Linear Subspaces - Geometry

No Invariants, so Capture Variation

- Each image = a pt. in a high-dimensional space.
  - Image: Each pixel a dimension.
  - Point set: Each coordinate of each pt. A dimension.
- Simplest rep. of variation is linear.
  - Basis (eigen) images: $x_1 \ldots x_k$
  - Each image, $x = a_1 x_1 + \ldots + a_k x_k$
- Useful if $k << n$.

When is this accurate?

- Approximately right when:
  - Variation approximately linear. Always true for small variation.
  - Some variations big, some small, can discard small.
- Exactly right sometimes.
  - Point features with scaled-orthographic projection.
  - Convex, Lambertian objects and distant lights.

Principal Components Analysis (PCA)

- All-purpose linear approximation.
- Given images (as vectors)
- Finds low-dimensional linear subspace that best approximates them.
  - Eg., minimizes distance from images to subspace.

What is PCA good for?

- I can measure how *face-like* a new image is by measuring its distance to a linear face space.
  - Project I onto face space. ai = <I,xi>.
    - This is nearest face-point to I.
  - Measure distance: I = sum(aixi).
- I can speed up comparison between faces by projecting them onto face space (Eigenfaces). This is a big win if I preprocess many faces.
- I can regularize a noisy image.
- Meaningful correspondences critical.

Derivation on whiteboard

- This is all taken from Duda, Hart and Stork *Pattern Classification* pp. 114-117. Excerpt in library.

PCA derivation

(this is all just taken from Duda, Hart and Stork)

Suppose we have a series of vectors, x1…xn, and we want to approximate them with a low-dimensional subspace. What is the best way to do this? If we want to approximate them with a 0 dimensional subspace, we can do this most accurately by approximating them by their mean, m. This is probably intuitive, but if not, Duda, Hart and Stork have a very nice proof (Eq. 80, p. 115).

Next we'll consider find the best 1-dimensional subspace, written as: x_i is approximated by m+a_ie, where e is a unit vector indicating the direction of the space. Then our goal is to choose a_i and e to minimize:

J(a1, …, an, e) = sum ||(m+ake)-xk}}^2

= sum ||ake – (xk – m)||^2

= sum ak^2||e||^2 – 2 sum ak e(xk – m) + sum ||xk – m||^2

||e|| = 1. Taking the derivatives w.r.t. ak and setting them to 0 we get:

2ak – 2 e(xk-m) = 0,

ak = e(xk-m).

We can skip this derivation, and just say that of course we get the best choice of ak by projecting xk-m onto e.

Now, if we set ak = e(xk-m), we get J as a function of e

J(e) = sum ak^2 – 2 sum ak^2 + sum ||xk-m||^2

= - sum [e(xk-m)]^2 + sum ||xk-m||^2

= - sum e(xk-m)(xk-m)e + sum ||xk-m||^2

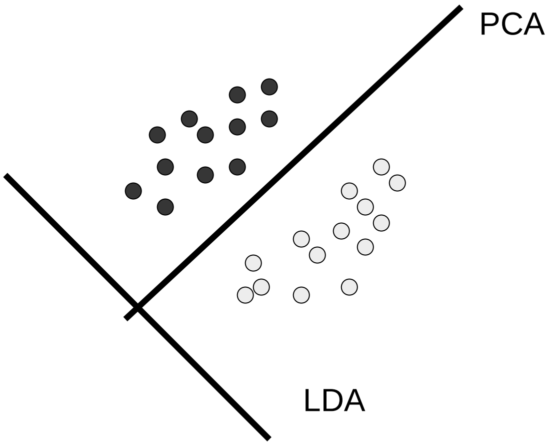
So we need to maximize eSe subject to ||e|| = 1.

We do this with Lagrange multipliers. We set:

U = eSe – lambda (ee – 1), differentiate w.r.t. e and set this to 0. We get:

partial u/ partial e = 2Se – 2 lambda e, so Se = lambda e. So e is an eigenvector of S, and we can see that eSe is maximized when e is the eigenvector associated with the largest eigenvalue.

Fisher Linear Discriminant



LDA for comparison

- Capture variations that distinguish different objects, throw away variations that don’t.

Suppose we have two classes. We project all points onto the direction w. Let the means of the classes be m1 and m2, and their projection onto w be m1' and m2'. If x are the points, and y are their projection, we the variance of each class, after the projection is

s'i = sum (y-m'i)^2

We will maximize J(w) = (m1' – m2')^2/ (s1'^2 + s2'^2).

That is, we maximize the separation in the means relative to the variance within the classes.

Define S1 to be the scatter matrix for points in class 1, similarly define S2, and Sw = S1 + S2.

si'^2 = sum (wx – wmi)^2 = sum w(x-mi)(x-mi)w = wSiw.

s1'^2 + s2'^2 = wSw w

Similarly, (m1' – m2')^2 = wSbw.

J(w) = wSbw/wSww. This is the generalized Rayleigh quotient problem. By taking the partials w.r.t. w and setting them to 0, we can show this is solved when:

Sb w = lambda Sw w, a generalized eigenvalue problem.

Sw^-1 Sb w = lambda w. Sb w is in direction m2-m1. So w = Sw^-1(m2-m1).

SVD

- Scatter matrix can be big, so computation non-trivial.
- Stack data into matrix X, each row an image. SVD gives $X = UDV^T$
  - D is diagonal with non-increasing values.
  - U and V have orthonormal rows.
- $V^T(:,1:k)$ gives first k principal components.
- *matlab*

Linear Combinations

$$
\begin{pmatrix}
u_1^1 & u_2^1 & . & . & . & u_n^1 \\
v_1^1 & v_2^1 & & & & v_n^1 \\
u_1^2 & u_2^2 & & & & u_n^2 \\
v_1^2 & v_2^2 & & & & v_n^2 \\
. & . & & & & . \\
. & . & & & & . \\
. & . & & & & . \\
u_1^m & u_2^m & & & & u_n^m \\
v_1^m & v_2^m & . & . & . & v_n^m
\end{pmatrix}
=
\begin{pmatrix}
s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\
s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\
s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\
s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \\
 & . & & \\
 & . & & \\
 & . & & \\
s_{1,1}^m & s_{1,2}^m & s_{1,3}^m & t_x^m \\
s_{2,1}^m & s_{2,2}^m & s_{2,3}^m & t_y^m
\end{pmatrix}
\begin{pmatrix}
x_1 & x_2 & . & . & . & x_n \\
y_1 & y_2 & & & & y_n \\
z_1 & z_2 & & & & z_n \\
1 & 1 & & & & 1
\end{pmatrix}
$$

I S P

Immediately apparent that *u* and *v* coordinates lie in a 4D linear subspace