresponding eigenvalues indicate the amount of variance.
- Therefore, we can transform the new space defined by the directions of maximum variation lie along the axes.

Step 1: Representing images as vectors



Step 1: Representing images as vectors

 $\begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_{N^2} \end{bmatrix}$  $\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_{N^2} \end{bmatrix}$  $\begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_{N^2} \end{bmatrix}$  $\begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_{N^2} \end{bmatrix}$

 $\begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_{N^2} \end{bmatrix}$  $\begin{bmatrix} f_1 \\ f_2 \\ \dots \\ f_{N^2} \end{bmatrix}$  $\begin{bmatrix} g_1 \\ g_2 \\ \dots \\ g_{N^2} \end{bmatrix}$  $\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_{N^2} \end{bmatrix}$

Step 2: computing the mean

$$\vec{m} = \frac{1}{M}\begin{bmatrix} a_1 + b_1 + \dots\dots + h_1 \\ a_2 + b_2 + \dots\dots + h_2 \\ \dots\dots \\ a_{N^2} + b_{N^2} + \dots\dots + h_{N^2} \end{bmatrix}$$

Step 3: subtracting the mean from each image

$$\vec{a} = \begin{bmatrix} a_1 - m_1 \\ a_2 - m_2 \\ \dots\dots \\ a_{N^2} - m_{N^2} \end{bmatrix} \quad \vec{b} = \begin{bmatrix} b_1 - m_1 \\ b_2 - m_2 \\ \dots\dots \\ b_{N^2} - m_{N^2} \end{bmatrix} \quad \vec{c} = \begin{bmatrix} c_1 - m_1 \\ c_2 - m_2 \\ \dots\dots \\ c_{N^2} - m_{N^2} \end{bmatrix} \quad \vec{d} = \begin{bmatrix} d_1 - m_1 \\ d_2 - m_2 \\ \dots\dots \\ d_{N^2} - m_{N^2} \end{bmatrix}$$

$$\vec{e} = \begin{bmatrix} e_1 - m_1 \\ e_2 - m_2 \\ \dots\dots \\ e_{N^2} - m_{N^2} \end{bmatrix} \quad \vec{f} = \begin{bmatrix} f_1 - m_1 \\ f_2 - m_2 \\ \dots\dots \\ f_{N^2} - m_{N^2} \end{bmatrix} \quad \vec{g} = \begin{bmatrix} g_1 - m_1 \\ g_2 - m_2 \\ \dots\dots \\ g_{N^2} - m_{N^2} \end{bmatrix} \quad \vec{h} = \begin{bmatrix} h_1 - m_1 \\ h_2 - m_2 \\ \dots\dots \\ h_{N^2} - m_{N^2} \end{bmatrix}$$

Step 3: Building a matrix

$$A = [\vec{a}, \vec{b}, \vec{c}, \vec{d}, \vec{e}, \vec{f}, \vec{g}, \vec{h}]$$

Covariance matrix

$$AA^T$$

Finding the eigenvectors

- The covariance matrix is too large and the computational effort is too big.

Trick in Eigenface paper

Instead of finding the eigenvectors of the larger system, consider finding the eigenvectors of the M x M system

$$A^T A v = \mu v$$

Multiplying both sides by A,

$$AA^T A v = \mu A v$$

Finding the eigenvectors

$$AA^T Av = \mu Av$$

- If v is an eigenvector of $A^T A$, then Av is an eigenvector of AA^T .

- The eigenvalues of the smaller system are the same as those of the larger system.

- It turns out that these are the M largest eigenvalues.

- To find the eigenvectors of the larger system, first find the eigenvectors of the smaller system, and then multiply the eigenvectors by A.

Eigenfaces

- Eigenfaces define a new coordinate system.
- We can project all the images into the face space.

$$U = Av$$

$$\Omega_1 = U^T \vec{a}; \;\; \Omega_2 = U^T \vec{b}; \;\; \Omega_3 = U^T \vec{c};$$

Face Detection and Recognition

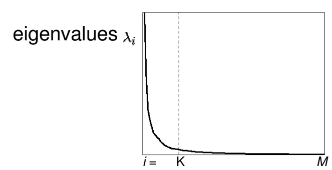
Given a new face

$$\begin{bmatrix} r_1 \\ r_2 \\ \cdots \\ r_{N^2} \end{bmatrix} \Rightarrow \vec{r} = \begin{bmatrix} r_1 - m_1 \\ r_2 - m_2 \\ \cdots\cdots \\ r_{N^2} - m_{N^2} \end{bmatrix}$$

$$\Omega = U^T \vec{r}$$

Whether it is a face and whether it is a known face

Choosing the Dimension K



- How many eigenfaces to use?
- Look at the decay of the eigenvalues
  - the eigenvalue tells you the amount of variance "in the direction" of that eigenface
  - ignore eigenfaces with low variance