L15 -- SVD
[Jeff Phillips - Utah - Data Mining]

Let P subset R^d and |P| = n
Then P = d x n   (usually n > d)

Want to place P in R^k  where k << d

Find R^k subset R^d where
  mu : R^d -> R^k
and minimize
  sum_{p in P} (p - mu(p))^2

Solution:  SVD (PCA)

--------------

U, S, V^T = svd(P)            (in matlab or octave  //  LAPACK)

in fact P = U S V^T

S = diag(s_1, s_2, ..., s_r)  where r<=d where r=rank(P)
    (d x n)
         s_1 >= s_2 >= ... >= s_r >= 0

U (dxd), V (nxn) are orthogonal matrices.

-------------

Orthogonal Matrix U
 - basically rotations about 0, can also do mirror flips
 - each ||u_i|| = 1
 - each u_i, u_j columns U have <u_i, u_j> = 0
 - U^T = U^{-1}

the columns (and rows) of U form a basis (usually not the original basis)
for any p in R^d we can write
  p  = sum_{i=1}^t = a_i u_i
where a_i = <p, u_i> is a scalar
  - permutation matrix is orthogonal

--> thus for any p in R^d  ||U p|| = ||p||  (rotation + flip)

--------------

Consider rank=2 matrix
A = (1/sqrt{2}) [sqrt{3} sqrt{3} ; -3 3 ; 1 1]

b = Ax
transforms circle in plane to ellipse in R^3
 - only uses 2 dimensions in R^3
 - stretches it out along certain axis

[U S V^T] :
U = [0 0.866 -.5;  -1 0 0; 0 0.5 0.866]
S = [3 0; 0 2; 0 0]
V^T = [0.707 0.707; -.707 0.707]

3 steps:
1.  from (x_1, x_2) circle  ->  rotation ->  (xi_1, xi_2)
   where two orthogonal vectors v_1, v_2 map to axis v_1', v_2'

   v_1, v_2  == right singular vectors of A
   V = [v_1 v_2]
   xi = V^T x

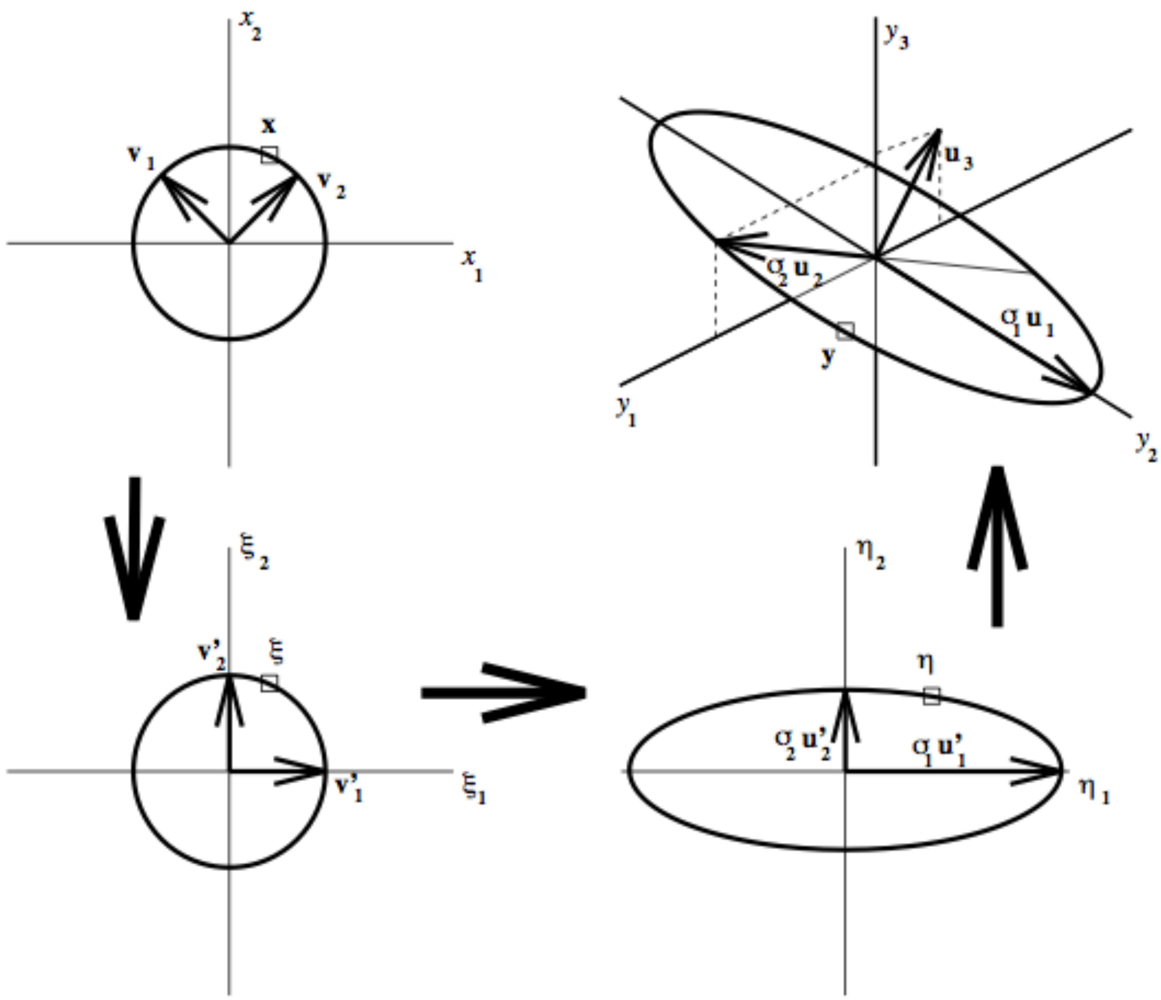
2.  from (xi_1, xi_2) circle -> stretch -> (eta_1, eta_2)
   where eta_1 = s_1 * v_1'
         eta_2 = s_2 * v_2'

   s_1, s_2 == singular values of A
   S = [s_1 0 ; 0 s_2; 0 0]
   eta = S xi

3.  from (eta_1, eta_2) -> rotation -> (y_1, y_2, y_3)
    where sigma_1 * u_1 = y_2
          sigma_2 * u_2 = in span(y_1, y_3)
          u_3 in span(y_1, y_3), but has none of circle
(orthogonal to)

     u_1, u_2, u_3 == left singular vectors of A
     U = [u_1 u_2 u_3]
     b = U eta

b = U S V^T x = A x




---------------

How does this help us get a projection?

given a point x in R^n (with similarities to all n points)
maps to y in R^d (in the space of dimensions)
each y_i is a linear combination of dimensions
y is an orthogonal linear combination of this basis of {y_i}

s_i tells us how much the ith dimension is scaled.
move to an r-dimensional space
- already centered (assumed)
- have Gaussian with std.dev on each axis y_i according to s_i
- if s_i is small, then maybe we don't care
- s_1 chosen to be as large as possible, s_2 as large from what's

left, $s_3$ ...

So set some $s_k$ such that $s_{k+1}$ is small enough.

- statistical data sets (small) typically decay quickly
  and usually $s_{k+1}$ close to 0
- internet data sets (huge) typically decay slowly,
  and $\sum_{j=k+1}^\infty \neq 10\%$

Vectors $u_i$ (n-dimensional) are linear combinations of points
so represent new basis Take $R^k = [u_1\ u_2\ \ldots\ u_k] = U_k$

V does the "bookkeeping" of moving original basis to new one
S stretches it appropriately
U puts the new basis in the proper projection

$P_k$ in $R^k$ <<--- $P_k = U_k^T S_k V_k$

$V_k$ rotates appropriately the top k directions, the others it does not care since gets set to 0.

(if we don't first recenter, then $u_1$, $s_1$ just point to the center)

All we need are $V_k^T$. We can then project to this basis.
$S_k$ tells us how much we save
$S_{k+1}^r$ tells us how much we lost (our "loss" function)

---------------

How do we compute SVD?
+ find top vector (convex problem, but NLA approach better)
+ project to space orthogonal to top vector
REPEAT
since finds large components first, numerically stable.

--------------

Relationship to eigen-decomposition
$P^T P V = V S^2$
so $v_i$ are eigenvectors of $P^T P$
$P P^T U = U S^2$
so $u_i$ are eigenvectors of $P P^T$
and $s_i^2$ are eigenvalues of $P^T P$ and of $P P^T$