Computational Methods in Statistics with Applications
Singular Value Decomposition

Maya Neytcheva, Lars Eldén

SeSe, September 2020

Matrix factorizations/decompositions

Matrix factorizations I

- *LU*, *LDU*, *Cholesky* LDL^T
- Tridiagonalization $Q^T AQ = T$, A - symmetric, T - tridiagonal
  Aasen's algorithm: $A = LTL^T$
- Bidiagonalization $Q^T AV = B$, $A(m, n)$, B - upper bi-diagonal
- $\dots$
- *QR*

Golub, Van Loan, *Matrix Computations*, many editions.
Note: Some of the algorithms are not numerically stable.

Matrix factorizations

- **Eigenvalue decomposition**

A - square, normal matrix. If all its eigenvectors are linearly independent, then $A = QDQ^T$, where Q is orthogonal and D is diagonal, containing the eigenvalues of A.

Matrix factorizations

- **Eigenvalue decomposition**
A - square, normal matrix. If all its eigenvectors are linearly independent, then $A = QDQ^T$, where Q is orthogonal and D is diagonal, containing the eigenvalues of A.
- **Schur decomposition**: $A = QTQ^*$
Any square matrix A is unitary similar to an upper triangular matrix T, which has the eigenvalues of A on its diagonal.
Note: Eigenvalue-revealing factorization

Matrix factorizations

- **Eigenvalue decomposition**
A - square, normal matrix. If all its eigenvectors are linearly independent, then $A = QDQ^T$, where Q is orthogonal and D is diagonal, containing the eigenvalues of A.
- **Schur decomposition**: $A = QTQ^*$
Any square matrix A is unitary similar to an upper triangular matrix T, which has the eigenvalues of A on its diagonal.
Note: Eigenvalue-revealing factorization

- Question: Can we diagonalize a general $(m \times n)$ matrix using unitary matrices?

Singular value decomposition *SVD*

$$Q_1 A Q_2^* = \Sigma$$

Singular value decomposition

SVD

Let $A(m,n)$, $n \leq m$ or $n \geq m$, $rank(A) = rank(A^*)$.

Definition

If there exist $\mu \neq 0$ and vectors $\mathbf{u}$ and $\mathbf{v}$, such that

$$A\mathbf{v} = \mu\mathbf{u} \quad \text{and} \quad A^*\mathbf{u} = \mu\mathbf{v}$$

then μ is called a singular value of A, and $\mathbf{u}, \mathbf{v}$ are a pair of singular vectors, corresponding to μ.

Historical notes

SVD has many different names:

- First derivation of the SVD by Eugenio Beltrami (1873)
- Full proof by Camille Jordan (1874)
- James Joseph Sylvester (1889), independently discovers SVD
- Erhard Schmidt (1907), first to derive an optimal, low-rank approximation of a larger problem
- Hermann Weyl (1912) - determination of the rank in the presence of errors
- Eckart-Young decomposition and optimality properties of SVD (1936), psychometrics
- Numerically efficient algorithms to compute the SVD - works by Gene Golub 1970 (Golub-Kahan)

The existence of singular values and vectors ...

... is shown via the following construction:

$$A\mathbf{v} = \mu\mathbf{u}, \quad A^*\mathbf{u} = \mu\mathbf{v}$$

can be written as

$$\widetilde{A}\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix} = \begin{bmatrix}0 & A^*\\ A & 0\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix} = \mu\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix}$$

The matrix $\widetilde{A}$ is selfadjoint, has real eigenvalues and a complete eigenvector space.
Furthermore, μ^2 is an eigenvalue of A^*A with eigenvector $\mathbf{u}$ and of AA^* with eigenvector $\mathbf{v}$, because

$$\begin{aligned} A\mathbf{v} &= \mu\mathbf{u}, \quad \to \quad A^*A\mathbf{v} = \mu A^*\mathbf{u} = \mu^2\mathbf{v}\\ A^*\mathbf{u} &= \mu\mathbf{v}, \quad \to \quad AA^*\mathbf{u} = \mu A\mathbf{v} = \mu^2\mathbf{u}\end{aligned}$$

Singular Value Decomposition

Theorem (SVD)

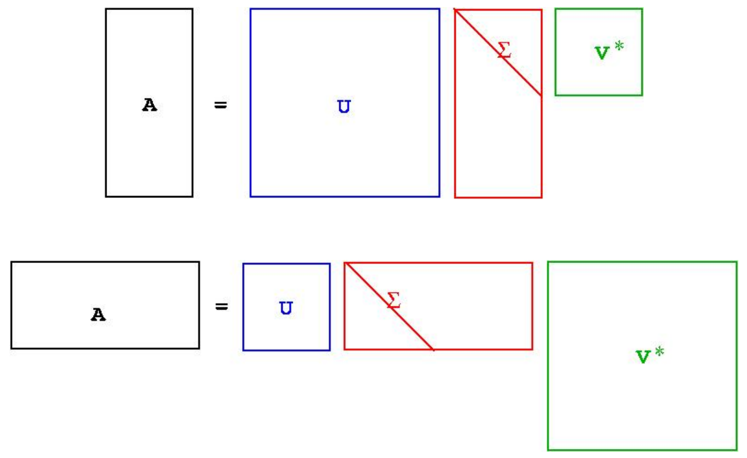
Any $m \times n$ matrix A with dimensions, say, $m \geq n$, can be factorized as

$$A = U\begin{bmatrix}\Sigma\\ 0\end{bmatrix}V^*,$$

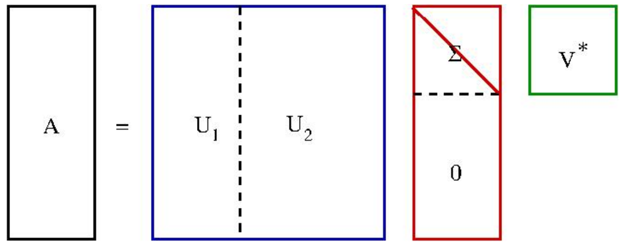
where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal,

$$\begin{aligned}\Sigma &= \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n),\\ \sigma_1 &\geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.\end{aligned}$$

SVD, $A(m \times n)$



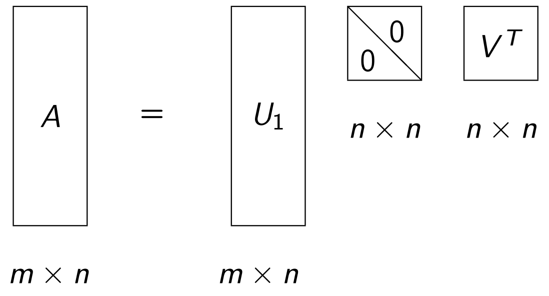
SVD, observation



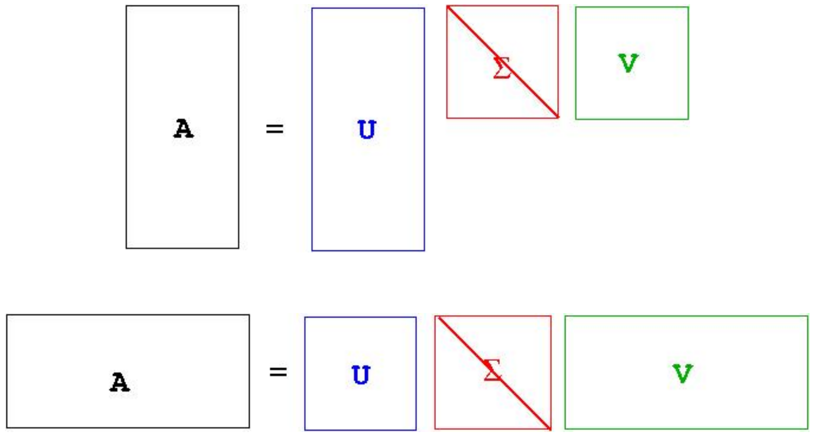
Thin SVD

Partition $U = (U_1\ U_2)$, where $U_1 \in \mathbb{R}^{m\times n}$,

$$A = U_1 \Sigma V^T,$$



Thin SVD



Fundamental Subspaces I

The *range of the matrix A:*

$$\mathcal{R}(A) = \{y \mid y = Ax, \text{ for arbitrary } x\}.$$

Assume that A has rank r:

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0.$$

Outer product form:

$$y = Ax = \sum_{i=1}^{r} \sigma_i u_i v_i^T x = \sum_{i=1}^{r} (\sigma_i v_i^T x) u_i = \sum_{i=1}^{r} \alpha_i u_i.$$

Fundamental Subspaces II

The *null-space of the matrix A*:

$$\mathcal{N}(A) = \{x \mid Ax = 0\}.$$

$$Ax = \sum_{i=1}^{r} \sigma_i u_i v_i^T x$$

Any vector $z = \sum_{i=r+1}^{n} \beta_i v_i$ is in the null-space:

$$Az = (\sum_{i=1}^{r} \sigma_i u_i v_i^T)(\sum_{i=r+1}^{n} \beta_i v_i) = 0.$$

Fundamental Subspaces

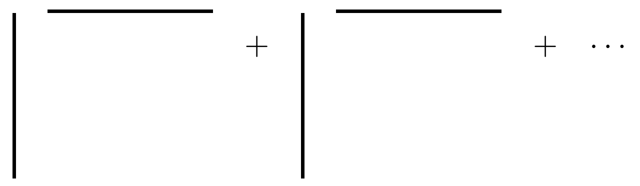
Theorem (Fundamental subspaces)

1. *The singular vectors $u_1, u_2, \ldots, u_r$ are an orthonormal basis in $\mathcal{R}(A)$ and*
$$\operatorname{rank}(A) = \dim(\mathcal{R}(A)) = r.$$
2. *The singular vectors $v_{r+1}, v_{r+2}, \ldots, v_n$ are an orthonormal basis in $\mathcal{N}(A)$ and*
$$\dim(\mathcal{N}(A)) = n - r.$$
3. *The singular vectors $v_1, v_2, \ldots, v_r$ are an orthonormal basis in $\mathcal{R}(A^T)$.*
4. *The singular vectors $u_{r+1}, u_{r+2}, \ldots, u_m$ are an orthonormal basis in $\mathcal{N}(A^T)$.*

SVD matrix expansion

$$A = U\Sigma V^T$$

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T =$$



A is represented as a sum of rank-one matrices.

Rank deficient matrix I

```
>> A(:,3)=A(:,1)+0.5*A(:,2)

A = 1.0000    1.0000    1.5000
    1.0000    2.0000    2.0000
    1.0000    3.0000    2.5000
    1.0000    4.0000    3.0000

>> [U,S,V]=svd(A,0)

U = 0.2612   -0.7948   -0.5000
    0.4032   -0.3708    0.8333
    0.5451    0.0533   -0.1667
    0.6871    0.4774   -0.1667
```

Rank deficient matrix II

```
S = 7.3944        0         0
         0   0.9072         0
         0        0         0

V = 0.2565  -0.6998    0.6667
    0.7372   0.5877    0.3333
    0.6251  -0.4060   -0.6667
```

SVD is rank-revealing!

Best approximation / Eckart-Young Property, 1936 I

2-norm, Frobenius norm

$$\|A\|_2 = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}, \quad \|A\|_F = \left(\sum_{i,j} a_{ij}^2\right)^{1/2}$$

One can show that $\|A\|_2 = \left(\max_{1\le i\le n} \lambda_i(A^TA)\right)^{\frac{1}{2}} = \sigma_1$

Best approximation / Eckart-Young Property, 1936 II

Theorem

Assume that the matrix $A \in \mathbb{R}^{m\times n}$ has rank r and choose k, such that $r > k$. The Frobenius norm matrix approximation problem

$$\min_{\mathrm{rank}(Z)=k} \|A - Z\|_F$$

has the solution

$$Z = A_k = U_k \Sigma_k V_k^T,$$

where $U_k = (u_1, \dots, u_k)$, $V_k = (v_1, \dots, v_k)$, and $\Sigma_k = \mathrm{diag}(\sigma_1, \dots, \sigma_k)$.

Best approximation / Eckart-Young Property, 1936 III

Theorem

Let $A \in C^{m\times n}$ have rank r and SVD $A = U\Sigma V^* = \sum_{i=1}^{r} \sigma_i u_i v_i^*$.

The solution of the problem

$$\min_X \|A - X\|_2$$

where X belongs to the set of matrices in $C^{m\times n}$ of rank $k < r$, is obtained for the matrix $X = A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^*$ and

$\|A - A_k\|_2 = \sigma_{k+1}$.

Best approximation / Eckart-Young Property, 1936 IV

Proof:
(1) Observe: $U^T A_k V = diag\{\sigma_1, \sigma_2, \cdots, \sigma_k, 0, \cdots, 0\}$, e.g. $rank(A_k) = k$.
$U^T(A - A_k)V = diag\{0, \cdots, 0, \sigma_{k+1}, \cdots, \sigma_r\}$, then $\|A - A_k\|_2 = \sigma_{k+1}$.

(2) Assume that there exists a matrix B of rank k, such that $\|A - B\|_2 < \sigma_{k+1}$.
$rank(B)) = k$, then there exists a basis $x_1, \cdots, x_{n-k}$ such that $(B) = \mathrm{span}\{x_1, \cdots, x_{n-k}\}$.
From dimension arguments
$\mathrm{span}\{x_1, \cdots, x_{n-k}\} \cap \mathrm{span}\{v_1 \cdots, v_{k+1}\} \neq \emptyset$.

Best approximation / Eckart-Young Property, 1936 V

Let z belong to the intersection. Then
$Az = \sum_{i=1}^{k+1} \sigma_i (v_1^T z) u_i$. Thus,

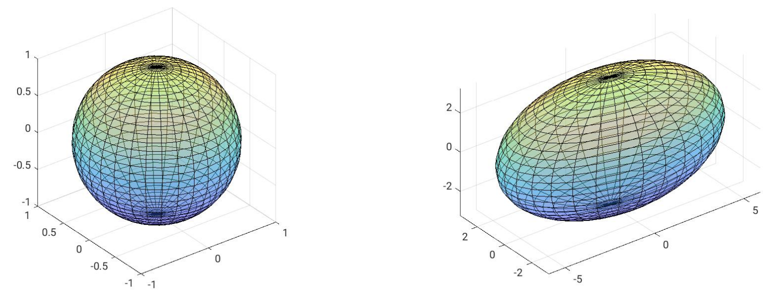
$$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2 = \sum_{i=1}^{k+1} \sigma_i (v_1^T z)^2 \geq \sigma_{k+1}.$$

The latter contradicts to the assumption.

Image compression, example

Matlab demo

SVD, geometric view



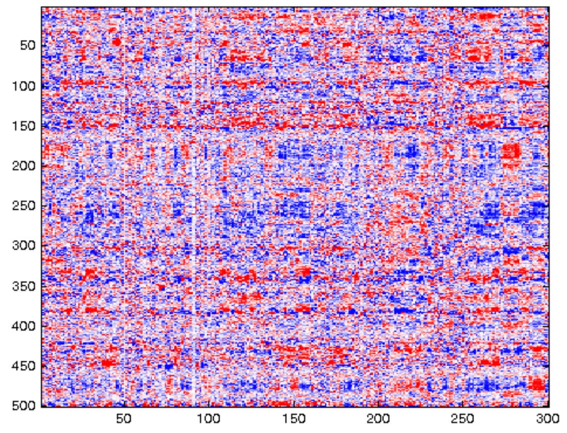
$$A = U\Sigma V^* \quad AV = U\Sigma$$

Example

Test example borrowed from
Computational Statistics with Application to Bioinformatics
Prof. William H. Press Spring Term, 2008, The University of Texas at Austin

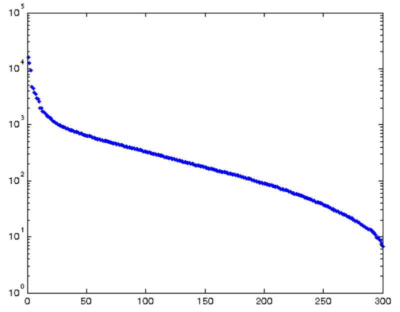
Consider some gene expression data, represented by the so-called 'design matrix' $X = \{X_{ij}\}$
Each column of X corresponds to a separate observation, in this case, a separate micro array experiment under a different condition.
N rows are genes (1:500) and M columns are the corresponding responses.
Assumptions:
- the individual experiments (columns of X) have zero mean.
- scale data to unit standard deviation.

```
load yeastarray_t2.txt;
size(yeastarray_t2)

ans = 500 300
yclip = prctile(yeastarray_t2(:),[1,99])
yclip = -204 244

data = max(yclip(1),min(yclip(2),yeastarray_t2));
dmean= mean(data,1);
dstd = std(data,1);
data = (data - repmat(dmean,[size(data,1),1]))./...
              repmat(dstd,[size(data,1),1]);
genecolormap = [min(1,(1:64)/32); 1-abs(1-(1:64)/32);
               min(1,(64-(1:64))/32)]';
figure(1),clf,colormap(genecolormap);
image(20*data+32)
```



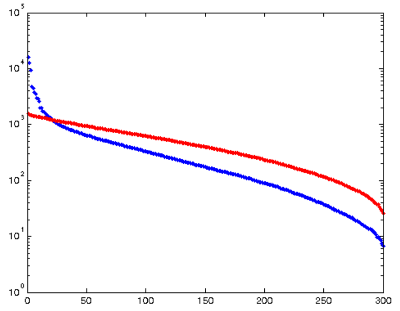
The squares of the singular values are proportional to the portion of the total variance (L_2 norm of X) that each accounts for.

```
ssq = diag(S).^2;
semilogy(ssq,'.b')
```



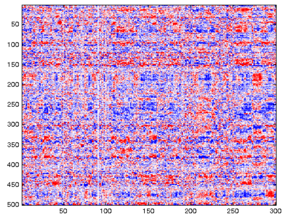
We can produce fake data and compare:

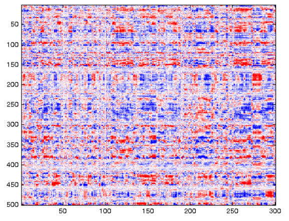
```
fakedata = randn(500,300);
[Uf Sf Vf] = svd(fakedata,0);
sfsq = diag(Sf).^2;
semilogy(sfsq,'.r')
```

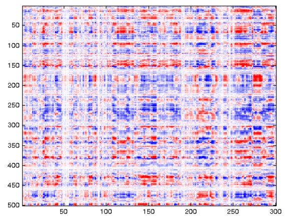


For the data in this example, a sensible use of PCA (i.e., SVD) would be to project the data into the subspace of the first 20 SVs, where we can be sure that it is not noise.

```
% Truncate the first 20 singular values/vectors
strunc = diag(S);
strunc(21:end) = 0;
filtdata20 = U*diag(strunc)*V';
figure(2),clf,colormap(genecolormap);
image(20*filtdata20+32)

% Truncate the first 5 singular values/vectors
strunc(6:end) = 0;
filtdata5 = U*diag(strunc)*V';
figure(3),clf,colormap(genecolormap);
image(20*filtdata5+32)
```

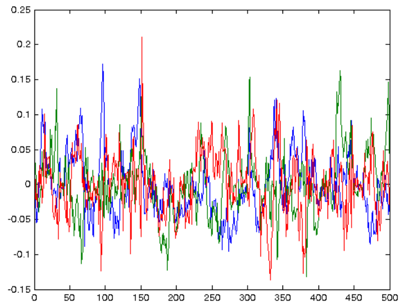

(a) The original


(b) truncate to 20



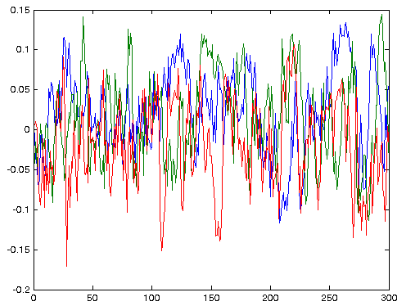
How to interpret the singular vectors? The first three vectors u are 'eigengenes', the linear combination of genes that explain the most data.

```
plot(U(:,1:3))
```



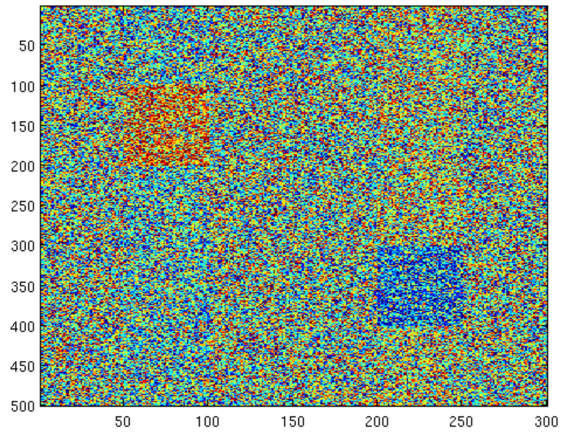
The first three vectors v are 'eigenarrays', the linear combination of experiments that explain the most data.

```
plot(V(:,1:3))
```



Consider a toy example

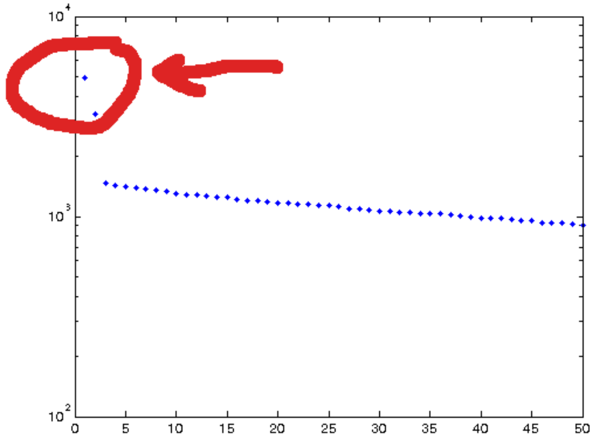
```
pdata = randn(500,300);
pdata(101:200,51:100) = pdata(101:200,51:100) + 1;
pdata(301:400,201:250) = pdata(301:400,201:250) - 1;
pmean = mean(pdata,1);
pstd = std(pdata,1);
pdata = (pdata - repmat(pmean,[size(pdata,1),1]))./...
                 repmat(pstd,[size(pdata,1),1]);
colormap(genecolormap)
image(20*pdata+32)
```



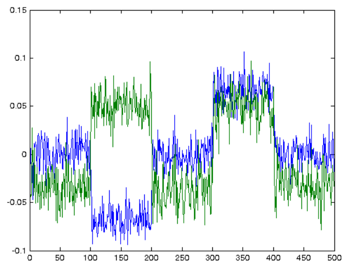
Consider a toy example

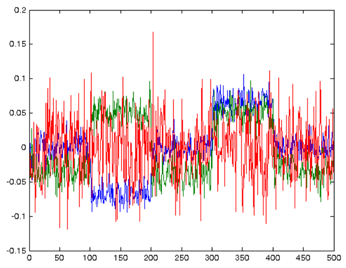
```
[Up Sp Vp] = svd(pdata,0);
spsq = diag(Sp).^2;
semilogy(spsq(1:50),'.b')
```

Should we expect the eigengenes/eigenarrays to show the separate main effects?

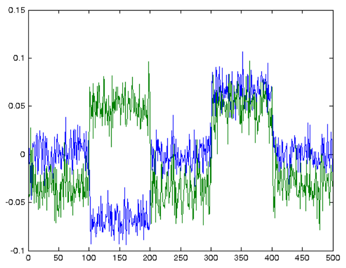


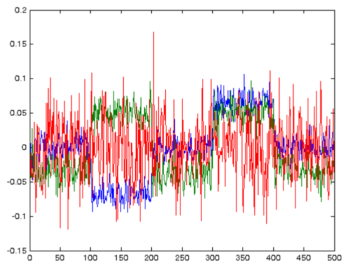
```
plot(Up(:,1:2)),
plot(Up(:,1:3))
```


(d)


(e)

```
plot(Vp(:,1:2)),
plot(Vp(:,1:3))
```


(f)


(g)

Linear dependence – SVD

Theorem

Let the singular values of A satisfy

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0.$$

Then the rank of A is equal to r.

Rank $=$ the number of linearly independent columns of A.

Linear dependence I

```
A=[1 1; 1 2; 1 3; 1 4]
singval=svd(A)

% Third col=linear combination of first two
A1=[A A(:,1)+0.5*A(:,2)]
singval1=svd(A1)
```

Linear dependence II

Result:

```
A =  1      1          singval = 5.7794
     1      2                    0.7738
     1      3
     1      4

A1 = 1.0000    1.0000    1.5000
     1.0000    2.0000    2.0000
     1.0000    3.0000    2.5000
     1.0000    4.0000    3.0000

singval1 = 7.3944
           0.9072
                0
```

Almost linear dependence I

```
A2=[A A(:,1)+0.5*A(:,2)+0.0001*randn(4,1)]
singval2=svd(A2)

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

A2 = 1.0000    1.0000    1.4999
     1.0000    2.0000    2.0001
     1.0000    3.0000    2.5000
     1.0000    4.0000    3.0001

singval2 = 7.3944
           0.9072
           0.0001
```

Almost linear dependence? I

Run Matlab demo
`~/.../STAT/Labs/Lab_QR_SVD/Small_singular_values.m`

How to compute the SVD? I

- Matlab:

```
[U,S,V] = svd(A); [U1,S,V]=svd(A,0);
```

- R:

```
(s <- svd(A))
s <- La.svd(A)
```

- python:

```
U, S, V = np.linalg.svd(A)
```

Computing the SVD in a numerically efficient way

Golub-Kahan (Golub-Kahan-Lanczos) bidiagonalization

$$A = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \to U_1^* A = \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \to U_1^* A V_1 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix}$$

Golub-Kahan (Golub-Kahan-Lanczos) bidiagonalization

$$U_4^* U_3^* U_2^* U_1^* A V_1 V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus, $U^* AV = B$, B - bidiagonal.

Golub-Kahan (Golub-Kahan-Lanczos) bidiagonalization

```
B =  1     2     0     0
     0     3     4     0
     0     0     5     6
     0     0     0     7

H=[0  B],   p=[5 1 6 2 7 3 8 4]
   B'  0]

H(p,p)=
           1
     1           2
           2           3
                 3           4
                       4           5
                             5           6
                                   6           7
                                         7
```

- How much does it cost to compute the SVD?
  The cost for the bidiagonalization is $4mn^2 - 4/3n^3$.
  The cost for SVD: $4m^2n + 8mn^2 + 9n^3$.

- How much does it cost to compute the SVD?
  The cost for the bidiagonalization is $4mn^2 - 4/3n^3$.
  The cost for SVD: $4m^2n + 8mn^2 + 9n^3$.

- How much does it cost to LU-factorize a full matrix?

- How much does it cost to compute the SVD?
  The cost for the bidiagonalization is $4mn^2 - 4/3n^3$.
  The cost for SVD: $4m^2n + 8mn^2 + 9n^3$.
- How much does it cost to LU-factorize a full matrix?
- And how much does it take to LU-factorize a sparse matrix?

Applications of SVD

Applications of SVD I

- Data compression
- Estimation/inversion
- Sensitivity of linear equations to data errors
  The condition number of A is defined as $\varkappa(A) = \sigma_{max}/\sigma_{min}$.
- Low rank approximation
- Image processing
- Distance to singularity Another interpretation of σ_i:

$$\sigma_i = \min\{\|A - B\|, s.t.rank(B) \leq i - 1\},$$

  i.e., σ_i is the distance, measured by the matrix norm, to the nearest rank $i - 1$ matrix.

Applications of SVD II

- General pseudoinverse Let $A = U\Sigma V^T$. Then

$$A^\dagger = V\Sigma^{-1}U^T$$

  is the so-called Moore-Penrose inverse of A.
  Let $A(m, n), m > n$. Then $A^\dagger = (A^TA)^{-1}A^T$ gives the least square solution of $Ax = y$, $x = A^\dagger y$.
  Let $m < n$. Then $A^\dagger = A^T(A^TA)^{-1}$ gives the least-norm solution of $Ax = y$

Applications of SVD, web searching

"Search engines like Google use enormous matrices of cross-references, which pages link to which other pages, and what words are on each page. When you do a Google search, the higher ranks usually go to pages with your key words that have lots of links to them. But there are billions of pages out there, and storing a billion by billion matrix is trouble, not to mention searching through it.
In searching, one only cares about the main directions that the Web is taking. So the first few singular values create a very good approximation for the enormous matrix, can be searched relatively quickly (just a few billion entries) and provide compression ratios of millions to one. "

LS via SVD

$$\|Ax-b\|_2^2 = \|U^T(AVV^Tx-b)\|_2^2 = \|\Sigma(V^Tx)-U^Tb\|_2^2$$
$$= \sum_{i=1}^{r}(\sigma_i z_i - u_i^T b)^2 + \sum_{i=r+1}^{m}(u_i^T b)^2.$$

Thus,

$$\min \|Ax-b\|_2^2 = \underbrace{\sum_{i=1}^{r}(\sigma_i z_i - u_i^T b)^2}_{\textit{set to } 0} + \underbrace{\sum_{i=r+1}^{m}(u_i^T b)^2}_{\textit{does not depend on } x}$$