

Mathematics of Big Data, I

Lecture 4: PCA and Dim Reduction, Spectral Decomposition, SVD, Generative Learning Algorithm, and Gaussian Discriminant Analysis

Weiqing Gu

Professor of Mathematics

Director of the Mathematics Clinic

Harvey Mudd College

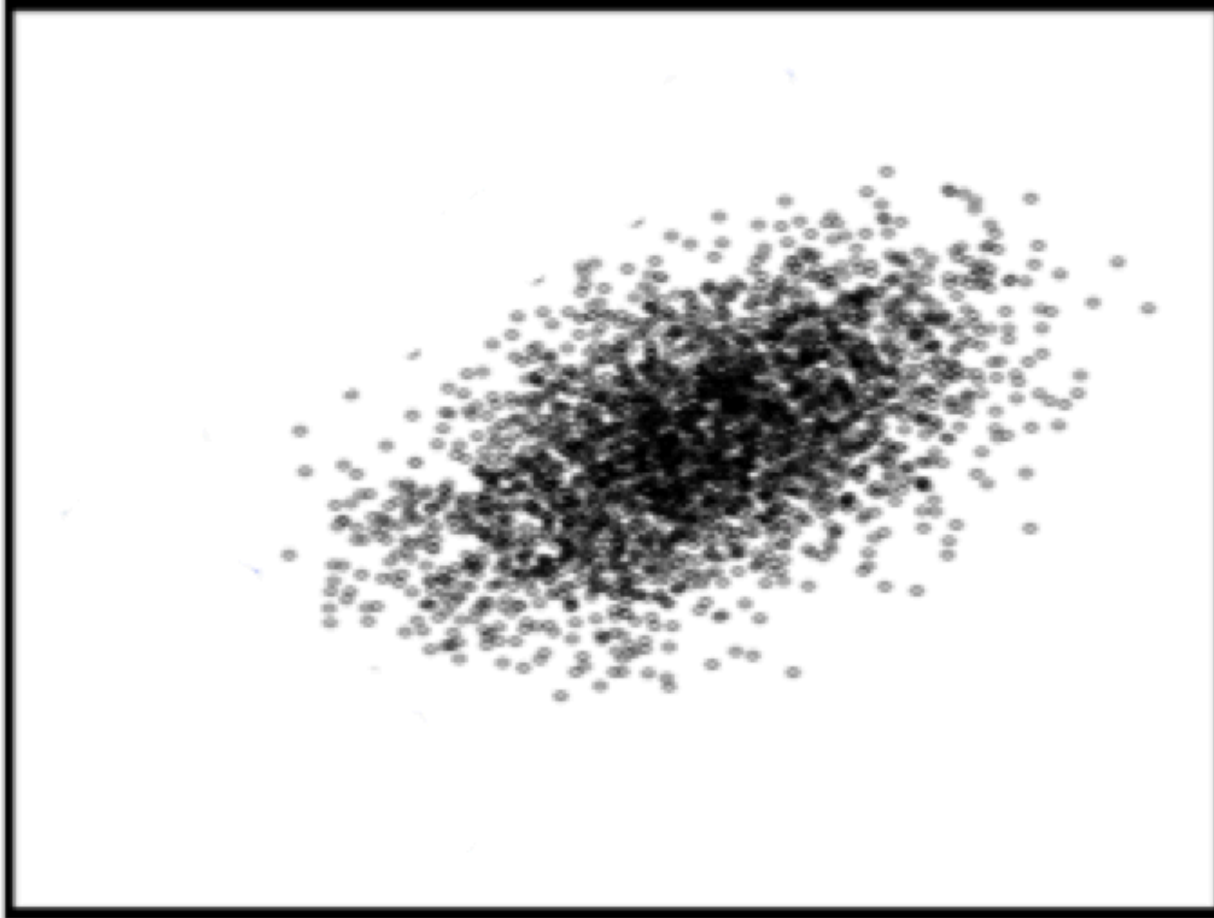
Summer 2017

Today's topics

- PCA and Dimensionality Reduction
- Spectral Decomposition
- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Generative Learning Algorithms
- Gaussian Discriminant Analysis
- Cholesky decomposition only if time permits (otherwise as reading materials.)

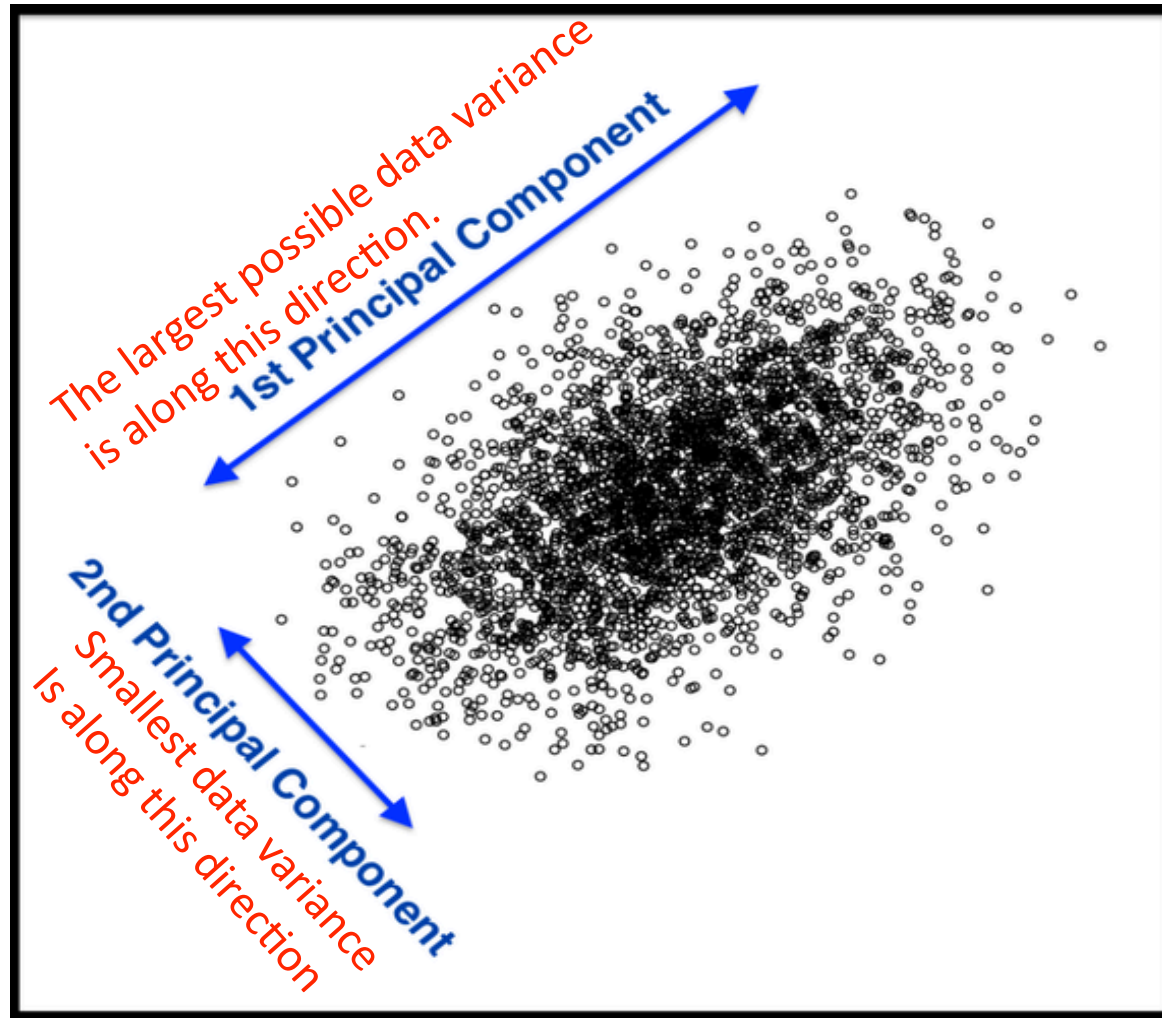
Principal Component Analysis (PCA)

An intuitive example

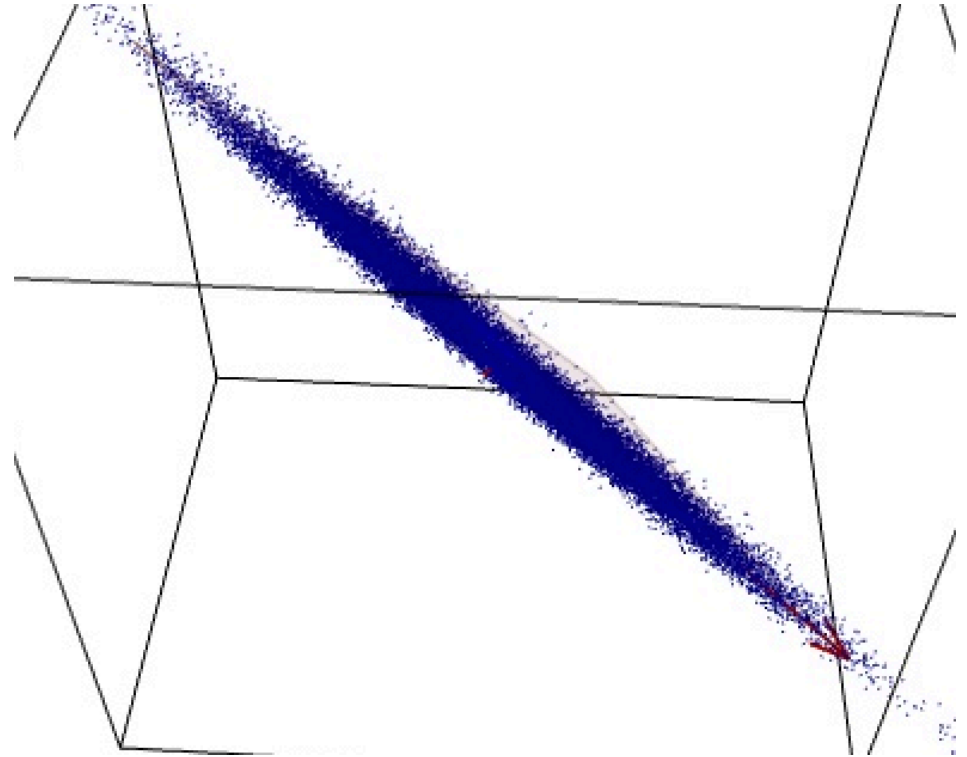
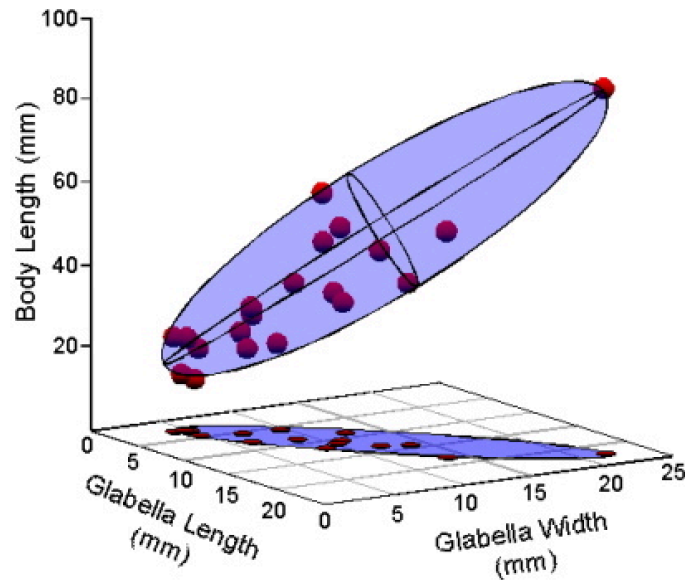


What kind of geometric curve will include almost all the data above?

Degree of Data Variation in different directions



Similarly in Higher Dimensional



All the data can be projected to
1-dimensional line!

Dimension Reduction

Almost all of the variation in the data is from left to right



2-D



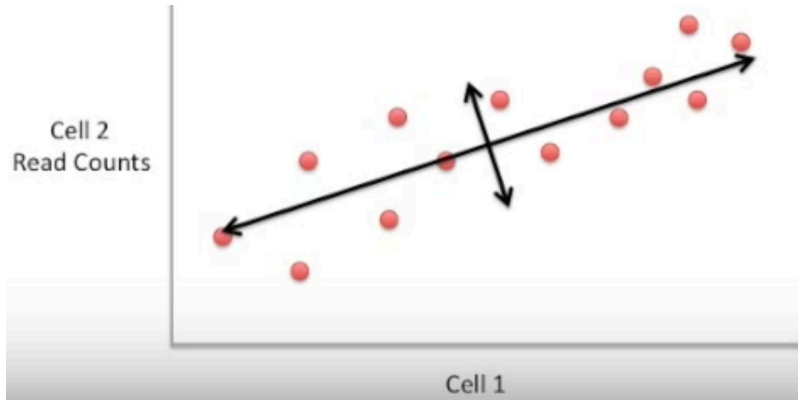
1-D

If we flattened the data (removed the up/down variation), it wouldn't look much different.

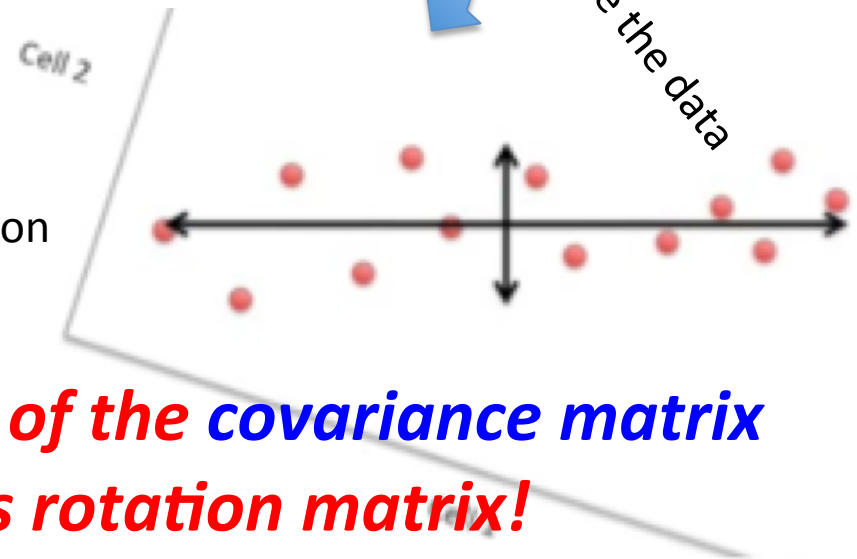


The number of principal components is less than or equal to the number of original variables.

Need to Identify the Rotation Matrix



Need the rotation matrix to rotate the data

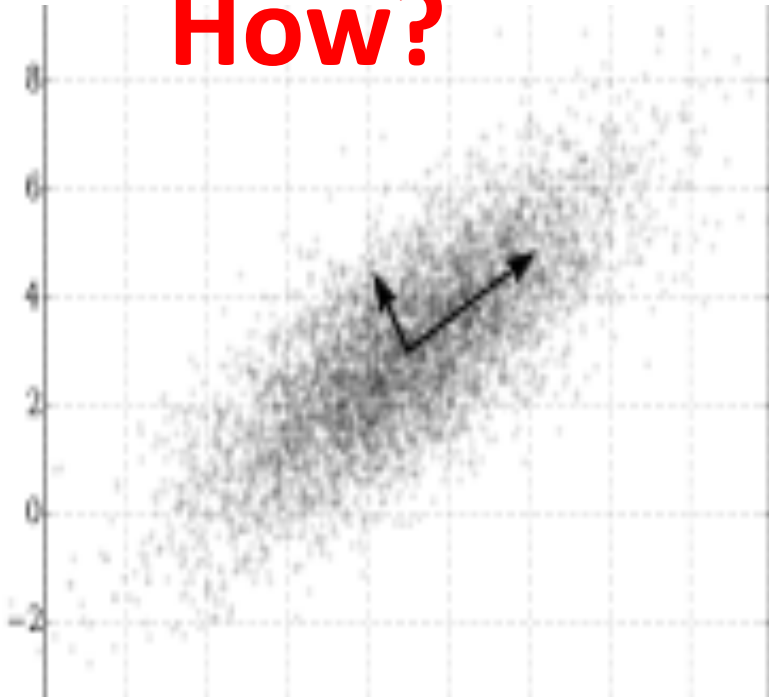


Key: How to find this orthogonal transformation
i.e. the orthogonal matrix?

***Use orthogonal eigenvectors of the covariance matrix
of the data as columns of this rotation matrix!***

Geometrically, we want to find two axis directions of the elliptic curve. They are called **principal axes**.

How?



Note: the x-value and y-value of the data are correlated.

Their correlation are reflected by the covariance matrix of the data.

- **Principal component analysis (PCA)** is a procedure to find an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated.
- **Procedure of find the principal directions:**
- Step 1: Find the covariance matrix of the data directly (Note: one can first standardize data:
 - Find the mean μ_1 of the x-value & the mean μ_2 of the y-value.
 - Subtract all x-value by μ_1 and subtract all y-value by μ_2 . *Geometrically, move the x-axis and y-axis to the data center.*)
- Step 2: Find eigenvalues and eigenvectors of the covariance matrix of the data.
- Step 3: Order the eigenvalues from largest to smallest. The eigenvector corresponding to the largest eigenvalue is called the 1st principal axis. So on and so forth.
- Step 4: Form the rotation matrix using the corresponding eigenvectors.

Recall Covariance Matrix

- The **covariance** between two rv's X and Y measures the degree to which X and Y are (linearly) related; defined as

$$\text{cov}[X, Y] \triangleq \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

Exercise

$$= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

If you standardize the data first, then it is equivalent to change coordinates to make them to 0.

If \mathbf{x} is a d -dimensional random vector, its **covariance matrix** is defined to be the following symmetric, positive definite matrix:

$$\text{cov}[\mathbf{x}] \triangleq \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^T]$$

Often denoted by Σ

$$= \begin{pmatrix} \text{var}[X_1] & \text{cov}[X_1, X_2] & \cdots & \text{cov}[X_1, X_d] \\ \text{cov}[X_2, X_1] & \text{var}[X_2] & \cdots & \text{cov}[X_2, X_d] \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}[X_d, X_1] & \text{cov}[X_d, X_2] & \cdots & \text{var}[X_d] \end{pmatrix}$$

correlation coefficient & correlation matrix

- The (Pearson) **correlation coefficient** between two rvs X and Y is defined as

$$\text{corr}[X, Y] \triangleq \frac{\text{cov}[X, Y]}{\sqrt{\text{var}[X] \text{var}[Y]}}$$

- If X and Y are indep., then $\text{cov}[X, Y] = 0$; say X and Y are uncorrelated.
- A **correlation matrix** of a random vector has the form:

$$\mathbf{R} = \begin{pmatrix} \text{corr}[X_1, X_1] & \text{corr}[X_1, X_2] & \cdots & \text{corr}[X_1, X_d] \\ \vdots & \vdots & \ddots & \vdots \\ \text{corr}[X_d, X_1] & \text{corr}[X_d, X_2] & \cdots & \text{corr}[X_d, X_d] \end{pmatrix}$$

Exercise: show that $-1 \leq \text{corr}[X, Y] \leq 1$ and

Show that $\text{corr}[X, Y] = 1$ iff $Y = aX + b$ for some parameters a and b .

Why do some people use XX^T for PCA?

And others just use data matrix X instead?

Recall: $\text{cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$

If each variable has a finite set of equal-probability values, x_i and y_i respectively for $i = 1, \dots, n$, then the covariance can be equivalently written in terms of the means $E(X)$ and $E(Y)$ as

$$\text{cov}(X, Y) = \frac{1}{n} \sum_{i=1}^n (x_i - E(X))(y_i - E(Y)).$$

If the data is standardized, then $E(X) = E(Y) = 0$, then

$$\text{cov}(X, Y) = (1/n) X^T Y.$$

The covariance matrix will be $(1/n) X^T X$.

Eigenvalues of $(1/n) X^T X$ will be just certain uniform scale of $X^T X$. It would not affect the ascent order the eigenvalues if people use $X^T X$.

If one uses the data matrix X directly, then one has to do a SVD. For details, see SVD next. ($X = USV^T$, and $X^T X = VDV^T$, So V is the one.)

Recall for the normal equation, we want to find

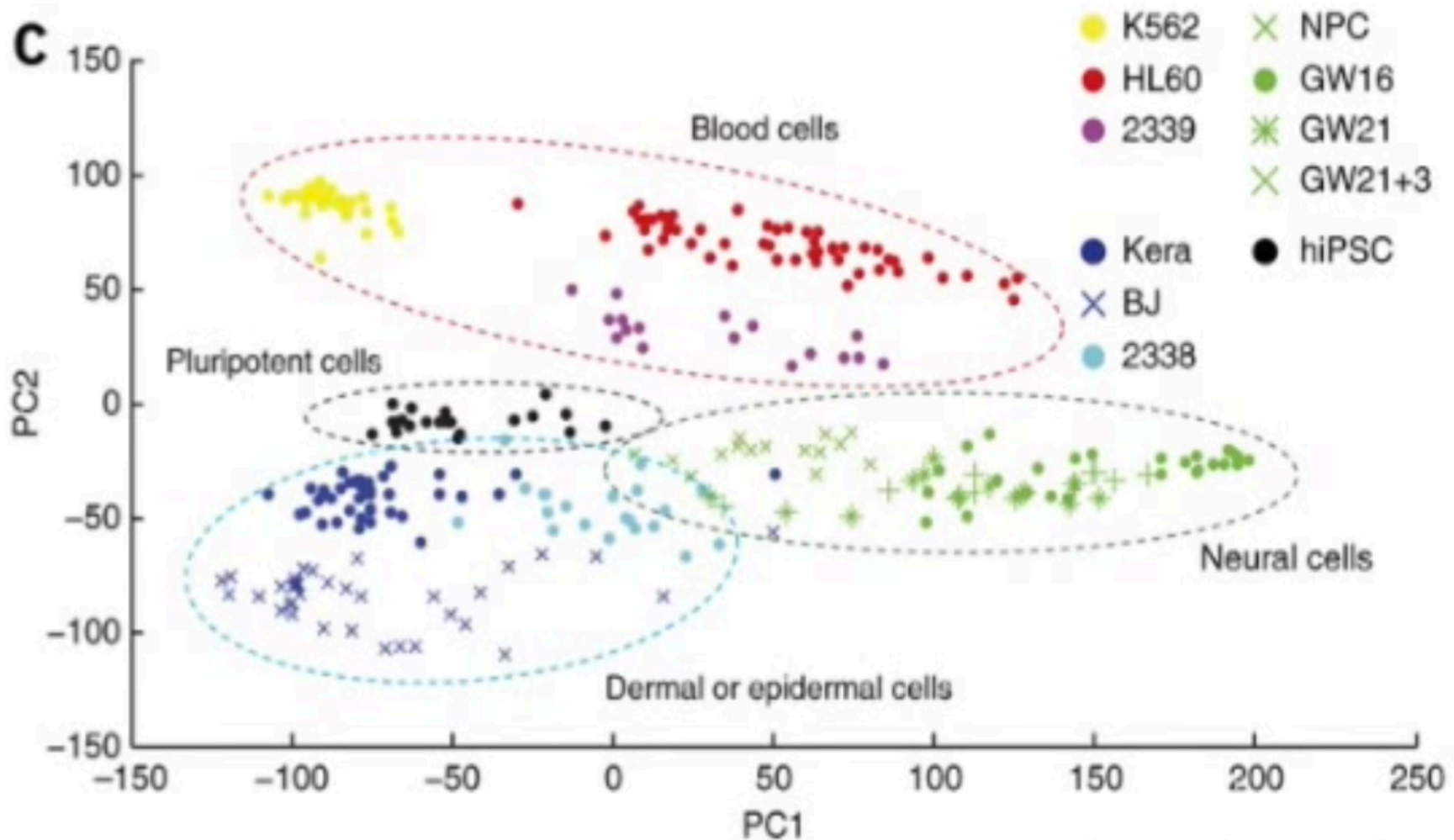
$$(X^T X)^{-1}$$

to solve for θ

$$\theta = (X^T X)^{-1} X^T \vec{y}.$$

If we write $X^T X = P^T D P$ (i.e. the spectral decomposition), where P is orthogonal, i.e. $P^{-1} = P^T$. Then $(X^T X)^{-1} = P^T D^{-1} P$. D^{-1} is very easy to calculate.

Principal Component Analysis (PCA) has a lots of applications in Biotechnology.



Singular Value Decomposition (SVD)

- What is SVD?
- It is a generalization of the notion of eigenvectors from *square* matrices to *any kind of matrix*.
- If A is a diagonalizable *square* matrix, then $A = PDP^{-1}$.
- If A is symmetric, then $A = PDP^T$. Where $P^T = P^{-1}$ (i.e. $PP^T = I$)
Here, D = diagonal matrix with eigenvalues on the diagonal
And P = Each column is the corresponding eigenvector

- X = (real) $N \times D$ matrix, then

$$\underbrace{\mathbf{X}}_{N \times D} = \underbrace{\mathbf{U}}_{N \times N} \underbrace{\mathbf{S}}_{N \times D} \underbrace{\mathbf{V}^T}_{D \times D}$$

The columns of U are called the left singular vectors, and the columns of V are called the right singular vectors.


$U = N \times N$ orthogonal matrix. i.e. $U^T U = U^T U = I$

$V = D \times D$ orthogonal matrix. i.e. $V^T V = V V^T = I$), and

$S = N \times D$ matrix containing the $r = \min(N, D)$ **singular values** $\sigma^i \geq 0$ on the main diagonal, with 0s filling the rest of the matrix.

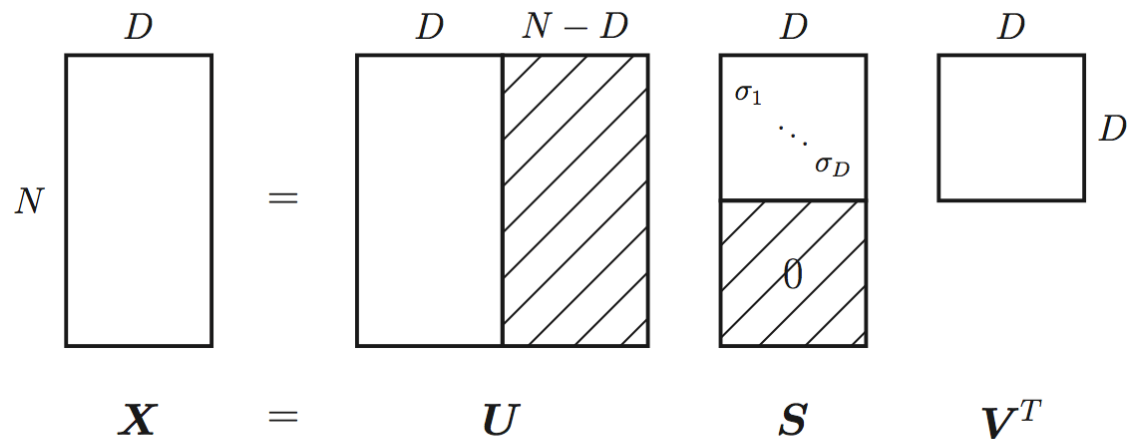
How to decompose X?

What are the singular values?

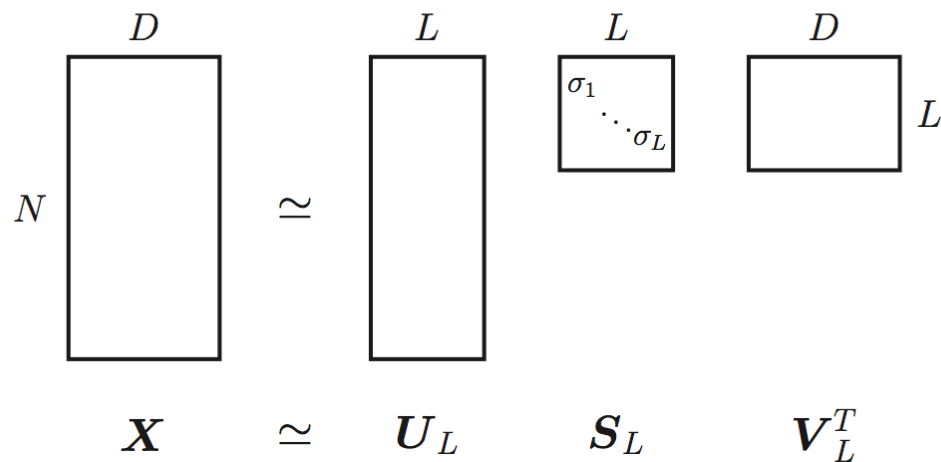
- Since $X^T X$ is symmetric, so there exist an orthogonal matrix V such that
 $X^T X = V D V^T$, where $V = \text{evec}(X^T X)$

eigenvalues
- But XX^T is also symmetric, so there exist an orthogonal matrix U such that
 $XX^T = U D_1 U^T$, where $U = \text{evec}(XX^T)$.
- How D and D_1 are related? **Claim: XX^T and $X^T X$ have the same *nonzero* eigenvalues!**
 - Let $\text{eval}(XX^T)$ = the set of non zero eigenvalues of XX^T
 - and $\text{eval}(X^T X)$ = the set of non zero eigenvalues of $X^T X$, we can show $\text{eval}(XX^T) = \text{eval}(X^T X)$.

Theorem: XX^T and $X^T X$ have the same none zero eigenvalues.

- That is to show $\text{eval}(XX^T) = \text{eval}(X^T X)$.
- Work out details with the students on the board.
- Now you order the eigenvalues of XX^T , in the same way as you order the eigenvalues of $X^T X$ in an ascent order. Since all the eigenvalues are ≥ 0 , then following the non zero eigenvalues there should be all zero eigenvalues.
- Now you find the U with the column putting eigenvectors corresponding to the ordered eigenvalues. Then U will be unique.
- Now $X = VDV^T = VD^{1/2}D^{1/2}V$, we want to stick the U in the middle. But U is $N \times N$. We have to be careful to make the dimension match. So we change $D^{1/2}$ to $S = N \times D$ matrix containing the $r = \min(N, D)$ singular values $\sigma^i \geq 0$, where σ^i is the square root the corresponding eigenvalue. Then stick $U^T U$, which is an identity matrix, in the middle. Please see the figure on the next slide.



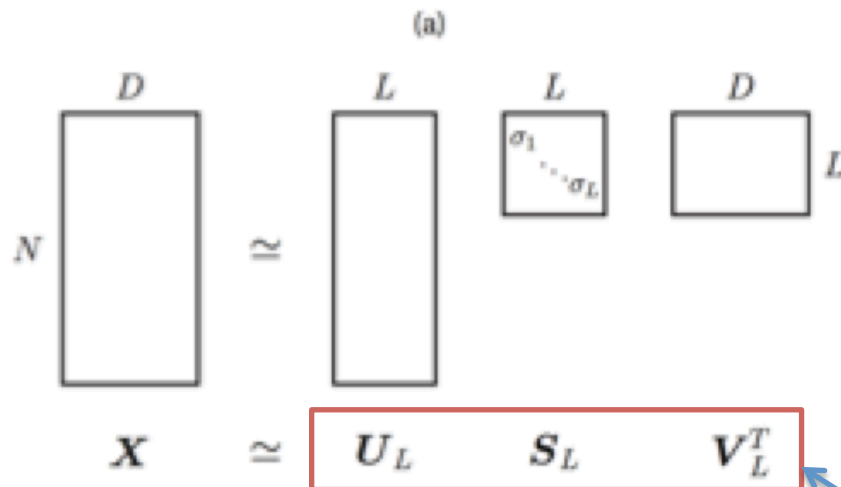
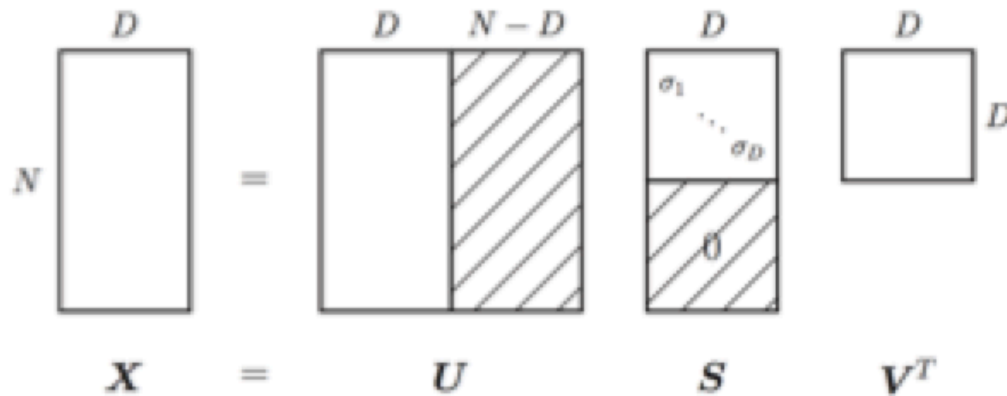
(a)



(b)

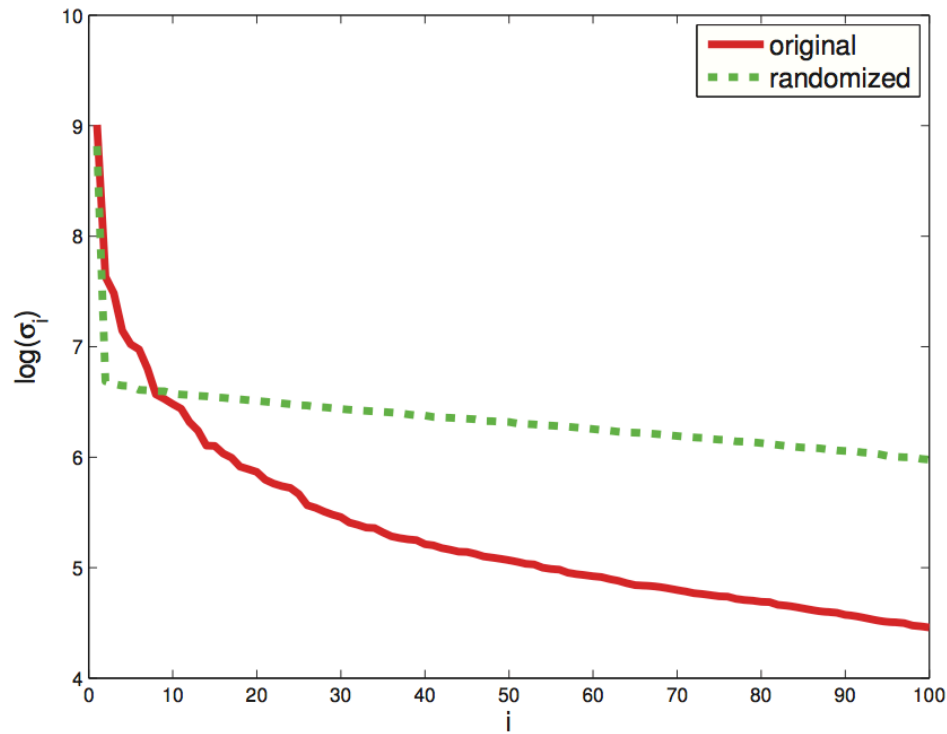
Figure 12.8 (a) SVD decomposition of non-square matrices $\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T$. The shaded parts of \mathbf{S} , and all the off-diagonal terms, are zero. The shaded entries in \mathbf{U} and \mathbf{S} are not computed in the economy-sized version, since they are not needed. (b) Truncated SVD approximation of rank L .

Application: SVD as a dimension-reduction technique



- **Example:** Data compression
- Rank the eigenvalues in descent order. Set all the small eigenvalues (equivalently singular values) to zero, you will compress the data.

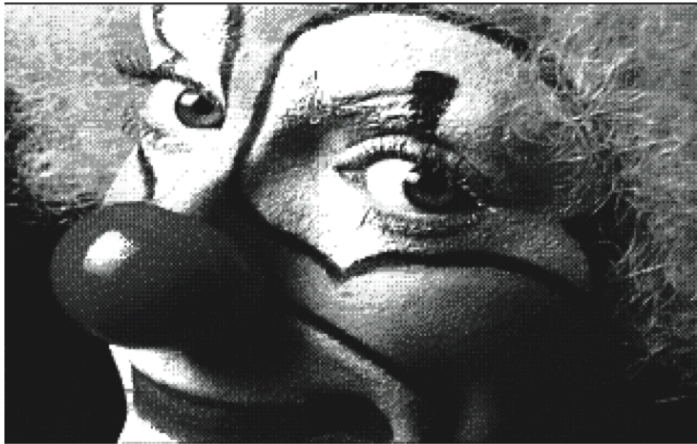
The error in this approximation is $\|X - X_L\|_F \approx \sigma_{L+1}$



12.10 First 50 log singular values for the clown image (solid red line), and for a data d by randomly shuffling the pixels (dotted green line). Figure generated by `svdImageDemo`.

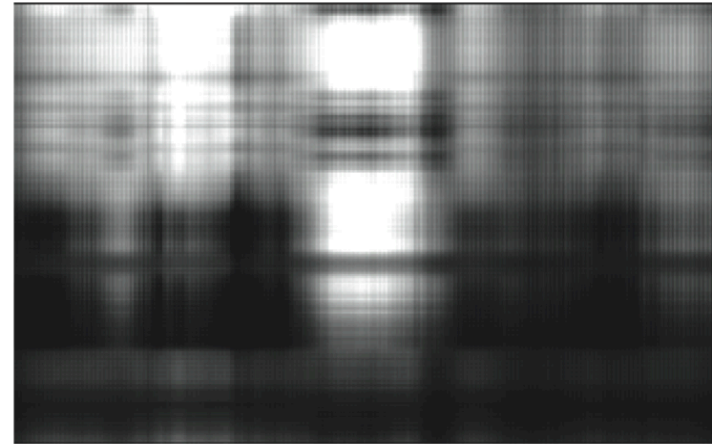
Consider the 200×320 pixel image. This has 64,000 numbers in it. We see that a rank 20 approximation, with only $(200 + 320 + 1) \times 20 = 10,420$ numbers is a very good approximation.

rank 200



(a)

rank 2



(b)

rank 5



(c)

rank 20



(d)

Rank = 20 means setting all $\sigma_L = 0$ for $L \geq 21$. Error is about σ_{21} .

Figure 12.9 Low rank approximations to an image. Top left: The original image is of size 200×320 , so has rank 200. Subsequent images have ranks 2, 5, and 20. Figure generated by `svdImageDemo`.

Data Compression

- If your boss wants to compress a data set, you may ask him/her what would be the threshold of an error acceptable. (Since naturally, compressing data will loss information, it is reasonable to ask the threshold of an acceptable error.)
- If your boss say 5%, you perform an SVD on your data, and find the singular value σ_L . Say $L = 30$, which is close to 5% and then set all $\sigma_L = 0$ for $L \geq 31$. You know after the data compress, your error is about σ_{31} which is about 5%.

$$X \approx \begin{bmatrix} U_L & S_L & V_L^T \end{bmatrix}$$


This sizes of those matrices are much smaller.

Changing gear:

Generative learning Algorithm

- Work out details with the students on the board.

A Different Approach, called **Generative** Learning Algorithm

- First, looking at cats, we can build a model of what cats look like.
- Then, looking at dogs, we can build a separate model of what dogs look like.
- Finally, to classify a new animal, we can match the new animal against the cat model, and match it against the dog model, to see whether the new animal looks more like the cats or more like the dogs we had seen in the training set.

Gaussian Discriminant Analysis (GDA) Model

- Work out details with the students on the board.

- **Change gear to techniques to design algorithm for fast computation.**
- (Only if time permits. Otherwise, please read the following slides instead.)

Recall we studied earlier:

Computational Techniques such as gradient descent and Newton's method

- For big data, the closed formula involving find inverse a huge matrix, it may not be the fastest or most stable way.
- Other Computational methods used:
 - Cholesky decomposition
 - Gradient descent
 - Stochastic gradient descent
 - Newton's method

How to make the computation fast?

Cholesky Decomposition

Theorem: If A is a $n \times n$ real, symmetric and positive definite matrix then there exists a unique lower triangular matrix G with positive diagonal element such that

$$A = GG^T.$$

Proof: Work out details with the students on the board.

Recall: If A is an upper triangular matrix, then $AX = b$ can be solved by backward substitution.

Upper Triangular Matrix:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Backward substitution

$$x_i = (b_i - \sum_{j=i+1}^n a_{ij} \cdot x_j) / a_{ii}$$

$$\begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

$$x_i = (b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j) / a_{ii}$$

Similarly when A is lower triangular.

If $A = LU$, then $Ax = b$ can be solved quickly.

- Then $Ax = b$ can be written as
- $LUx = b$
- Let $y = Ux$
- Then $Ly = b$, solve y first
- Then use $Ux = y$ to solve for x .

Another way to find Cholesky decomposition

Another way to find out the cholesky decomposition

Suppose

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

We need to solve
the equation

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \underbrace{\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}}_L \underbrace{\begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & l_{22} & \cdots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & l_{nn} \end{bmatrix}}_{L^T}$$

Example of find Cholesky Decomposition

Suppose

$$A = \begin{bmatrix} 4 & 2 & -2 \\ 2 & 10 & 2 \\ -2 & 2 & 5 \end{bmatrix}$$

Then Cholesky Decomposition

For k from 1 to n

$$l_{kk} = \left(a_{kk} - \sum_{s=1}^{k-1} l_{ks}^2 \right)^{1/2}$$

For j from $k+1$ to n

$$l_{jk} = \left(a_{jk} - \sum_{s=1}^{k-1} l_{js} l_{ks} \right) / l_{kk}$$

Entry	General formula	Output
l_{11}	$\sqrt{a_{11}}$	2
l_{21}	a_{21}/l_{11}	1
l_{31}	a_{31}/l_{11}	-1
l_{22}	$\sqrt{a_{22} - l_{21}^2}$	3
l_{32}	$(a_{32} - l_{21}l_{31})/l_{22}$	1
l_{33}	$\sqrt{a_{33} - l_{31}^2 - l_{32}^2}$	$\sqrt{3}$

Now,

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 3 & 0 \\ -1 & 1 & \sqrt{3} \end{bmatrix}$$

The third way to do Cholesky decomposition

- $\mathbf{PAP}^T = \mathbf{D}$, where P is an orthogonal matrix. (Why?)
- *Work out more details with the students on the board.*
- *Note: This is a relatively expensive method if you want to use determinant to find eigenvalues.*

Here are some R code to find Cholesky Decomposition

- `x<-matrix(c(4,2,-2, 2,10,2, -2,2,5), ncol=3, nrow=3)`
- `cl<-chol(x)`
- **If we Decompose A as LDL^T then**

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ -1/2 & 1/3 & 1 \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Applications of Cholesky Decomposition

- We use Cholesky Decomposition to solve the system of linear equation $Ax=b$, where A is real symmetric and positive definite.
- In regression analysis, we also use it to estimate the parameter if $X^T X$ is positive definite.
- In Kernel principal component analysis, we also need to use Cholesky decomposition. (Weiya Shi; Yue-Fei Guo; 2010)