LECTURE 4: LINEAR ALGEBRA

- Linear algebra is used to solve linear problems of different nature
- It does so by doing matrix decompositions
- Three main linear algebra decompositions are
- 1. Singular Value Decomposition $\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T$
  2. Gram-Schmidt Orthogonalization $\mathbf{A} = \mathbf{Q}\,\mathbf{R}$
  3. Gaussian Elimination $\mathbf{A} = \mathbf{L}\mathbf{U}$
- Linear problems are the basis of data science: often one gets good intuition by understanding linear problem first

How to solve linear system of equations?

- We wish to solve for a linear system of equations.
  For now assume equal number of equations as variables N.

$$\begin{aligned} a_{11}\ x_1 &+ a_{12}\ x_2 + \cdots a_{1m}\ x_n = d_1 \\ a_{21}\ x_1 &+ a_{22}\ x_2 + \cdots a_{2m}\ x_n = d_2 \\ \vdots & \quad\ \vdots \quad\quad\ \vdots \quad\ \vdots \\ a_{n1}\ x_1 &+ a_{n2}\ x_2 + \cdots a_{nm}\ x_n = d_n \end{aligned} \quad \rightarrow \quad \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}$$

$Ax = b$: formal solution is $x = A^{-1}b$; A is NxN matrix.
Matrix inversion is very slow.

Gaussian Elimination

- $Ax = b$
- Multiply any row of A by any constant, and do the same on b
- Take linear combination of two rows, adding or subtracting them, and the same on b
- We can keep performing these operations until we set all elements of first column to 0 except 1st one, which can be set to 1
- Then we can repeat the same to set to 0 all elements of 2nd column, except first 2…
- We end up with an upper diagonal matrix with 1 on the diagonal

Gaussian Elimination

$$\begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & -1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix} \blacktriangleright \begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & -1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

$$\blacktriangledown$$

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 2.5 & -7 & -2.5 \\ 0 & -4.5 & 1 & 4.5 \\ 0 & -3 & -3 & 2 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 9 \\ 11 \\ 11 \end{pmatrix} \blacktriangleleft \begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 2.5 & -7 & -2.5 \\ 1 & -4 & -1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 9 \\ 9 \\ 7 \end{pmatrix}$$

$$\blacktriangledown$$

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 1 & -2.8 & -1 \\ 0 & -4.5 & 1 & 4.5 \\ 0 & -3 & -3 & 2 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 3.6 \\ 11 \\ 11 \end{pmatrix} \blacktriangleright \begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 1 & -2.8 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 3.6 \\ -2 \\ 1 \end{pmatrix}$$

Backsubstitution

$$\begin{pmatrix} 1 & 0.5 & 2 & 0.5 \\ 0 & 1 & -2.8 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 3.6 \\ -2 \\ 1 \end{pmatrix}$$

$$1 \cdot z = 1 \;\rightarrow\; z = 1$$

$$1 \cdot y + 0 \cdot z = -2 \;\rightarrow\; y = -2$$

$$1 \cdot x + -2.8 \cdot y - 1 \cdot z = 3.6 \;\rightarrow\; x = -1$$

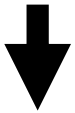
$$1 \cdot w + 0.5 \cdot x + 2 \cdot y + 0.5 \cdot z = -2 \;\rightarrow\; w = 2$$

Back just means we start at the bottom and move up

Pivoting

- What if the first element is 0?
- Swap the rows (partial pivoting) or rows and columns (full pivoting)!
- In practice simply pick the largest element (keep in mind this changes det sign)

$$\begin{pmatrix} 0 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$



$$\begin{pmatrix} 3 & 4 & -1 & -1 \\ 0 & 1 & 4 & 1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ -4 \\ 9 \\ 7 \end{pmatrix}$$

LU Decomposition

- We want to solve $Ax = b$ varying b, so we'd like to have a decomposition of A that is done once and then can be applied to several b
- Suppose we can write $\boldsymbol{A = LU}$, where L is lower diagonal and U is upper diagonal: $L(Ux) = b$
- Then we can first solve for y in $Ly = b$ using forward substitution, followed by $Ux = y$ using backward substitution
- Operation count N^3, meaning that the number of basic operations (additions, multiplications etc) scales as N^3. This is a very steep count: change N by 10 and number of operations goes up by 1000

Revisit Gauss Elimination: Make A an upper triangular matrix.

- $L_0 A = A'$: try to make one column of A' to have (1,0,0,0). Solution:

$$\underbrace{\frac{1}{a_{00}}\begin{pmatrix} 1 & 0 & 0 & 0 \\ -a_{10} & a_{00} & 0 & 0 \\ -a_{20} & 0 & a_{00} & 0 \\ -a_{30} & 0 & 0 & a_{00} \end{pmatrix}}_{L_0 \text{ lower triangular}} \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & b_{01} & b_{02} & b_{03} \\ 0 & b_{11} & b_{12} & b_{13} \\ 0 & b_{21} & b_{22} & b_{23} \\ 0 & b_{31} & b_{32} & b_{33} \end{pmatrix}$$

- $L_1 A' = A''$

$$\underbrace{\frac{1}{b_{11}}\begin{pmatrix} b_{11} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -b_{21} & b_{11} & 0 \\ 0 & -b_{31} & 0 & b_{11} \end{pmatrix}}_{L_1 \text{ lower triangular}} \begin{pmatrix} 1 & b_{01} & b_{02} & b_{03} \\ 0 & b_{11} & b_{12} & b_{13} \\ 0 & b_{21} & b_{22} & b_{23} \\ 0 & b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} 1 & c_{01} & c_{02} & c_{03} \\ 0 & 1 & c_{12} & c_{13} \\ 0 & 0 & c_{22} & c_{23} \\ 0 & 0 & c_{32} & c_{33} \end{pmatrix}$$

Get L_2, L_3 similarly

$$L_3 L_2 L_1 L_0 \, A \, x = L_3 L_2 L_1 L_0 b$$

Lower triangular

Upper triangular

Define $L \equiv L_0^{-1} L_1^{-1} L_2^{-1} L_3^{-1}, \quad U \equiv L_3 L_2 L_1 L_0 A$

$$\rightarrow LU = A$$

- $L_3 L_2 L_1 L_0 A$ (=U) is upper diagonal. It gives: $\begin{pmatrix} 1 & \# & \# & \# \\ 0 & 1 & \# & \# \\ 0 & 0 & 1 & \# \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- L is lower diagonal because: $L_0 = \frac{1}{a_{00}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -a_{10} & a_{00} & 0 & 0 \\ -a_{20} & 0 & a_{00} & 0 \\ -a_{30} & 0 & 0 & a_{00} \end{pmatrix} \rightarrow L_0^{-1} = \frac{1}{a_{00}} \begin{pmatrix} a_{00} & 0 & 0 & 0 \\ a_{10} & 1 & 0 & 0 \\ a_{20} & 0 & 1 & 0 \\ a_{30} & 0 & 0 & 1 \end{pmatrix}$

$$L = L_0^{-1} L_1^{-1} L_2^{-1} L_3^{-1} = \frac{1}{a_{00}} \begin{pmatrix} a_{00} & 0 & 0 & 0 \\ a_{10} & b_{11} & 0 & 0 \\ a_{20} & b_{21} & c_{22} & 0 \\ a_{30} & b_{31} & c_{32} & d_{33} \end{pmatrix}$$

We have $N(N+1)/2$ components for L and $N(N-1)/2$ for U because $U_{ii}=1$.
So in total N^2 as in A

What if the matrix A is symmetric and positive definite?

- $A_{12}=A_{21}$ hence we can set $U=L^T$, so the LU decomposition is $A=LL^T$
- Symmetric matrix has $N(N+1)/2$ elements, same as L
- This is the fastest way to solve such a matrix and is called Cholesky decomposition (still N^3)
- Pivoting is needed for LU, while Cholesky is stable even without pivoting
- Use numpy.linalg import solve
- L can be viewed as a square root of A, but this is not unique (discussed later)

Inverse and Determinant

- $AX = I$ and solve with LU (use inv in linalg)
- det $A = L_{00}L_{11}L_{22}\ldots$ (note that $U_{ii} = 1$) times number of row permutations (if pivoting)
- Can be very large: better to compute $ln\ detA = lnL_{00} + lnL_{11} + \ldots$

Tridiagonal and Banded Matrices

- Solved with Gaussian substitution: $O(N)$ instead of N^3 in CPU, N instead of N^2 in storage

$$A = \begin{pmatrix} a_{00} & a_{01} & 0 & 0 & 0 \\ a_{10} & a_{11} & a_{12} & 0 & 0 \\ 0 & a_{21} & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{43} & a_{44} \end{pmatrix}$$ Tridiagonal

E.g.

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 3 & 4 & -5 & 0 \\ 0 & -4 & 3 & 5 \\ 0 & 0 & 1 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0.5 & 0 & 0 \\ 3 & 4 & -5 & 0 \\ 0 & -4 & 3 & 5 \\ 0 & 0 & 1 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 2.5 & -5 & 0 \\ 0 & -4 & 3 & 5 \\ 0 & 0 & 1 & 3 \end{pmatrix}$$

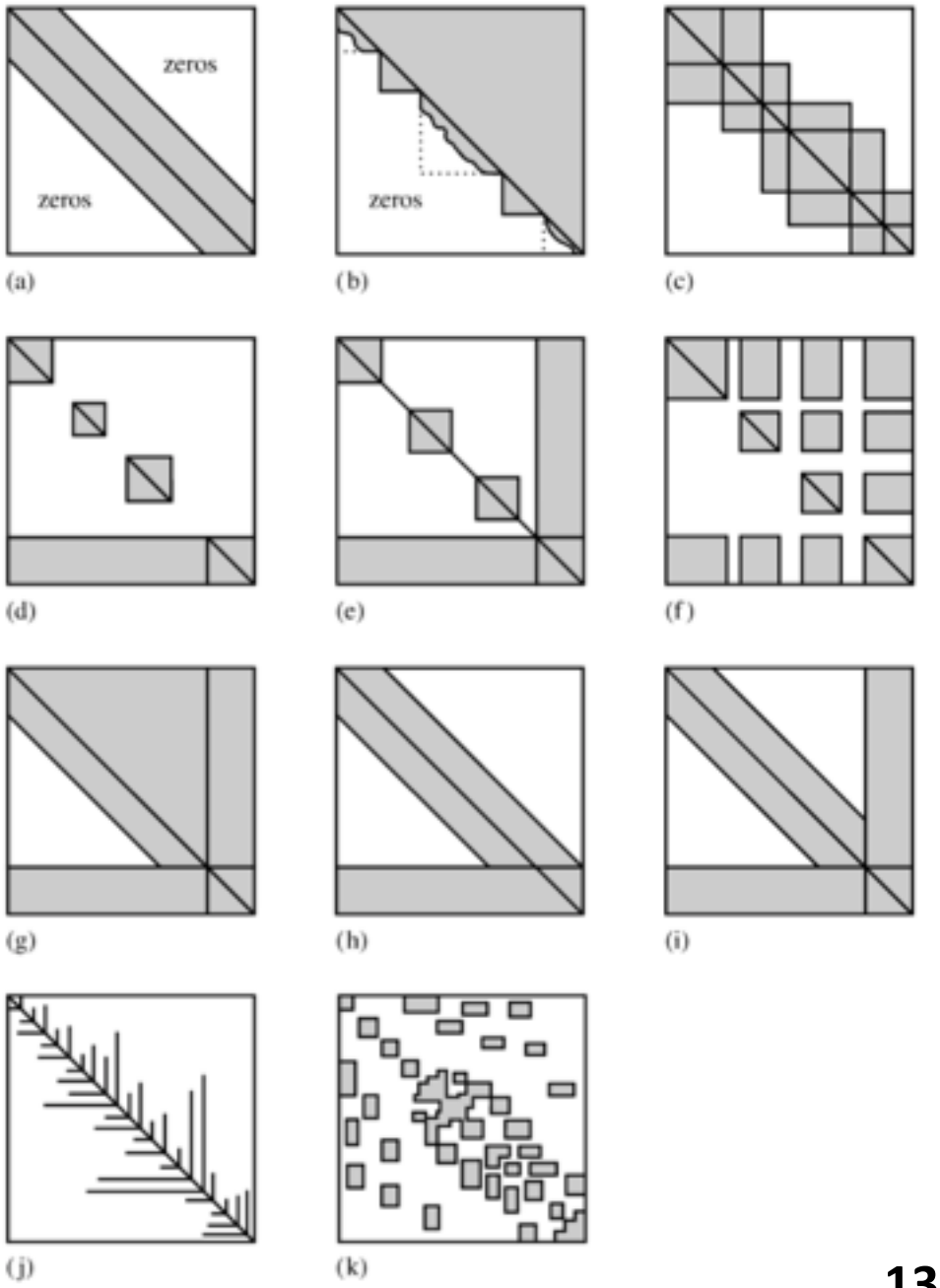
$$\downarrow$$

$$\begin{pmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 4 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & -5 & 5 \\ 0 & 0 & 1 & 3 \end{pmatrix}$$

Same approach can be used for banded matrices

General sparse matrices

- Allow solutions to scale faster than N^3
- No single algorithm
- General advice: be aware that such matrices can have much faster solutions, so when you encounter them check that the routine takes advantage of this



Credit: NR, Press etal, chapter 2.7

Sherman-Morrison Formula

- If we have a matrix A we can solve (e.g. tridiagonal etc) and we can add rank 1 component then we can get A^{-1} in $3N^2$:

$$\mathbf{A} \rightarrow (\mathbf{A} + \mathbf{u} \otimes \mathbf{v})$$

$$\begin{aligned}(\mathbf{A} + \mathbf{u} \otimes \mathbf{v})^{-1} &= (\mathbf{1} + \mathbf{A}^{-1} \cdot \mathbf{u} \otimes \mathbf{v})^{-1} \cdot \mathbf{A}^{-1} \\ &= (\mathbf{1} - \mathbf{A}^{-1} \cdot \mathbf{u} \otimes \mathbf{v} + \mathbf{A}^{-1} \cdot \mathbf{u} \otimes \mathbf{v} \cdot \mathbf{A}^{-1} \cdot \mathbf{u} \otimes \mathbf{v} - \ldots) \cdot \mathbf{A}^{-1} \\ &= \mathbf{A}^{-1} - \mathbf{A}^{-1} \cdot \mathbf{u} \otimes \mathbf{v} \cdot \mathbf{A}^{-1} (1 - \lambda + \lambda^2 - \ldots) \\ &= \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1} \cdot \mathbf{u}) \otimes (\mathbf{v} \cdot \mathbf{A}^{-1})}{1 + \lambda} \end{aligned} \tag{2.7.2}$$

where

$$\lambda \equiv \mathbf{v} \cdot \mathbf{A}^{-1} \cdot \mathbf{u} \tag{2.7.3}$$

$$\mathbf{z} \equiv \mathbf{A}^{-1} \cdot \mathbf{u} \qquad \mathbf{w} \equiv (\mathbf{A}^{-1})^T \cdot \mathbf{v} \qquad \lambda = \mathbf{v} \cdot \mathbf{z} \tag{2.7.4}$$

to get the desired change in the inverse

$$\mathbf{A}^{-1} \rightarrow \mathbf{A}^{-1} - \frac{\mathbf{z} \otimes \mathbf{w}}{1 + \lambda} \tag{2.7.5}$$

Example: cyclic tridiagonal systems

- This happens for finite difference differential equations with periodic boundary conditions

$$\begin{bmatrix} b_0 & c_0 & 0 & \cdots & & & \beta \\ a_1 & b_1 & c_1 & \cdots & & & \\ & & & \cdots & & & \\ & & & \cdots & a_{N-2} & b_{N-2} & c_{N-2} \\ \alpha & & & \cdots & 0 & a_{N-1} & b_{N-1} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ \cdots \\ r_{N-2} \\ r_{N-1} \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} \gamma \\ 0 \\ \vdots \\ 0 \\ \alpha \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \beta/\gamma \end{bmatrix} \qquad b'_0 = b_0 - \gamma, \qquad b'_{N-1} = b_{N-1} - \alpha\beta/\gamma$$

$\mathbf{A} \rightarrow (\mathbf{A} + \mathbf{u} \otimes \mathbf{v})$ where A is tridiagonal

Changing the scaling from N^3 to N^2 is worth doing

Generalization: Woodbury formula

- Successive application of Sherman-Morrison to rank P, with $P << N$

$$(\mathbf{A} + \mathbf{U} \cdot \mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \left[\mathbf{A}^{-1} \cdot \mathbf{U} \cdot (\mathbf{1} + \mathbf{V}^T \cdot \mathbf{A}^{-1} \cdot \mathbf{U})^{-1} \cdot \mathbf{V}^T \cdot \mathbf{A}^{-1}\right]$$

- U and V are now NxP matrices
- Proof same as for Sherman-Morrison

Inversion by Partitioning

- Sometimes we can decompose the matrix into block sub-matrices P (dimension p x p), S (s x s) and Q and R (p x s and s x p)

$$\mathbf{A} = \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix} \qquad \mathbf{A}^{-1} = \begin{bmatrix} \tilde{\mathbf{P}} & \tilde{\mathbf{Q}} \\ \tilde{\mathbf{R}} & \tilde{\mathbf{S}} \end{bmatrix}$$

$$\tilde{\mathbf{P}} = (\mathbf{P} - \mathbf{Q} \cdot \mathbf{S}^{-1} \cdot \mathbf{R})^{-1}$$
$$\tilde{\mathbf{Q}} = -(\mathbf{P} - \mathbf{Q} \cdot \mathbf{S}^{-1} \cdot \mathbf{R})^{-1} \cdot (\mathbf{Q} \cdot \mathbf{S}^{-1})$$
$$\tilde{\mathbf{R}} = -(\mathbf{S}^{-1} \cdot \mathbf{R}) \cdot (\mathbf{P} - \mathbf{Q} \cdot \mathbf{S}^{-1} \cdot \mathbf{R})^{-1}$$
$$\tilde{\mathbf{S}} = \mathbf{S}^{-1} + (\mathbf{S}^{-1} \cdot \mathbf{R}) \cdot (\mathbf{P} - \mathbf{Q} \cdot \mathbf{S}^{-1} \cdot \mathbf{R})^{-1} \cdot (\mathbf{Q} \cdot \mathbf{S}^{-1})$$

$$\det \mathbf{A} = \det \mathbf{P} \det(\mathbf{S} - \mathbf{R} \cdot \mathbf{P}^{-1} \cdot \mathbf{Q}) = \det \mathbf{S} \det(\mathbf{P} - \mathbf{Q} \cdot \mathbf{S}^{-1} \cdot \mathbf{R})$$

- Worth doing if submatrix inversions can be fast, otherwise less obvious

Vandermode and Toeplitz Matrices

- Can be solved with N^2
- Vandermonde

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{N-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{N-1} & x_{N-1}^2 & \cdots & x_{N-1}^{N-1} \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix}$$

- Toeplitz

$$\begin{bmatrix} R_0 & R_{-1} & R_{-2} & \cdots & R_{-(N-2)} & R_{-(N-1)} \\ R_1 & R_0 & R_{-1} & \cdots & R_{-(N-3)} & R_{-(N-2)} \\ R_2 & R_1 & R_0 & \cdots & R_{-(N-4)} & R_{-(N-3)} \\ \cdots & & & \cdots & & \\ R_{N-2} & R_{N-3} & R_{N-4} & \cdots & R_0 & R_{-1} \\ R_{N-1} & R_{N-2} & R_{N-3} & \cdots & R_1 & R_0 \end{bmatrix}$$

Just remember their structure in case you encounter them

Summary

- Linear algebra allows us to solve linear systems of equations, compute inverse and determinant of a matrix…
- N^3 scaling is very steep: we cannot do it above $N = 10^4$-10^5
- If one can reduce it to N^2 it helps a lot
- For larger dimensions iterative methods are needed: these will be discussed when we discuss optimization

Matrix Diagonalization, Singular Value Decomposition and Principal Component Analysis

- We wish to diagonalize a square NxN matrix $\mathbf{A}$: $\mathbf{Ax} = \lambda\mathbf{x}$.
  Here $\mathbf{x}$ is the eigenvector and λ the eigenvalue of $\mathbf{A}$
- We could solve $\det|\mathbf{A}-\lambda\mathbf{I}|=0$ and expand it terms of a polynomial of N-th order in λ
- There is a shift symmetry: $(\mathbf{A}+\tau\mathbf{I})\mathbf{x}=(\lambda+\tau)\mathbf{x}$ that does not change eigenvectors, but changes eigenvalues

Common Matrix Types

- Symmetric: $\mathbf{A}=\mathbf{A}^T$ or $a_{ij}=a_{ji}$
- Hermitian (self-adjoint): $\mathbf{A}=\mathbf{A}^+$ or $a_{ij}=a_{ji}^*$. Important concept because eigenvalues are real (quantum mechanics!)
- Orthogonal: transpose equals inverse $\mathbf{A}\mathbf{A}^T=\mathbf{A}^T\mathbf{A}=\mathbf{I}$
- Unitary: Hermitian conjugate equals inverse $\mathbf{A}\mathbf{A}^+=\mathbf{I}$
- Normal: commutes with Hermitian conjugate $\mathbf{A}\mathbf{A}^+=\mathbf{A}^+\mathbf{A}$. Important because its eigenvectors are complete and orthogonal.
- For real matrices Hermitian is symmetric and unitary is orthogonal, both of which are normal.
- A normal non-symmetric matrix is hard to diagonalize.
- Symmetric matrices are easier to diagonalize.

Matrix Forms and Similarity Transform

- We can define the matrix of right eigenvectors as $\mathbf{X}_R$ and the matrix of eigenvalues as $\mathbf{D}$, which is diagonal with λ_i on the diagonal: $\mathbf{A}\mathbf{X}_R = \mathbf{X}_R\mathbf{D}$
- We can also define left eigenvectors $\mathbf{X}_L\mathbf{A}=\mathbf{D}\mathbf{X}_L$
- $\mathbf{X}_L\mathbf{X}_R\mathbf{D} = \mathbf{D}\mathbf{X}_L\mathbf{X}_R$ implies $\mathbf{X}_L\mathbf{X}_R = \mathbf{I}$ (with appropriate normalization) and so $\mathbf{A} = \mathbf{X}_R\mathbf{D}\mathbf{X}_L = \mathbf{X}_R\mathbf{D}\mathbf{X}_R^{-1}$ and $\mathbf{D} = \mathbf{X}_L\mathbf{A}\mathbf{X}_R = \mathbf{X}_R^{-1}\mathbf{A}\mathbf{X}_R$
- For a symmetric matrix $\mathbf{A}=\mathbf{X}_R\mathbf{D}\mathbf{X}_R^T$,so no need to distinguish L and R (inverse is hard, transpose is easy!)
- A transformation $\mathbf{Z}^{-1}\mathbf{A}\mathbf{Z}$ leaves eigensystem unchanged:
  $\det|\mathbf{Z}^{-1}\mathbf{A}\mathbf{Z}-\lambda\mathbf{I}| = \det|\mathbf{Z}^{-1}\mathbf{A}\mathbf{Z}-\lambda\mathbf{Z}^{-1}\lambda\mathbf{Z}| =$
  $\det|\mathbf{Z}^{-1}|\,\det|\mathbf{A}-\lambda\mathbf{I}|\,\det|\mathbf{Z}| = \det|\mathbf{A}-\lambda\mathbf{I}|$
- Many of the methods consist of a series of such transforms
- For real symmetric matrices $\mathbf{Z}^{-1}=\mathbf{Z}^T$

QR Decomposition

A: a set of **N** column vectors a_0, …, a_{N-1}

$$A = \begin{pmatrix} | & | & | & \dots & | \\ \mathbf{a_0} & \mathbf{a_1} & \mathbf{a_2} & \dots & a_{N-1} \\ | & | & | & \dots & | \end{pmatrix}$$

Define:

$$u_0 = a_0, \quad q_0 = \frac{u_0}{|u_0|}$$

$$u_1 = a_1 - (q_0 \cdot a_1) q_0, \quad q_1 = \frac{u_1}{|u_1|}$$

$$u_2 = a_2 - (q_0 \cdot a_2) q_0 - (q_1 \cdot a_2) q_1, \quad q_2 = \frac{u_2}{|u_2|}$$

$$\vdots$$

$$u_i = a_i - \sum_{j=0}^{i-1} (q_j \cdot a_i) q_j, \quad q_i = \frac{u_i}{|u_i|}$$

Each vector q_i is orthogonal to all previous ones and normalized

We can show (by induction) that $q_i q_j = \delta_{ij}$

Note that this is closely related to Gram-Schmidt orthogonalization

Gram-Schmidt orthogonalization and QR/RQ Decomposition

- The RQ decomposition transforms a matrix **A** into the product of an upper triangular matrix **R** (also known as right-triangular) and an orthogonal matrix **Q**. The only difference from QR decomposition is the order of these matrices.
- QR decomposition is Gram–Schmidt orthogonalization of columns of **A**, started from the first column.
- RQ decomposition is Gram–Schmidt orthogonalization of rows of **A**, started from the last row.

QR Decomposition

Let us reverse the process of q construction

$$u_0 = a_0, \quad q_0 = \frac{u_0}{|u_0|}$$
$$u_1 = a_1 - (q_0 \cdot a_1) q_0, \quad q_1 = \frac{u_1}{|u_1|}$$
$$u_2 = a_2 - (q_0 \cdot a_2) q_0 - (q_1 \cdot a_2) q_1, \quad q_2 = \frac{u_2}{|u_2|}$$
$$\vdots$$
$$u_i = a_i - \sum_{j=0}^{i-1} (q_j \cdot a_i) q_j, \quad q_i = \frac{u_i}{|u_i|}$$

$$a_0 = |u_0| q_0, \quad a_1 = |u_1| q_1 + (q_0 \cdot a_1) q_0$$
$$a_2 = |u_2| q_2 + (q_0 \cdot a_2) q_0 + (q_1 \cdot a_2) q_1 \ldots$$

$$\mathbf{A} = \begin{pmatrix} | & | & | & \ldots \\ | & | & | & \ldots \\ \mathbf{a_0} & \mathbf{a_1} & \mathbf{a_2} & \ldots \\ | & | & | & \ldots \end{pmatrix} = \underbrace{\begin{pmatrix} | & | & | & \ldots \\ | & | & | & \ldots \\ \mathbf{q_0} & \mathbf{q_1} & \mathbf{q_2} & \ldots \\ | & | & | & \ldots \end{pmatrix}}_{\mathbf{Q}} \cdot \underbrace{\begin{pmatrix} |u_0| & q_0 \cdot a_1 & q_0 \cdot a_2 & \ldots \\ 0 & |u_1| & q_1 \cdot a_2 & \ldots \\ 0 & 0 & |u_1| & \ldots \\ \vdots & \vdots & \vdots & \ldots \end{pmatrix}}_{\mathbf{R}}$$

- We have obtained a decomposition into an orthogonal matrix **Q** and upper diagonal matrix **R**

QR Decomposition for symmetric A: matrix diagonalization

- We can repeat these QR steps

$$A = Q_1 R_1, \quad Q_1^T A = Q_1^T Q_1 R_1 = R_1$$

$$A_1 \equiv R_1 Q_1 = Q_1^T A Q_1$$ Similarity Transform

Repeat

$$A_1 = Q_2 R_2, \quad A_2 \equiv R_2 Q_2 = Q_2^T A_1 Q_2 = Q_2^T Q_1^T A Q_1 Q_2$$

$$A_3 = Q_3^T Q_2^T Q_1^T A Q_1 Q_2 Q_3$$

$$\vdots$$

A_k converges to **D**

$$X_R = Q_1 Q_2 Q_3 ... Q_k$$

$$D = X_R^T A X_R \;\rightarrow\; A X_R = X_R D$$

General Packaged Solutions

- QR works better if the matrix **A** is first transformed into tridiagonal form (Hauseholder or Givens method): standard method for symmetric **A**
- Diagonalization methods: there are many different methods depending on what matrix we have: **A** can be real banded symmetric, real symmetric, complex Hermitian, real non-symmetric…
- Depending of what we need: just the eigenvectors or both eigenvectors and eigenvalues
- Look at numpy.linalg and choose depending on what your needs are: eigh, eigvalsh…
- Summary: numerical diagonalization is complicated
- Ask yourself: do you need it (e.g. "matrix square root")?

Generalized Eigenvalue Problems

- $\mathbf{Ax} = \lambda\mathbf{Bx}$ can be handled as $(\mathbf{B}^{-1}\mathbf{A})\mathbf{x} = \lambda\mathbf{x}$
- If **A** and **B** symmetric $\mathbf{B}^{-1}$ is not, but we can use Cholesky decomposition $\mathbf{B} = \mathbf{L}\mathbf{L}^T$ and by multiplying with $\mathbf{L}^{-1}$ we get $\mathbf{C}(\mathbf{L}^T\mathbf{x}) = \lambda\mathbf{L}^T\mathbf{x}$ where $\mathbf{C} = \mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^{-1})^T$ is a symmetric matrix. This diagonalization gives the same eigenvalues with eigenvectors given by $\mathbf{L}^T\mathbf{x}$.
- Nonlinear problems: $(\mathbf{A}\lambda^2+\mathbf{B}\lambda+\mathbf{C})\mathbf{x} = 0$. Add $\mathbf{y}=\lambda\mathbf{x}$ which can be solved by solving $2N$x$2N$ problem

$$\begin{pmatrix} 0 & I \\ -A^{-1}C & -A^{-1}B \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}$$

"Shortcomings" of Diagonalization

- Eigenvectors are not orthogonal for a non-symmetric *N*x*N* matrix
- There may not be enough eigenvectors, i.e. they may not be complete: **A** is "defective"
- It cannot be defined as eigenvalue problem $\mathbf{Ax}=\lambda\mathbf{x}$ unless **A** is *N*x*N*
- All these "defects" can be cured by SVD
- None of these "defects" apply to a symmetric *N*x*N* matrix: SVD and diagonalization are the same in that case if **A** is semi-positive definite

Singular Value Decomposition (SVD)

- Is a decomposition of a general *N*x*M* matrix $\mathbf{A} = \mathbf{U\Sigma V}^T$
- SVD deals with singular matrix problems: it identifies them and offers lowest norm solution: we will start with this motivation
- Define rank *r* of matrix **A** as the maximum number of linearly independent column vectors in the **matrix** or the maximum number of linearly independent row vectors in the **matrix** (both **definitions** are equivalent)
- Another way: define range as the subspace of **b** reached by **Ax** = **b**. Rank is its dimension.
- For *NxN*, if rank r<*N* then matrix is singular
- If the matrix is singular we can have two options: if b is in range then SVD can offer the solution closest to 0
- If b is outside the range SVD finds the solution closest to it

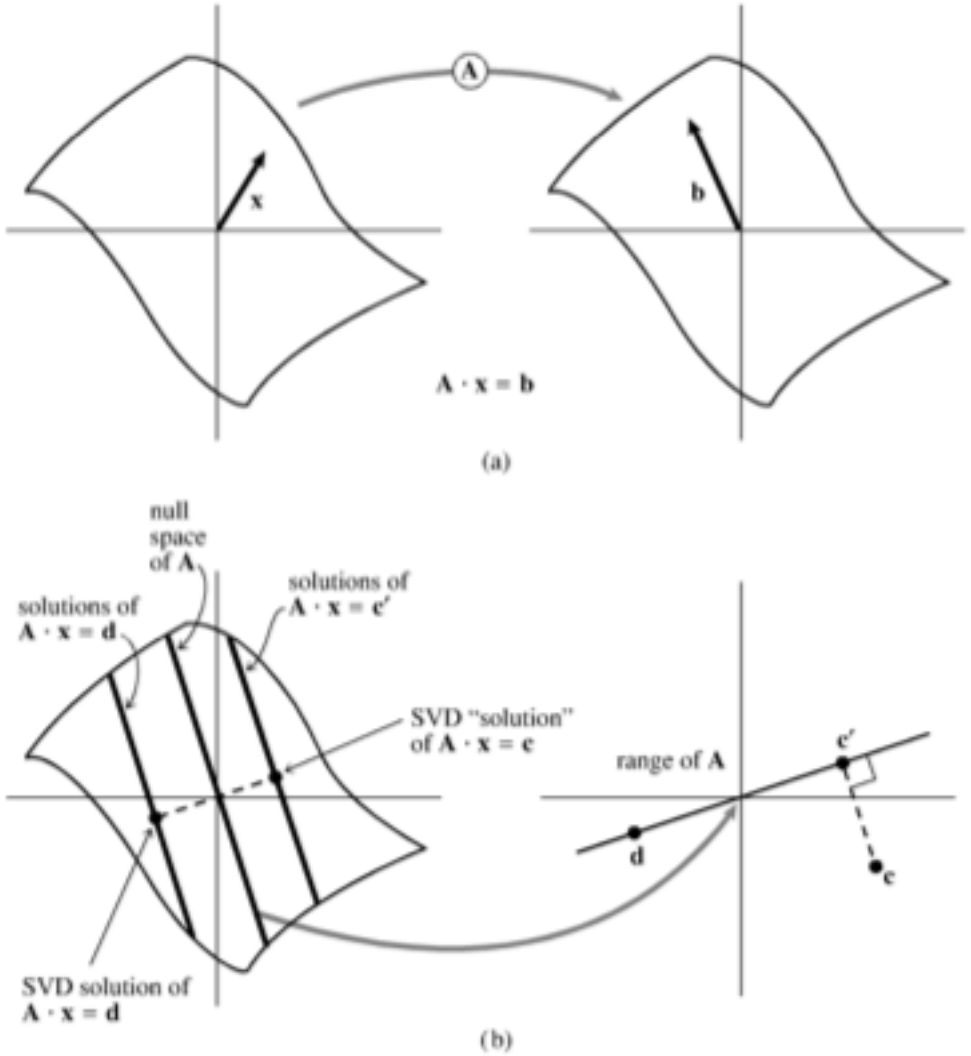


Figure 2.6.1. (a) A nonsingular matrix **A** maps a vector space into one of the same dimension. The vector **x** is mapped into **b**, so that **x** satisfies the equation $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$. (b) A singular matrix **A** maps a vector space into one of lower dimensionality, here a plane into a line, called the "range" of **A**. The "nullspace" of **A** is mapped to zero. The solutions of $\mathbf{A} \cdot \mathbf{x} = \mathbf{d}$ consist of any one particular solution plus any vector in the nullspace, here forming a line parallel to the nullspace. Singular value decomposition (SVD) selects the particular solution closest to zero, as shown. The point **c** lies outside of the range of **A**, so $\mathbf{A} \cdot \mathbf{x} = \mathbf{c}$ has no solution. SVD finds the least-squares best compromise solution, namely a solution of $\mathbf{A} \cdot \mathbf{x} = \mathbf{c}'$, as shown.

Non-singular matrix

Singular matrix

From Numerical Recipes, Press et al

Singular Value Decomposition (SVD)

- Define two orthonormal bases, with vectors $\mathbf{u}$ in R^M and $\mathbf{v}$ in R^N. In matrix form $\mathbf{U}$ is MxM and $\mathbf{V}$ is NxN
- $\mathbf{u}_1$, … , $\mathbf{u}_r$ is an orthonormal basis for the column space
- $\mathbf{u}_{r+1}$, ... , $\mathbf{u}_M$ is an orthonormal basis for the left null space N (defined as $\mathbf{A}^T\mathbf{x} = 0$)
- $\mathbf{v}_1$, … , $\mathbf{v}_r$ is an orthonormal basis for the row space
- $\mathbf{v}_{r+1}$, … ,$\mathbf{v}_N$ is an orthonormal basis for the null space N ($\mathbf{Ax}$=0).
- These bases "diagonalize" $\mathbf{A}$: $\mathbf{Av}_1 = \sigma_1\mathbf{u}_1$, $\mathbf{Av}_2 = \sigma_2\mathbf{u}_2$ …
- Also called Karhunen-Loeve (KL) Transform

SVD continued

Reduced SVD

$$\begin{matrix}(m \text{ by } n)(n \text{ by } r)\\ \boldsymbol{AV_r = U_r\Sigma_r}\\ (m \text{ by } r)(r \text{ by } r)\end{matrix} \quad A\begin{bmatrix} & & \\ \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_r \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_r \\ & & \end{bmatrix}\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$$

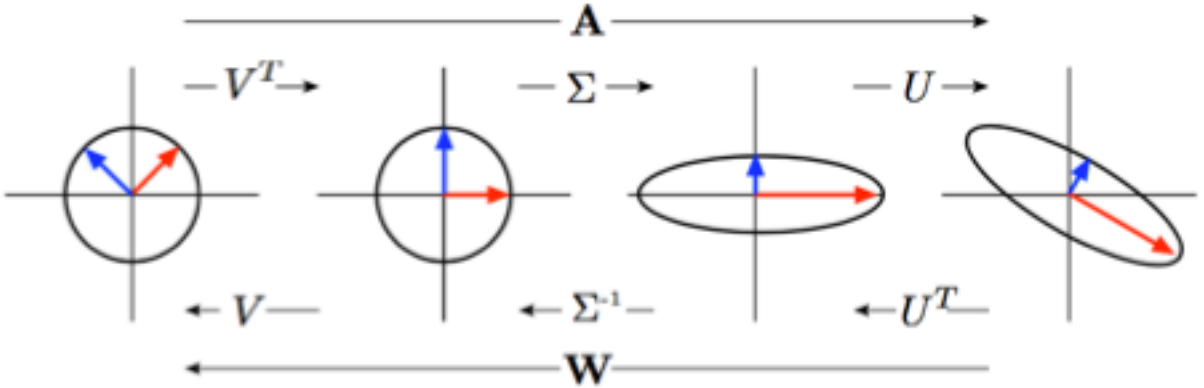
- Add null space, which is already orthogonal to first *r* **v**'s and **u**'s to go from reduced to full SVD

$$\begin{matrix}(m \text{ by } n)(n \text{ by } n)\\ \boldsymbol{Av} \text{ equals } \boldsymbol{U\Sigma}\\ (m \text{ by } m)(m \text{ by } n)\end{matrix} \quad A\begin{bmatrix} & & & & \\ \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_r & \cdots & \boldsymbol{v}_n \\ & & & & \end{bmatrix} = \begin{bmatrix} & & & & \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_r & \cdots & \boldsymbol{u}_m \\ & & & & \end{bmatrix}\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$$

- Now the last matrix $\boldsymbol{\Sigma}$ is *M*x*N* with first *r* on the diagonal $\mathbf{s_i}$, and the rest 0. **U** and **V** are now square matrices (*M*x*M* and *N*x*N*) hence $\mathbf{AV = U\Sigma}$ or $\mathbf{A = U\Sigma V^T = u_1\sigma_1 v_1 + \ldots u_r\sigma_r v_r}$
- Rank order $\sigma_1 > \sigma_2 > \ldots > \sigma_r$

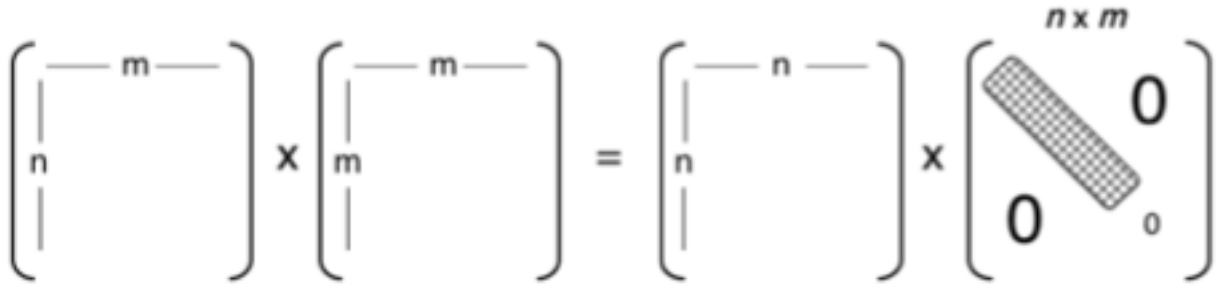
SVD in Pictures

- SVD as a sequence of rotation-stretch-rotation $\mathbf{A} = \mathbf{U\Sigma V}^T$

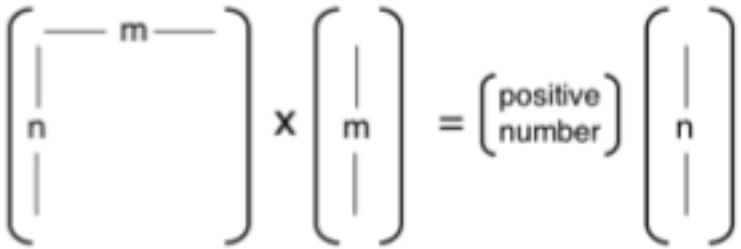


- $\mathbf{AV} = \mathbf{U\Sigma}$

$$\Sigma \equiv \begin{bmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_r & \\ 0 & & & \ddots \\ & & & & 0 \end{bmatrix}$$



- $\mathbf{A}\mathbf{v}_1 = \sigma_1 \mathbf{u}_1$



Example

- If $\mathbf{A} = \mathbf{x}\mathbf{y}^T$ (rank 1) with unit vectors $\mathbf{x}$ and $\mathbf{y}$, what is the SVD of $\mathbf{A}$?

Solution:

- The reduced SVD is exactly $\mathbf{x}\mathbf{y}^T$, with rank $r = 1$. It has $\mathbf{u}_1 = \mathbf{x}$ and $\mathbf{v}_1 = \mathbf{y}$ and $\sigma_1 = 1$. For the full SVD, complete $\mathbf{u}_1 = \mathbf{x}$ with N-1 additional vectors (N is dimensionality of $\mathbf{x}$) to an orthonormal basis of $\mathbf{u}$'s, and complete $\mathbf{v}_1 = \mathbf{y}$ to an orthonormal basis of $\mathbf{v}$'s (M-1 additional vectors, M is dimensionality of $\mathbf{y}$). No new σ's, only $\sigma_1 = 1$

How to Construct u's and v's?

- **v**'s will be orthonormal eigenvectors of $\mathbf{A}^T\mathbf{A}$:
- $\mathbf{A}^T\mathbf{A} = (\mathbf{U\Sigma V}^T)^T\mathbf{U\Sigma V}^T = \mathbf{V\Sigma}^T\mathbf{U}^T\mathbf{U\Sigma V}^T = \mathbf{V\Sigma}^T\mathbf{\Sigma V}^T$
- We see that $\sigma^2=\lambda$ of $\mathbf{A}^T\mathbf{A}$.
- u's can easily be computed from $\mathbf{Av_i} = \sigma_i\mathbf{u_i}$. They are orthogonal:

$$\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{u}_j = \left(\frac{A\boldsymbol{v}_i}{\sigma_i}\right)^{\mathrm{T}}\left(\frac{A\boldsymbol{v}_j}{\sigma_j}\right) = \frac{\boldsymbol{v}_i^{\mathrm{T}}A^{\mathrm{T}}A\boldsymbol{v}_j}{\sigma_i\sigma_j} = \frac{\sigma_j^2}{\sigma_i\sigma_j}\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v}_j = \mathbf{zero}.$$

- Show that **u**'s are eigenvectors of $\mathbf{AA}^T$
- Soln: $\mathbf{AA}^T = \mathbf{U\Sigma V}^T(\mathbf{U\Sigma V}^T)^T = \mathbf{U\Sigma V}^T\mathbf{V\Sigma}^T\mathbf{U}^T = \mathbf{U\Sigma\Sigma}^T\mathbf{U}^T$

Linear Algebra with SVD

- Linear algebra $\mathbf{Ax} = \mathbf{b}$, assume $\mathbf{A}$ is NxN
- Let's do matrix inversion: $\mathbf{A}^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{T}$
- This can be solved unless some $\sigma_i = 0$: matrix is singular, i.e. there is non-zero null space
- More generally we can define condition number σ_1/σ_N: if this is large the matrix is ill-conditioned
- SVD will diagnose the condition situation of the matrix
- If we want to solve homogeneous equation $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{b} = 0$ we use the null space of $\mathbf{U}/\mathbf{V}$, corresponding to $\sigma_i=0$

Solving Singular Matrix Problems

- $\mathbf{Ax} = \mathbf{b}$ can be solved if $\mathbf{b}$ is in the range of $\mathbf{A}$. If so we have an infinite number of solutions, since we can add null space solutions to it in any linear combination
- We can put additional constraint, such as the norm $||\mathbf{x}||$ is minimized (i.e. pick the solution with the shortest length $|\mathbf{x}|^2$). Then we can set the null space solutions to 0. This is achieved by setting $\sigma_i^{-1}=0$ wherever $\sigma_i=0$ (null space) and solve $\mathbf{x} = \mathbf{V\Sigma^{-1}U^T b}$
- If b is not in range we will not have an exact solution. But we can still use $\mathbf{x} = \mathbf{V\Sigma^{-1}U^T b}$ as the solution which minimizes $r = |\mathbf{Ax-b}|$. So SVD here acts as an optimization routine (optimization=minimization or maximization). Optimization is a topic of lecture 6.

Proofs that $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}$

- **b** is in range: suppose **x**' is in null space, and $\boldsymbol{\Sigma}^{-1}$ is the modified inverse of $\boldsymbol{\Sigma}$. What happens to the norm of **x**+**x**'

$$|\mathbf{x}+\mathbf{x}'| = |\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}+\mathbf{x}'| = |\mathbf{V}(\boldsymbol{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}+\mathbf{V}^T\mathbf{x}')| = |\boldsymbol{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}+\mathbf{V}^T\mathbf{x}'| > |\mathbf{x}|$$

unless **x**' = 0 because the two are orthogonal (**x**' is in null space, **x** is not).

- **b** is not in range: we add **x**' to **x**, **Ax-b** is modified by **b**'=**Ax**' (in range), show $r = |\mathbf{Ax}-\mathbf{b}|$ is minimized:

$$|\mathbf{Ax}-\mathbf{b}+\mathbf{b}'| = |(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T)\,\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}-\mathbf{b}+\mathbf{b}'|$$

$$= |(\mathbf{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T-\mathbf{1})\mathbf{b}+\mathbf{b}'| = |\mathbf{U}[(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}-\mathbf{1})\mathbf{U}^T\mathbf{b}+\mathbf{U}^t\mathbf{b}']|$$

$$= |(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}-\mathbf{1})\mathbf{U}^T\mathbf{b}+\mathbf{U}^t\mathbf{b}'| > |(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}-\mathbf{1})\mathbf{U}^t\mathbf{b}| \text{ unless } \mathbf{b}'=0.$$

- In practice we want to fix ill-conditioned matrices with SVD: throw away all singular values whose value is too small to be meaningful (machine precision roundoff errors etc.). So if $\sigma_i < \varepsilon$ we set $\sigma_i^{-1} = 0$, where ε is machine precision.

Non-square Matrices

- If there are fewer equations M than variables N the system is underdetermined and we have N-M dimensional family of solutions. SVD provides the family of solutions by providing $\sigma_i < \varepsilon$ (zero or close to zero). We then zero these in $\mathbf{\Sigma}^{-1}$ and get $\mathbf{x} = \mathbf{V\Sigma}^{-1}\mathbf{U}^T\mathbf{b}$ as the particular solution. Columns of $\mathbf{V}$ corresponding to zeroed σ_i give us the basis vectors which, when added to the particular solution, span the solution space.
- If there are more equations M than variables N the system is overdetermined and we can only obtain the least square solution, which is given by $\mathbf{x} = \mathbf{V\Sigma}^{-1}\mathbf{U}^T\mathbf{b}$ (same proof). There will be no zeros of σ_i unless there are hidden degeneracies.

Key Ideas of SVD

- The SVD factors **A** into $\mathbf{U\Sigma V}^T$, with r singular values $\sigma_1 \geq .. .\geq \sigma_r > 0$.
- The numbers σ_1^2 , … , σ_r^2 are the nonzero eigenvalues of $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$.
- The orthonormal columns of **U** and **V** are eigenvectors of $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$.
- Those columns hold orthonormal bases for the four fundamental subspaces of **A**.
- Those bases diagonalize the matrix: $\mathbf{Av_i} = \sigma_i\mathbf{u_i}$ for $i \leq r$. This is $\mathbf{AV} = \mathbf{U\Sigma}$.
- $\mathbf{A} = \sigma_1\mathbf{u_1v^T_1} + \cdots + \sigma_r\mathbf{u_rv^T_r}$ and σ_1 is the maximum of the ratio $||\mathbf{Ax}||/||\mathbf{x}||$.

Applications of SVD

- Diagnosis of matrix condition number
- Construction of orthonormal basis (we can also do that with QR/RQ)
- Lower rank approximation of matrices
  $\mathbf{A} = \mathbf{U\Sigma V}^T = \mathbf{u}_1\sigma_1\mathbf{v}_1 + \ldots \mathbf{u}_k\sigma_k\mathbf{v}_k$, or $a_{ij} = \Sigma_{k=1}^{N} \sigma_k u_{ik} v_{jk}$ where $k < r$.
  The challenge is that to construct this we need r^3 process, which can be expensive
- Construction of orthonormal basis for a subspace: suppose we have N vectors in M dimensional space, with N<M, so N vectors span some subspace of the full space. Construct A as a *MxN* matrix with the N columns as the vectors. Do SVD. The columns of U (which is MxM) corresponding to non-zero σ_k (check if dim=N, otherwise N vectors are not independent)
- This can also be done with QR (Gram-Schmidt orthogonalization)

From Lecture 3: Multivariate Linear Least Squares

- We can generalize the model to a generic functional form
  $y_i = a_0 X_0(x_i) + a_1 X_1(x_i) + \ldots + a_{M-1} X_{M-1}(x_i)$
- The problem is linear in a_j and can be nonlinear in x_i,
  e.g. $X_j(x_i) = x_i^j$

$$\chi^2 = \sum_{i=0}^{N-1} \left[\frac{y_i - \sum_{k=0}^{M-1} a_k X_k(x_i)}{\sigma_i} \right]^2$$

- We can define design matrix $A_{ij} = X_j(x_i)/\sigma_i$ and
- $b_i = y_i/\sigma_i$

$$\chi^2 = |\mathbf{A} \cdot \mathbf{a} - \mathbf{b}|^2$$

Solving Least Squares with SVD

- We want to find **a** for which $\chi^2=|\mathbf{Aa-b}|^2$ is minimized.
- $\mathbf{a} = \Sigma_{i=1}^{M} (\mathbf{U_i b}/\sigma_i)\mathbf{V_i}$
- The errors are $\mathrm{Var}(a_j)=\Sigma_{i=1}^{M} (V_{ji}^2/\sigma_i^2)$
- Note that this is no different than diagonalizing precision matrix $\mathbf{A^T A=V\Sigma^T\Sigma V^T}$, where $(\mathbf{\Sigma^T\Sigma})_{ii}=\sigma_i^2$ and $(\mathbf{\Sigma^T\Sigma})_{ij}=0$ for j not equal to i.
- This is because the inverse precision matrix is covariance matrix, $(\mathbf{A^T A})^{-1}=\mathbf{V}(\mathbf{\Sigma^T\Sigma})^{-1}\mathbf{V^T}$,
  so $(\mathbf{A^T A})_{jj}=\Sigma_{i=1}^{M} (V_{ji}^2/\sigma_i^2)$ and $(\mathbf{A^T A})_{jk}=\Sigma_{i=1}^{M} (V_{ji}V_{ik}/\sigma_i^2)$

Solving Least Squares with SVD

- When normal equations are close to singular we have an under-determined system. This happens when the variables are strongly correlated.
- This is ill-conditioned problem and SVD is its cure: we want to find a for which $\chi^2=|\mathbf{Aa}-\mathbf{b}|^2$ is minimized.
- $\mathbf{a}=\Sigma_{i=1}^{M}(\mathbf{U}_i\mathbf{b}/\sigma_i)\mathbf{V}_i$ where we set, if $\sigma_i<\varepsilon$, $\sigma_i^{-1}=0$.
- This is equivalent to eliminating some parameters and replacing them with a single linear combination, which is better constrained.
- Note that this is no different than diagonalizing $\boldsymbol{\alpha}=\mathbf{C}^{-1}=\mathbf{A}^T\mathbf{A}$ and doing the same procedure there (eliminating parameter combinations that are poorly determined to reduce the dimensionality)

Singular Covariance Matrix

- What do we do if the covariance matrix is singular, i.e. some eigenvalues of precision matrix are nearly 0?
- If we invert precision matrix we will get infinity in covariance matrix. This is just a statement that we have too many variables to determine them all
- We need to "regularize" our solution by reducing the number of parameters
- One way to do so is to use eigenvectors, columns of $\mathbf{X}_L$ corresponding to the non-zero eigenvalues λ_i of precision matrix, to define new variables as the linear combinations of original variables: $\mathbf{u}=\mathbf{X}_L\mathbf{a}$
- the variables u_j are uncorrelated with variance λ_i^{-1}
- Posterior becomes proportional to
  $\exp(-\boldsymbol{\delta a}^T\boldsymbol{\alpha}\boldsymbol{\delta a}/2)=\exp(-\boldsymbol{\delta a}^T\mathbf{X}^T_L\mathbf{D}\mathbf{X}_L\boldsymbol{\delta a}/2)=\exp(-\boldsymbol{\delta u}^T\mathbf{D}\boldsymbol{\delta u}/2)$
- This is now a product of independent variables, but fewer than original

Decorrelating the Variables (matrix square root)

- We can decorrelate the variables using spectral principal axis rotation (diagonalization). One way to do so is to use $\boldsymbol{\alpha}=\mathbf{X}^T_L\mathbf{D}\mathbf{X}_L$
- Use eigenvectors, columns of $\mathbf{X}_L$ corresponding to the non-zero eigenvalues λ_i of precision matrix, to define new variables as the linear combinations of original variables: $\mathbf{u}=\mathbf{X}_L\mathbf{a}$
- the variables u_j are uncorrelated with variance λ_i^{-1}
- This procedure is for example needed if we want to generate correlated variables, since we only have random number generator for a single uncorrelated variable: we generate u's and then use $\mathbf{X}_L$ to rotate back into the original basis of a's.

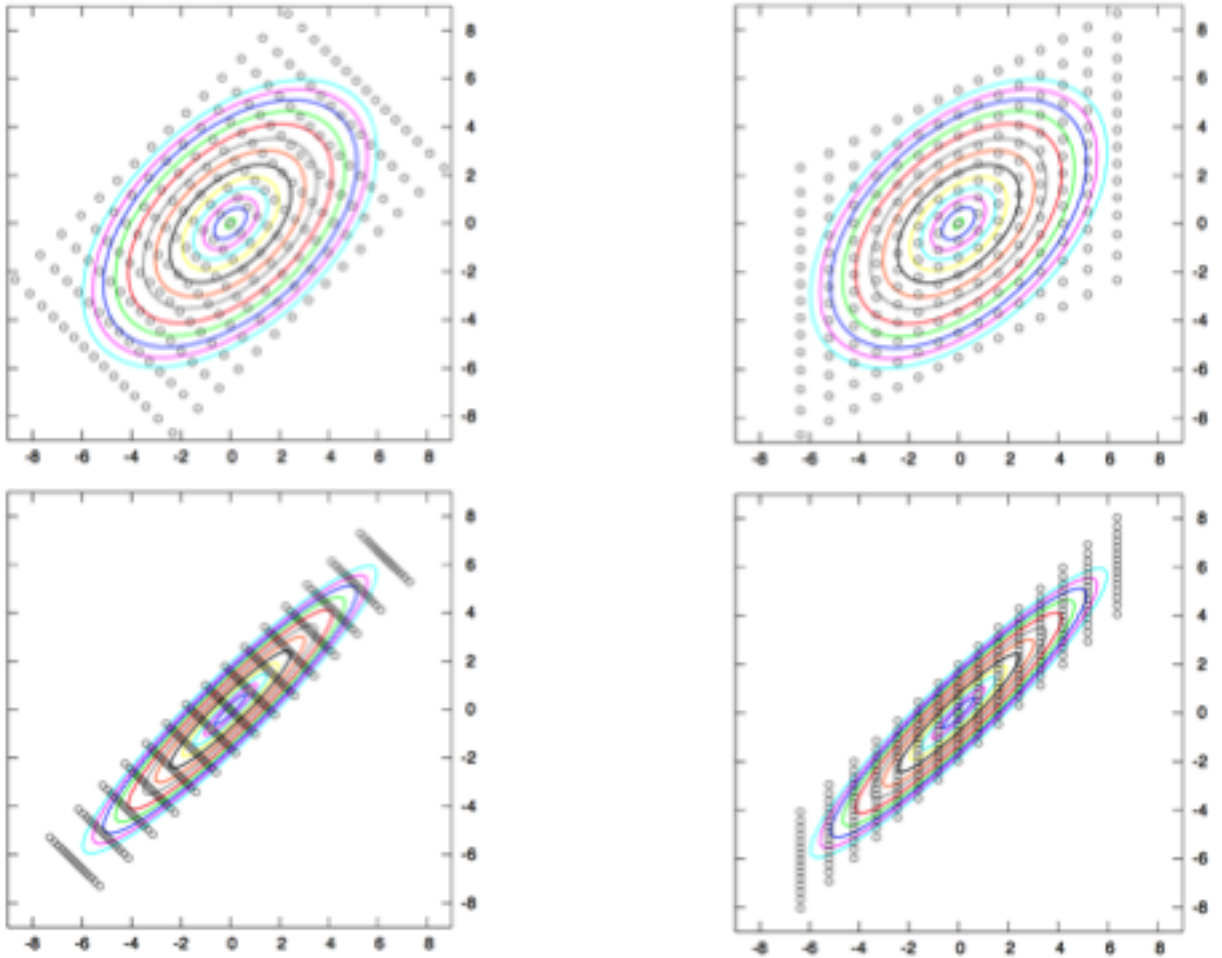
Decorrelating the Variables

- Posterior becomes proportional to
  $\exp(-\delta a^T \alpha \delta a/2) = \exp(-\delta a^T X^T_L D X_L \delta a/2) = \exp(-\delta u^T D \delta u/2)$
- This is now a product of independent variables
- $\delta a^T X^T_L D X_L \delta a = \delta u^T D \delta u$
- We can further rescale (normalize) $u_i' = u_i \lambda_i^{-1/2}$
- $\delta a^T X^T_L D X_L \delta a = \delta u'^T \delta u'$
- Alternative: we can also perform principal axis rotation using Cholesky $a=L^T L$: $\delta a^T L^T L \delta a = \delta u''^T \delta u''$, where $\delta u'' = L \delta a$
- The two are not the same
- Cholesky faster to compute, so it is the standard "matrix square root"

Spectral (eigen)-decomposition versus Cholesky decomposition: uniform sampling

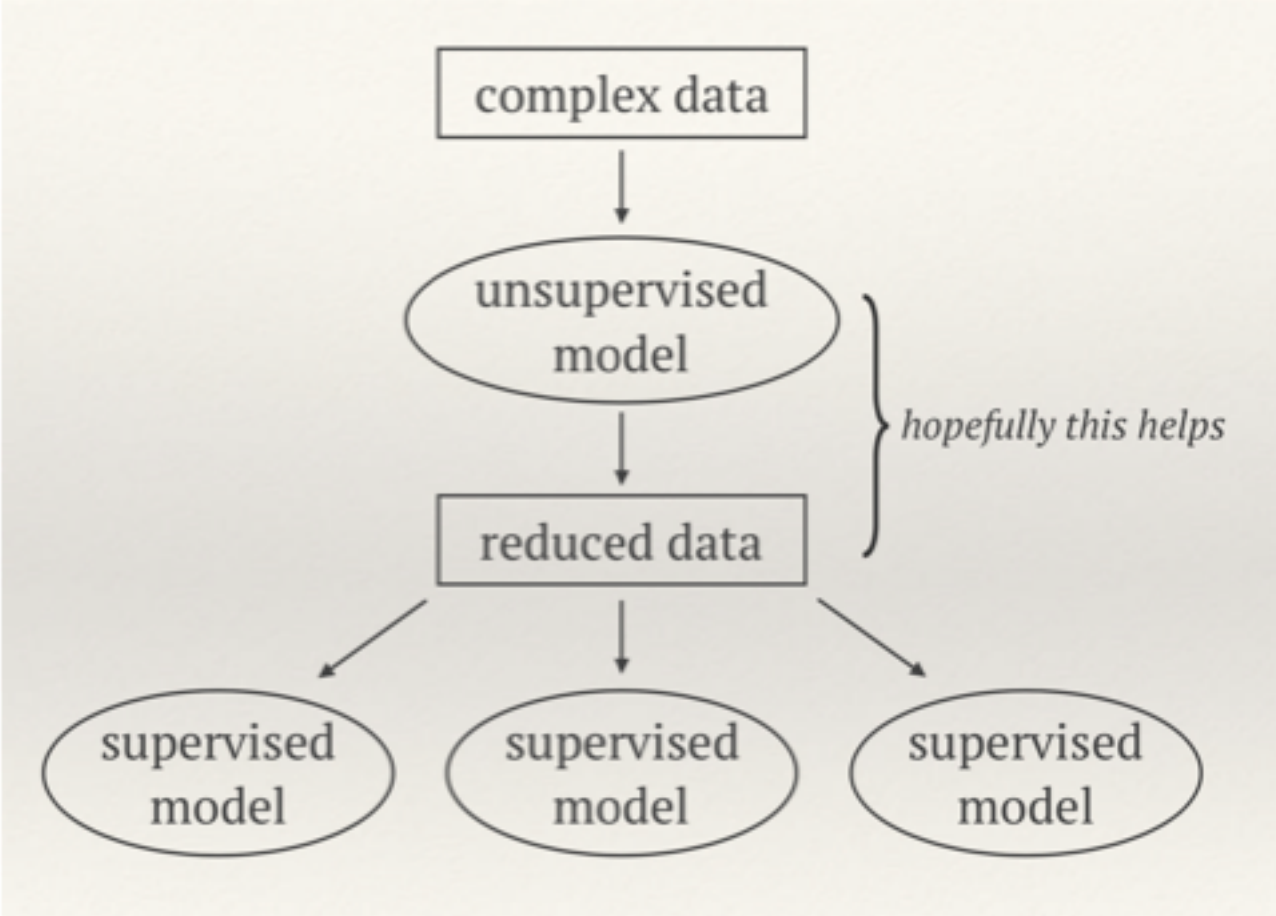
Spectral (eigenvectors)

Cholesky



Unsupervised machine learning

- Discovering structure in unlabeled data
- Examples: clustering, dimensionality reduction
- The promise:



Dimensionality reduction

- Want to reduce it preserving pairwise distance between data points (e.g. t-SNE).
- If reduced too much we get crowding problem

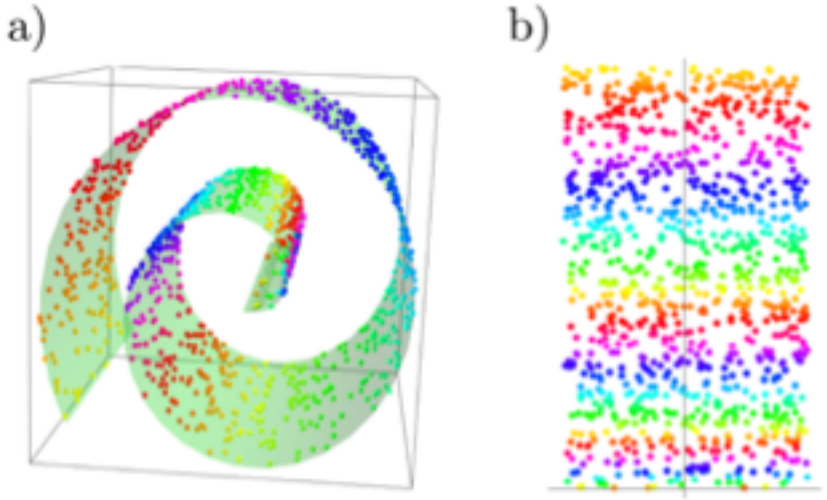


FIG. 48 Data distributed in a three dimensional space (a) that can effectively be described on a two-dimensional surface (b). A common goal of dimensional reduction is to preserve the local structure in the data. The embedding of (a) to (b) preserves the local structure of the data as can be seen by inspecting the color gradient.

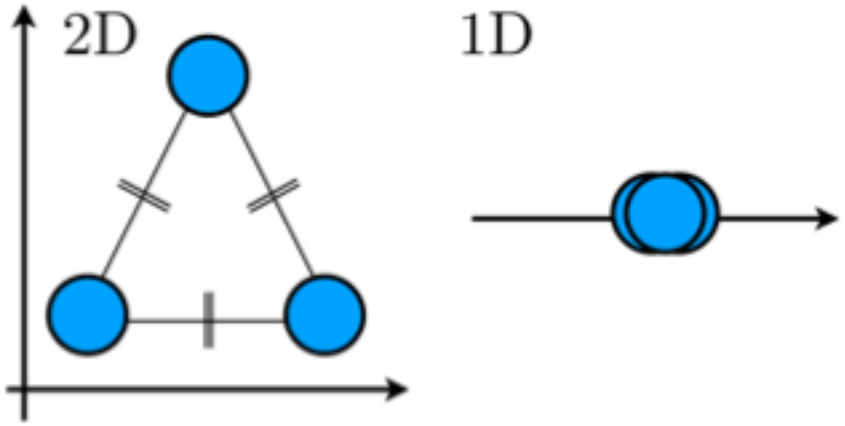
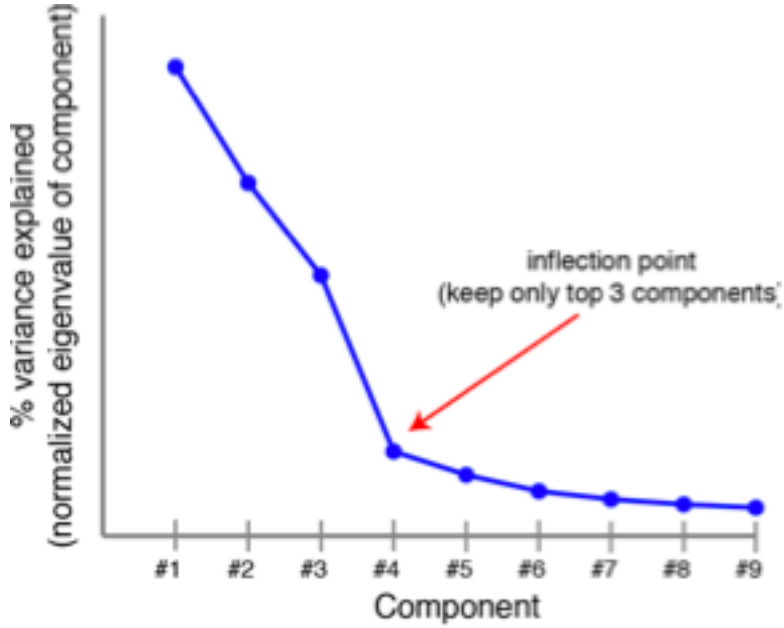


FIG. 49 Illustration of the crowding problem. (left) Consider an original space in 2 dimensions with three pairwise equidistant points. (right) Mapped space: if one wants to preserves the ordination or distances in a 1D space *exactly*, all points must collapse onto one another.

Principal Component Analysis (PCA)

- PCA is diagonalizing $\mathbf{A}^T\mathbf{A}$ or performing SVD of $\mathbf{A}$ and use the top few eigenvalues/principal components as a solution. We do this to reduce the rank of the matrix: dimensionality reduction. Note that $\mathbf{A}^T\mathbf{A}$ is MxM, and $(\mathbf{A}^T\mathbf{A})_{ij}$ measures the correlation between i and j data components. If N is large (lots of data) SVD may be expensive, so we can do PCA by diagonalizing $\mathbf{A}^T\mathbf{A}$.
- The idea is that if the correlation matrix is of low rank it can be approximated by a few largest eigenvectors only. If so this will become clear by the pattern of singular values (which are sqrt of eigenvalues of $\mathbf{A}^T\mathbf{A}$): a few will be large and the rest will be small and relatively constant (because uncorrelated noise produces M eigenvalues of equal value).

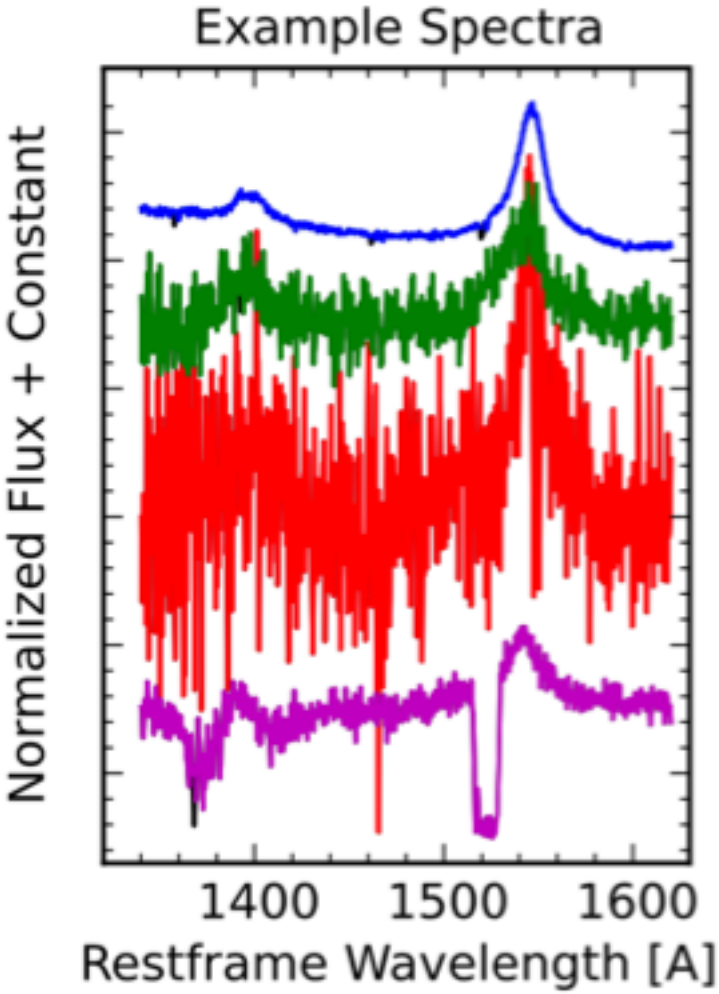


Principal component analysis as data analysis tool

- We do not need to remove just singular values with $\sigma_i<\varepsilon$. If we have strong ordering of singular values we can just keep the top few.
- Suppose we have some set of data measured as a function of some coordinate x_i. As an example, in HW4 we give data of quasar (QSO) light intensity as a function of wavelength for several QSOs. This is called QSO spectrum.
- We measure the data for several samples (e.g. QSOs) and we would like to describe individual variations seen in the data. For example, most QSOs look very similar in their light intensity as a function of wavelength, but there are differences in regions of strong metal line emission.
- We first normalize all the QSO spectra (so that the mean intensity is the same) and then define the mean QSO spectrum by averaging over all spectra. We subtract this mean from each spectrum. We can form NxM matrix $\mathbf{A}$, where N is the number of data vectors (QSOs) and M is the number of QSO pixels.
- Now we want to describe the variations from the mean. It is possible that these will be described with a set of a few simple 1-d spectra. To test this hypothesis we perform PCA. We want to minimize $\chi^2=|\mathbf{Aa-b}|^2$, where $\mathbf{b}$ is the data vector for each QSO and a are the coefficients. Instead of the trivial solution $a_i=1$ we do this by reducing the rank of A with SVD.

QSO Example (HW 4)

- We want to identify variations in QSO spectrum shapes by doing PCA
- Shown are first 4 PCA components
- Noise is a problem: for now we assume noise is the same for all spectra
- Metal absorptions are a problem: very non-Gaussian noise (outliers)
- Incomplete data are a problem
- Welcome to the world of real data analysis! We will discuss some of these issues in the class later



Noise in PCA

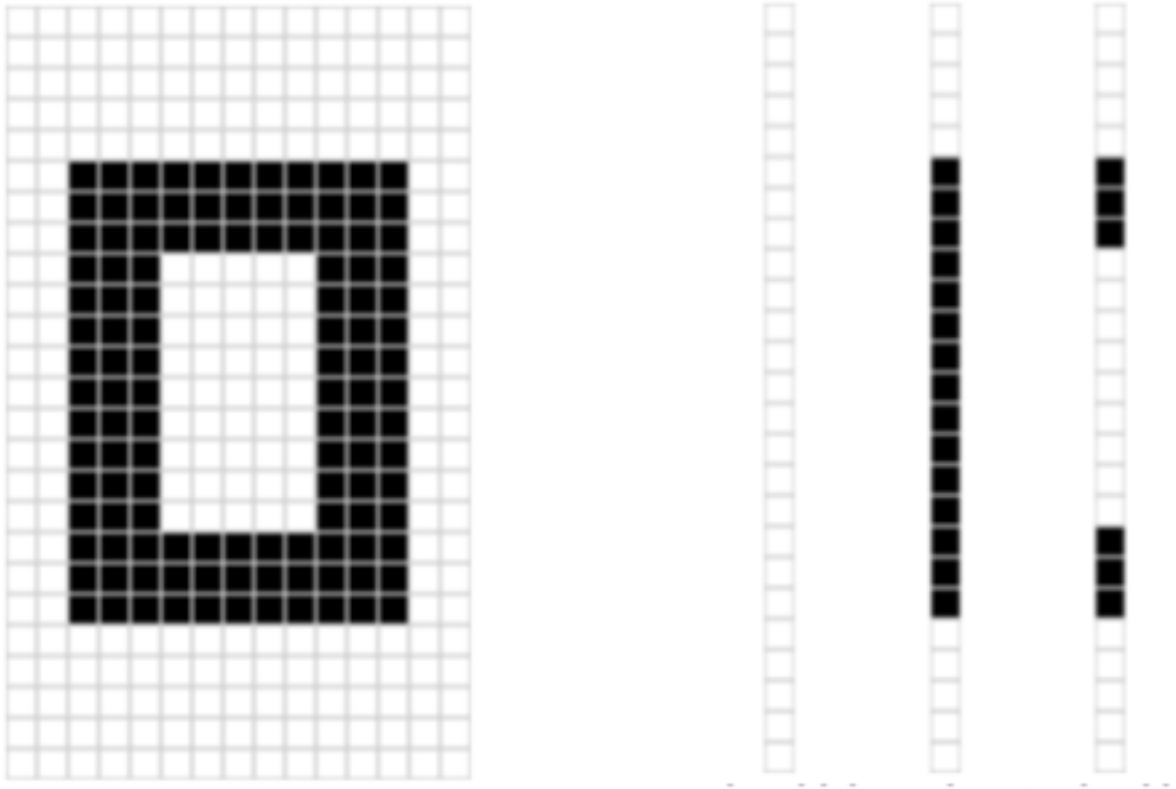
- If noise is uncorrelated it adds a diagonal to $\mathbf{A}^T\mathbf{A}$.
- If in the absence of noise the data were described with a few PCA only, in the presence of noise all PCAs may be non-zero, but higher PCAs are all noise generated, and of slowly varying amplitude.
- We look at PCA values and observe where they stop decreasing significantly, or maybe use a large jump in value.

SVD Properties, Generalizations

- With the PCA eigenvectors we can fit each QSO spectrum to these by taking a dot product between the data and the eigenvector to get the coefficient. Then we approximate the spectrum with a low rank version: $\mathbf{A} = \mathbf{U\Sigma V}^T = \mathbf{u}_1\sigma_1\mathbf{v}_1 + \ldots \mathbf{u}_k\sigma_k\mathbf{v}_k$
- It is non-parametric since we do not fit the data with some parametrized model. Unique answers, but may not give the best answer when we have additional information
- It is a dimensionality reduction method (lowering matrix rank). One can instead require independence of components: independent component analysis (ICA, next lecture)
- It is a linear method. More sophisticated non-supervised algorithms are nonlinear. For example, we may be able to make the data analysis more linear if we perform a non-linear transformation of variables: kernel PCA

An image reduction example: 15x25 block

- This block has only 3 column (row) patterns. We can do covariance analysis of columns: for any *i* and *j* column (15) we add products of *ik* and *jk* pixels over all *k* (25), but this can give only 3x3 possible products



We can represent it with 0's and 1's

Let's perform SVD of **M** or diagonalization of $\mathbf{M}^T\mathbf{M}$

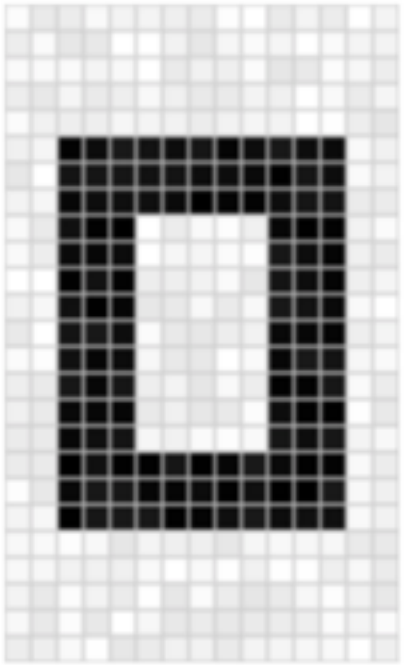
$$M = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}$$

We find 3 singular components:

$\sigma_1 = 14.72$
$\sigma_2 = 5.22$
$\sigma_3 = 3.31$

- We can write $M = \mathbf{u}_1\sigma_1\,\mathbf{v}_1^T + \mathbf{u}_2\sigma_2\,\mathbf{v}_2^T + \mathbf{u}_3\sigma_3\,\mathbf{v}_3^T$
- What if we have noise? We get

$\sigma_1 = 14.15$
$\sigma_2 = 4.67$
$\sigma_3 = 3.00$
$\sigma_4 = 0.21$
$\sigma_5 = 0.19$
...
$\sigma_{15} = 0.05$

- 

Let us truncate at $r = 3$

$$M \approx \mathbf{u}_1\sigma_1 \mathbf{v}_1^T + \mathbf{u}_2\sigma_2 \mathbf{v}_2^T + \mathbf{u}_3\sigma_3 \mathbf{v}_3^T$$

noisy reconstructed

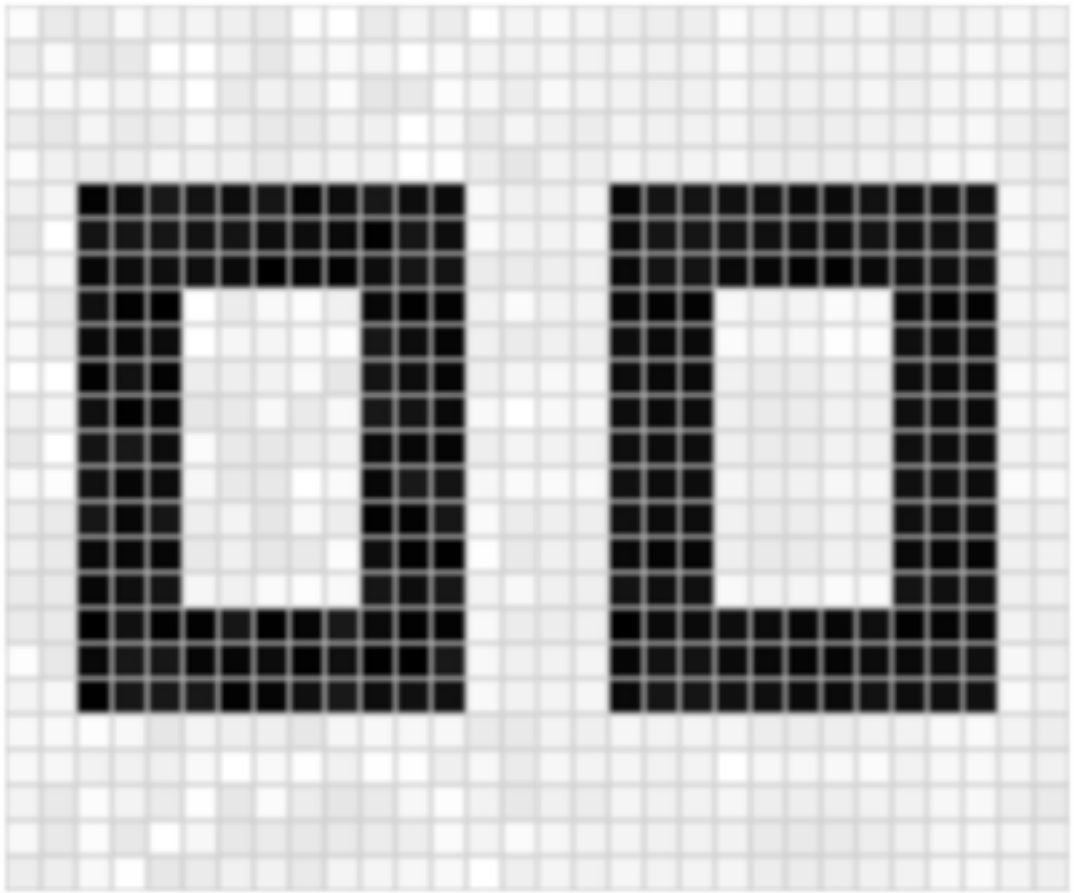


Image compression: A=600x600 SVD (not a natural application of SVD)

Original

10 singular values

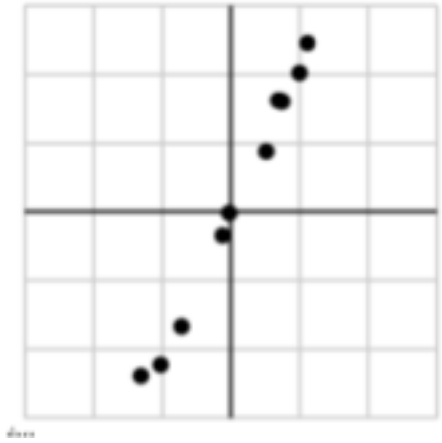
Image compression: A=600x600 SVD (not a natural application of SVD)

50 SV

100 SV

Data Analysis

- We can do data analysis with SVD/PCA
- We have collected these data:



- Let us collect the data into 2x10 matrix

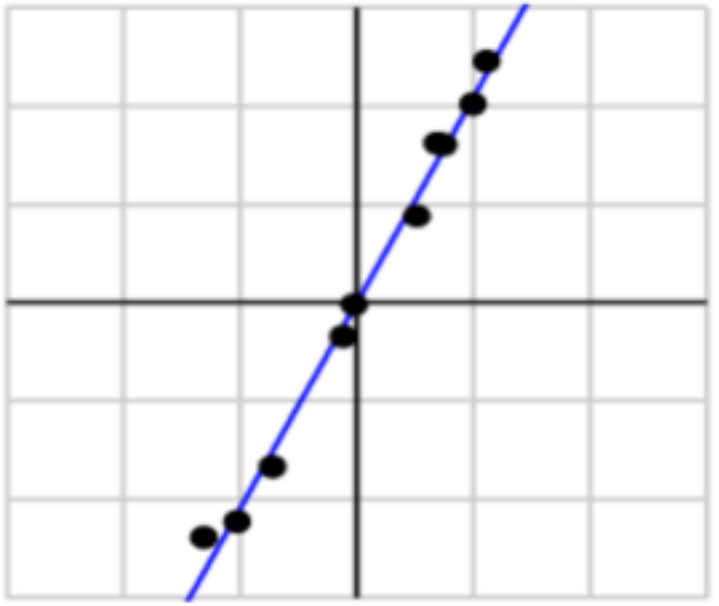
$$\begin{matrix} -1.03 & 0.74 & -0.02 & 0.51 & -1.31 & 0.99 & 0.69 & -0.12 & -0.72 & 1.11 \\ -2.23 & 1.61 & -0.02 & 0.88 & -2.39 & 2.02 & 1.62 & -0.35 & -1.67 & 2.46 \end{matrix}$$

- SVD gives $\sigma_1 = 6.04$, $\sigma_2 = 0.22$ so one value is large, second may be noise

What is SVD in this case?

- The matrix $A^T A$ is given by $\begin{pmatrix} \langle x^2 \rangle & \langle xy \rangle \\ \langle xy \rangle & \langle y^2 \rangle \end{pmatrix}$
- It has 2 eigenvalues. If *y* is perfectly correlated with *x* the matrix is singular and one eigenvalue is 0.
- In this case we then have a linear relation between x and y.
- Note that PCA is a linear method:

if relation is nonlinear it may fail

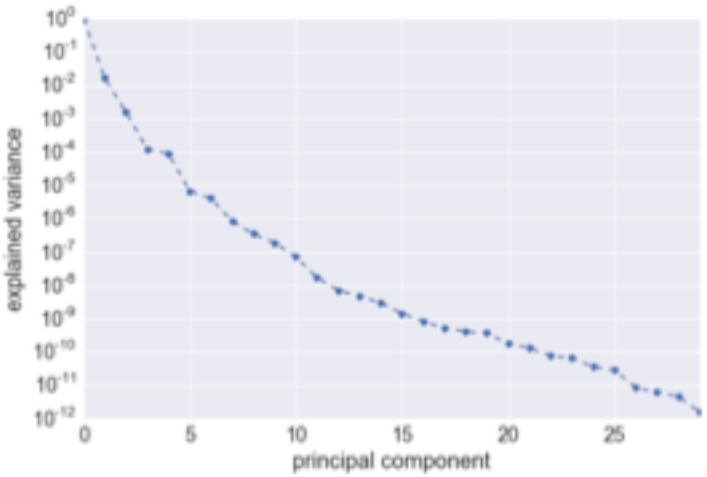


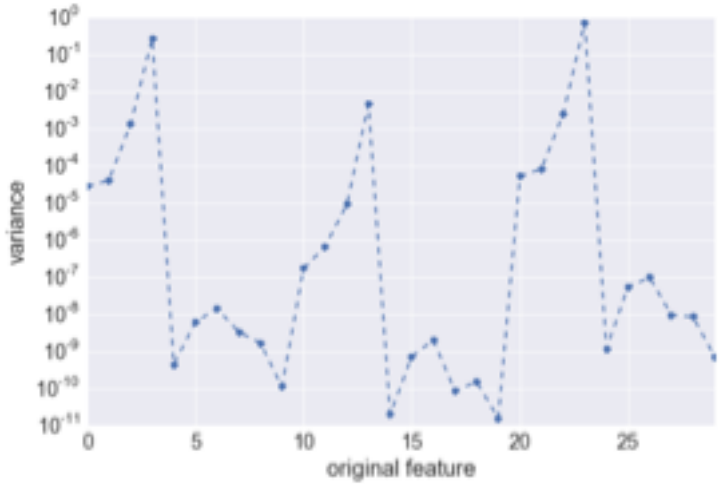
SVD Preprocessing

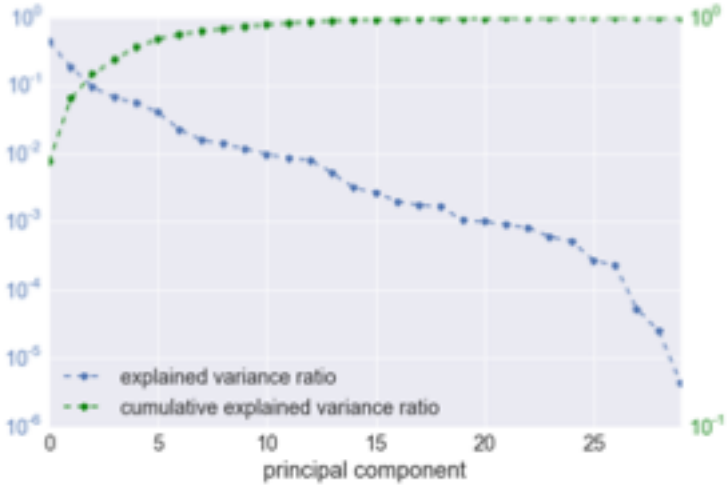
Centering: taking the mean out.
Common data analysis method.

SVD Preprocessing

- Scaling or normalizing: use the units where the data are all similar in numbers, otherwise rescale (also common to do)







Main Decompositions of Linear Algebra

- 1. Singular Value Decomposition $\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T$
  2. Gram-Schmidt Orthogonalization $\mathbf{A} = \mathbf{Q}\,\mathbf{R}$ (can be defined for NxM)
  3. Gaussian Elimination $\mathbf{A} = \mathbf{L}\mathbf{U}$
- We might prefer to separate out singular values σ_i, heights h_i and pivots d_i:
  1. $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ with unit vectors in **U** and **V**.
  The singular values σ_i are in **Σ**.
  2. $\mathbf{A} = \mathbf{Q}\mathbf{H}\mathbf{R}$ with unit vectors in **Q** and diagonal 1's in **R**.
  The heights h_i are in **H**.
  3. $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{U}$ with diagonal 1's in **L** and **U**.
  The pivots d_i are in **D**.
- Each h_i tells the height of column i above the plane of columns 0 to $i-1$. The volume of the full N-dimensional box (r=M=N) comes from $\mathbf{A}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T=\mathbf{L}\mathbf{D}\mathbf{U}=\mathbf{Q}\mathbf{H}\mathbf{R}$:
- $|\det \mathbf{A}| = |$ *product of* σ's $| = |$ *product of* d's $| = |$ *product of* h's $|$.

Literature

- *Numerical Recipes*, Press et al., Chapter 2, 11, 15
- *Computational Physics,* M. Newman, Chapter 6