

UVA CS 6316: Machine Learning

Lecture 16: Principal Component Analysis (PCA)

Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Where are we ? →

Five major sections of this course

- ❑ Regression (supervised)
- ❑ Classification (supervised)
 - ❑ Feature selection
- ❑ Unsupervised models
 - ❑ Dimension Reduction (PCA)
 - ❑ Clustering (K-means, GMM/EM, Hierarchical)
- ❑ Learning theory
- ❑ Graphical models

	X_1	X_2	X_3
S_1			
S_2			
S_3			
S_4			
S_5			
S_6			

d

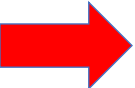
a data matrix of n observations on p variables x_1, x_2, \dots, x_p

Unsupervised learning = learning from raw (unlabeled, unannotated, etc) data, as opposed to supervised data where a classification/regression label of examples is given

- **Data/points/instances/examples/samples/records:** [rows]
- **Features/attributes/dimensions/independent variables/covariates/predictors/regressors:** [columns]

Today

■ Dimensionality Reduction (unsupervised) with Principal Components Analysis (PCA)

- 
- Review of eigenvalue, eigenvector
 - How to project samples into a line capturing the variation of the whole dataset → Eigenvector / Eigenvalue of covariance matrix
 - PCA for dimension reduction
 - Eigenface → PCA for face recognition

Review: Mean and Variance

- Variance:

$$\text{Var}(X) = E((X - \mu)^2)$$

- Discrete RVs:

- Continuous RVs:

$$V(X) = \sum_{v_i} (v_i - \mu)^2 P(X = v_i)$$

$$V(X) = \int_{-\infty}^{+\infty} (x - \mu)^2 f(x) dx$$

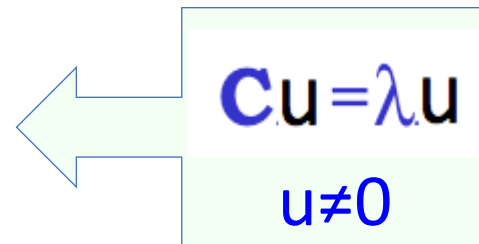
- Covariance:

$$\text{Cov}(X, Y) = E((X - \mu_x)(Y - \mu_y)) = E(XY) - \mu_x \mu_y$$

Review: Eigenvector / Eigenvalue

- The eigenvalues λ_i are found by solving the equation

$$\det(C - \lambda I) = 0$$


$$Cu = \lambda u$$
$$u \neq 0$$

- Eigenvectors are columns of the matrix U such that

$$C = UDU^T$$

- Where

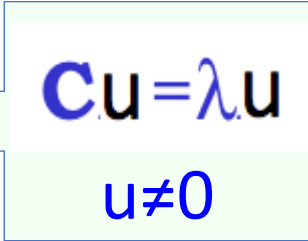
$$D = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & & & \\ 0 & \dots & \dots & \lambda_p \end{pmatrix}$$

Review: Eigenvalue, e.g.

- Let us take two variables with covariance $c > 0$

- $\mathbf{C} = \begin{pmatrix} 1 & c \\ c & 1 \end{pmatrix}$ $\mathbf{C} - \lambda \mathbf{I} = \begin{pmatrix} 1 - \lambda & c \\ c & 1 - \lambda \end{pmatrix}$

$$\det(\mathbf{C} - \lambda \mathbf{I}) = (1 - \lambda)^2 - c^2$$


$$\mathbf{C}u = \lambda u$$
$$u \neq 0$$

- Solving this we find $\lambda_1 = 1 + c$

$$\lambda_2 = 1 - c < \lambda_1$$

Review: Eigenvector, e.g.

- Any eigenvector \mathbf{u} satisfies the condition

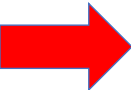
$$\mathbf{C}\mathbf{u} = \lambda\mathbf{u}$$

$$\mathbf{u} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad \mathbf{C}\mathbf{u} = \begin{pmatrix} 1 & c \\ c & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} a_1 + ca_2 \\ ca_1 + a_2 \end{pmatrix} = \begin{pmatrix} \lambda a_1 \\ \lambda a_2 \end{pmatrix}$$

Solving we find $\mathbf{u}_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$, $\mathbf{u}_2 = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$

Today

■ Dimensionality Reduction (unsupervised) with Principal Components Analysis (PCA)

- Review of eigenvalue, eigenvector
-  □ How to project samples into a line capturing the variation of the whole dataset → Eigenvector / Eigenvalue of covariance matrix
- PCA for dimension reduction
- Eigenface → PCA for face recognition

	X_1	X_2	X_3
S_1			
S_2			
S_3			
S_4			
S_5			
S_6			

d

a data matrix of n observations on p variables x_1, x_2, \dots, x_p

- **Data/points/instances/examples/samples/records:** [rows]
- **Features/attributes/dimensions/independent variables/covariates/predictors/regressors:** [columns]

The Goal

We wish to **explain/summarize the underlying variance-covariance structure of a large set of variables** through a few linear combinations of these variables.

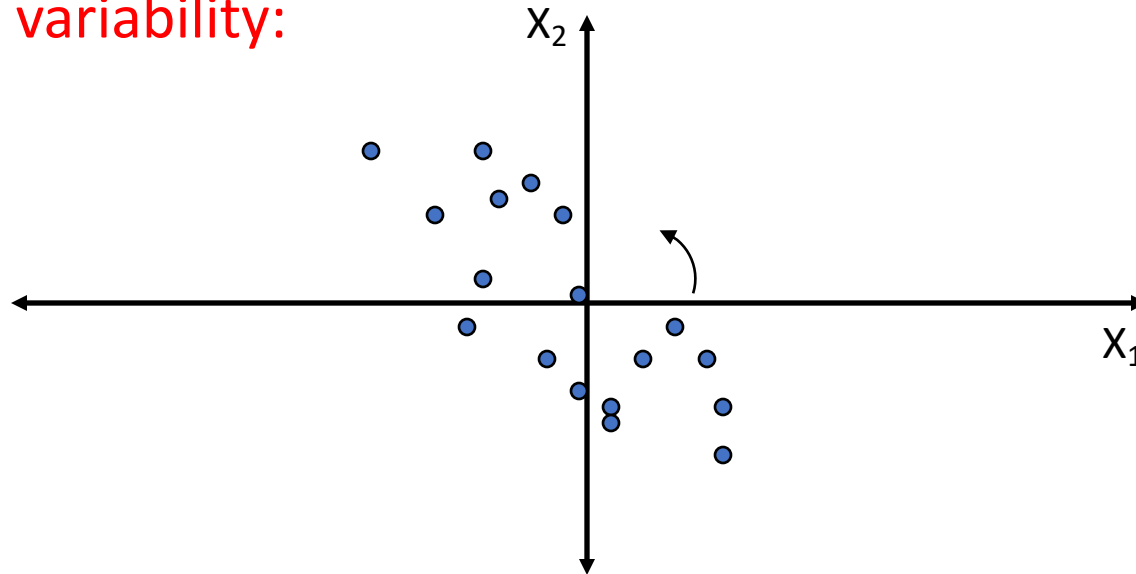
PCA is introduced by Pearson (1901) and Hotelling (1933)

Trick: Rotate Coordinate Axes

Suppose we have a sample population measured on p random variables X_1, \dots, X_p .

Our goal is to develop a new set of K ($K < p$) axes

(linear combinations of the original p axes) **in the directions of greatest variability:**

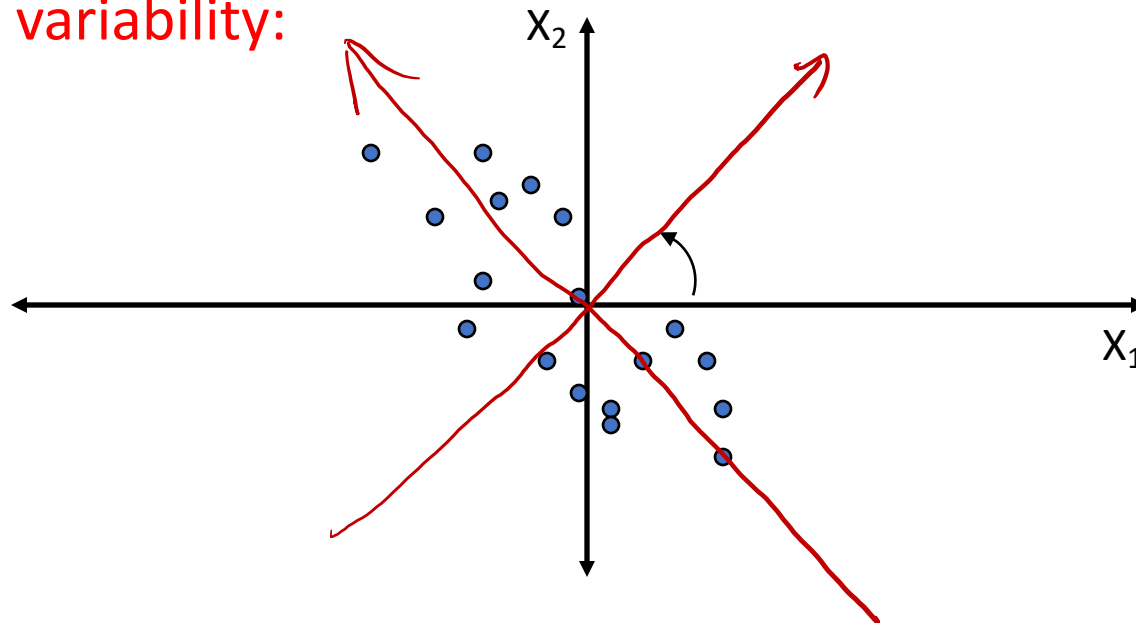


This could be accomplished by rotating the axes (if data is centered).

Trick: Rotate Coordinate Axes

Suppose we have a sample population measured on p random variables X_1, \dots, X_p .

Our goal is to develop a new set of K ($K < p$) axes (linear combinations of the original p axes) in the directions of greatest variability:



This could be accomplished by rotating the axes (if data is centered).

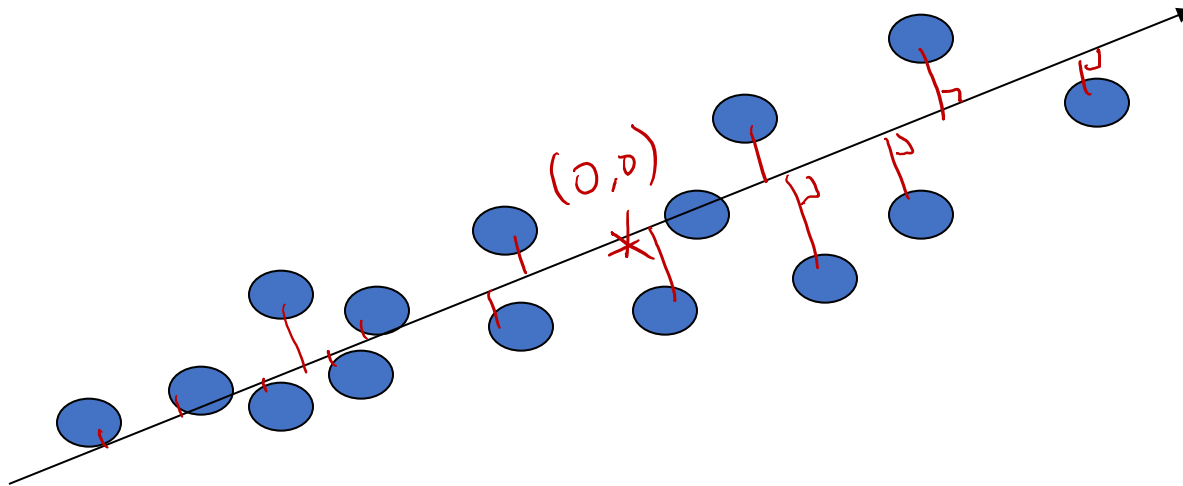
Algebraic Interpretation

- Given n points in a p dimensional space,
- for large p , how to **project** on to **a lower-dimensional ($K < p$)** space while preserving **broad trends** in the data and allowing it to be visualized?

FROM NOW we assume Data matrix is centered: \rightarrow (we subtract the mean along each dimension, and center the original axis system at the centroid of all data points, for simplicity)

Algebraic Interpretation – (k=1)

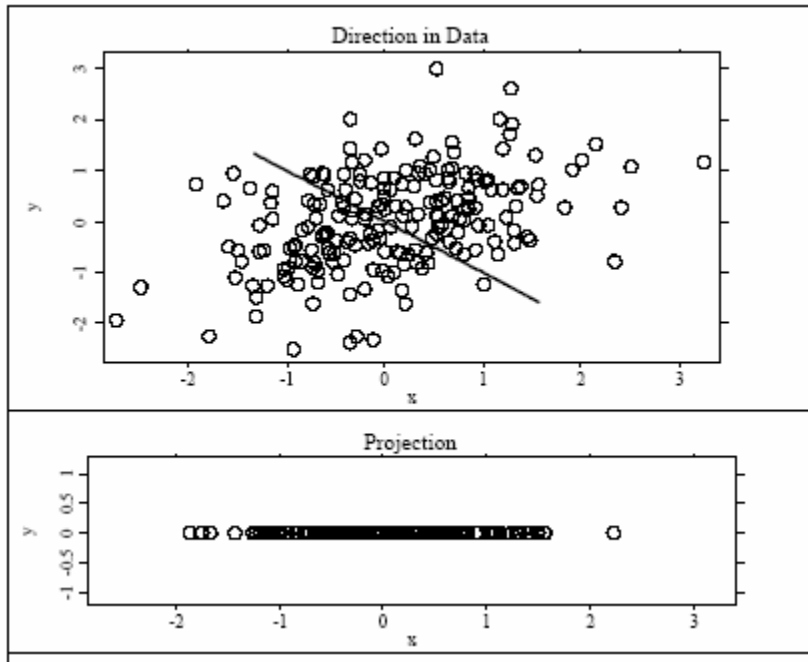
- Given n points in a p dimensional space, how to project **on to a 1 dimensional** space?



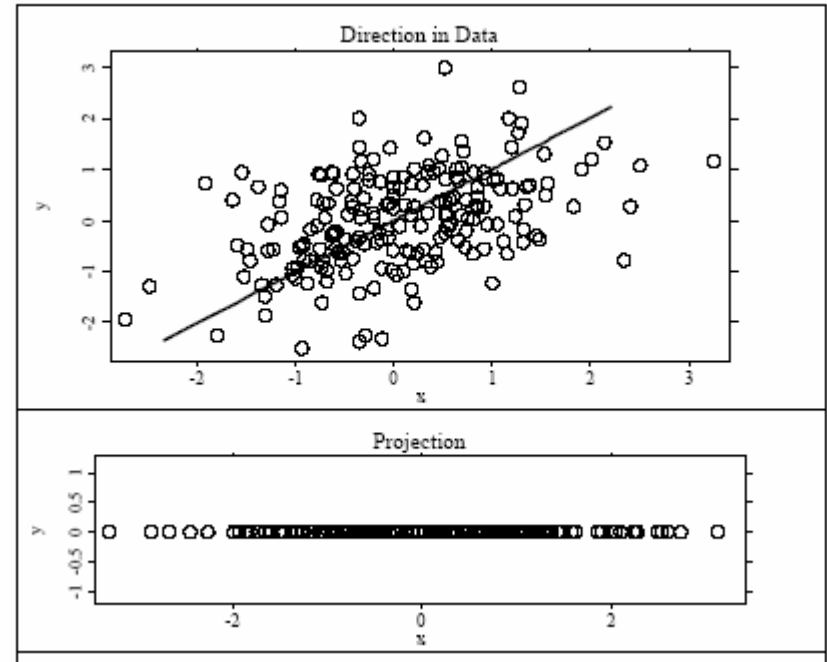
- Choose a line that fits the data so **the points are spread out well along the line**

Let us see it on a figure

Good

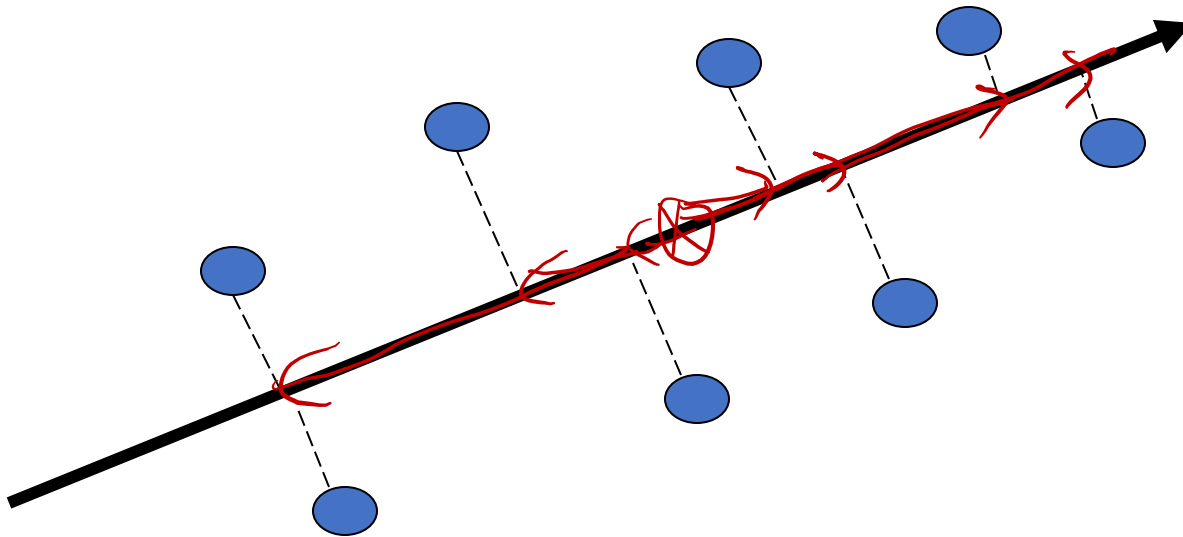


Better



Algebraic Interpretation – (k=1)

- Formally, to find a line that \rightarrow Maximizing the sum of squares of data samples' projections on that line



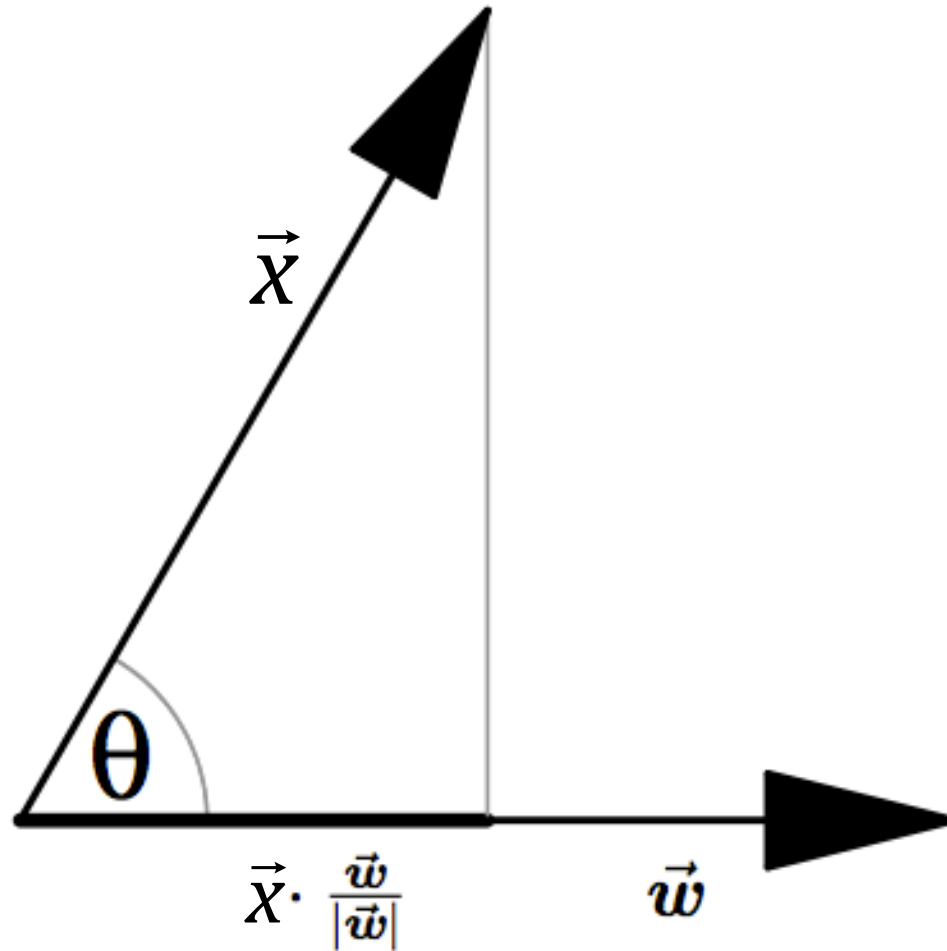
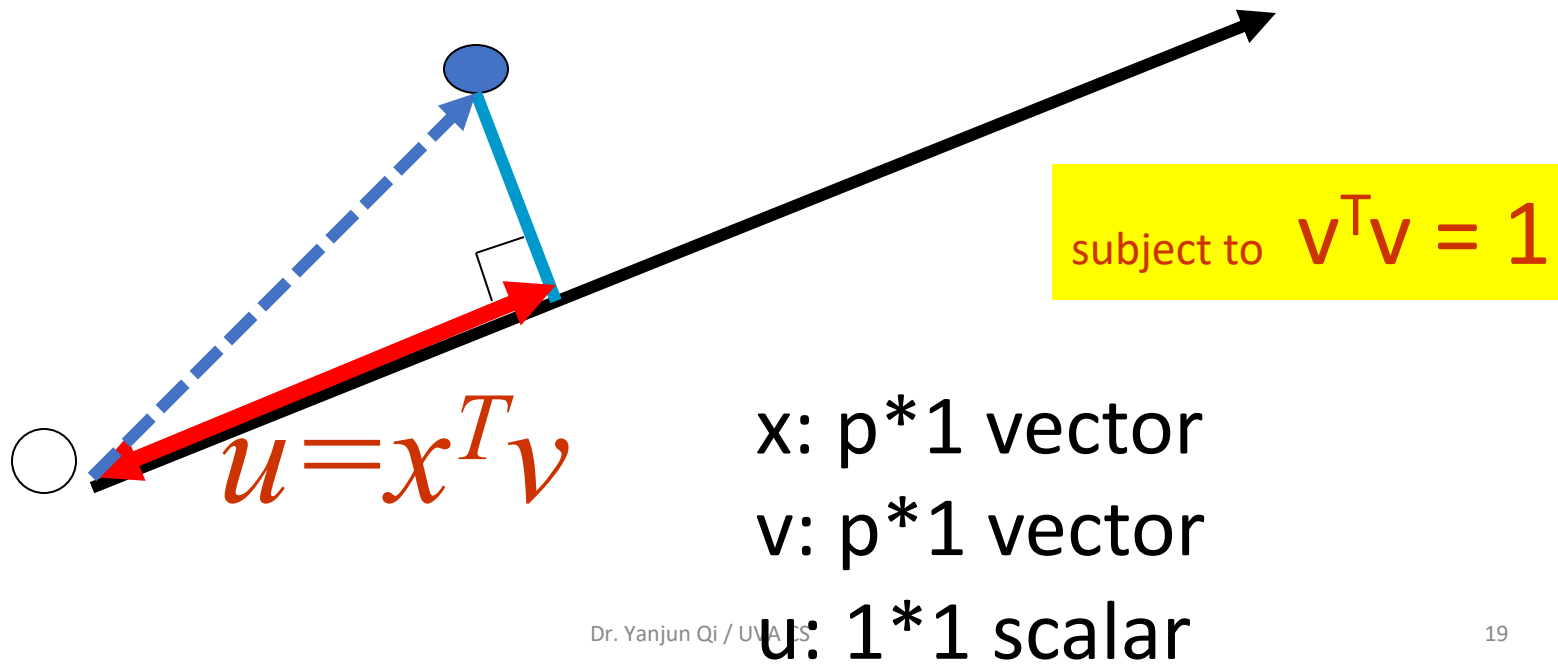


Figure 1: The dot product is fundamentally a projection.

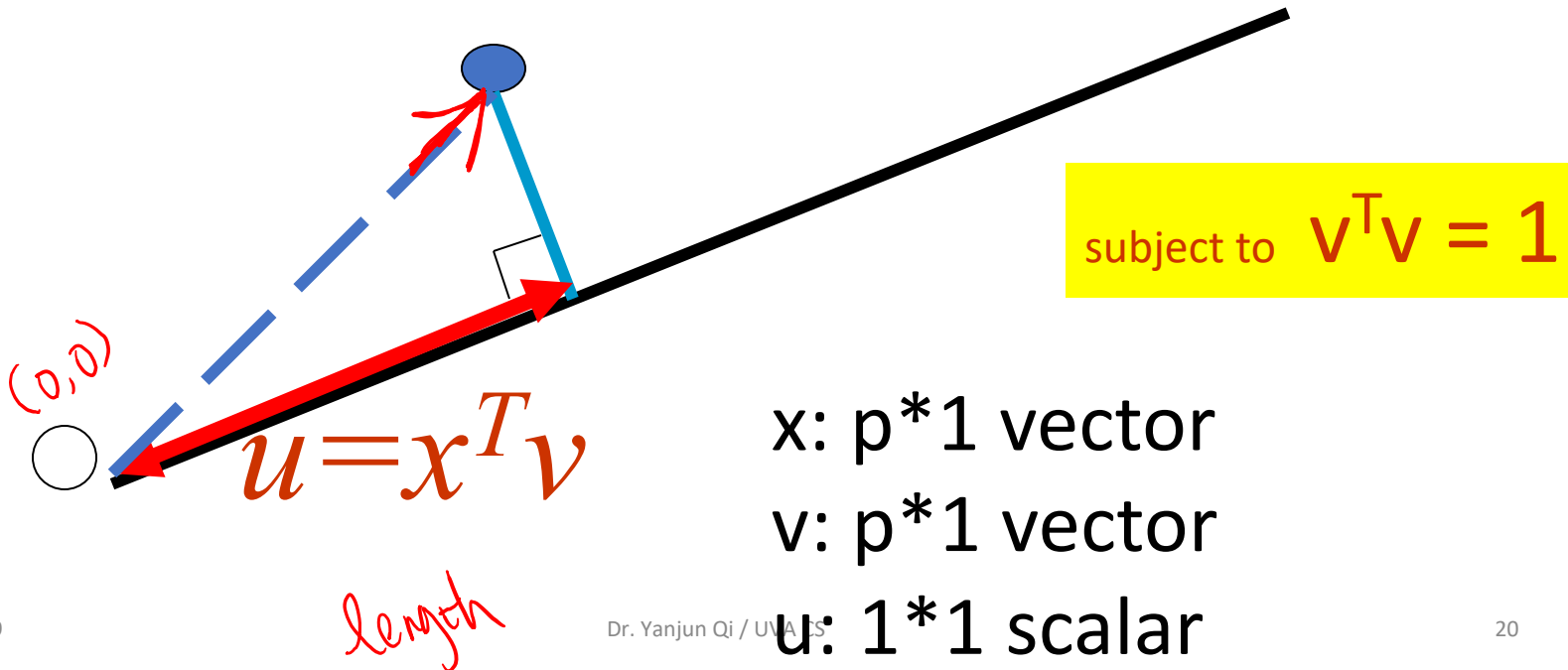
Algebraic Interpretation – 1D

- Formally, to find a line (direction) that → Maximizing the sum of squares of data samples' projections on that line



Algebraic Interpretation – 1D

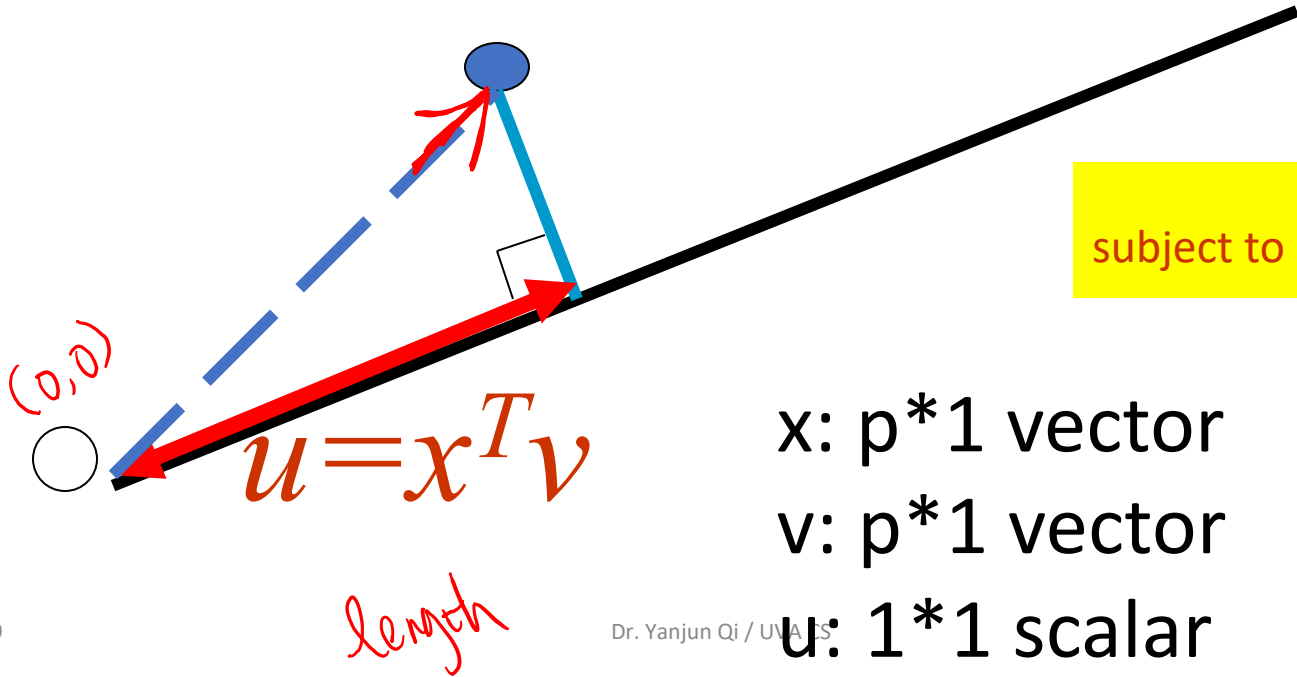
- Formally, to find a line (direction) that → Maximizing the sum of squares of data samples' projections on that line



Algebraic Interpretation – 1D

- Formally, to find a line (direction) that → Maximizing the sum of squares of data samples' projections on that line

size of x 's projection on vector v → $u = x^T v = v^T x$



subject to $v^T v = 1$

x : $p \times 1$ vector
 v : $p \times 1$ vector
 u : 1×1 scalar

Algebraic Interpretation

- 1D

$$\arg \max_v \sum u_i (u_i)^2$$

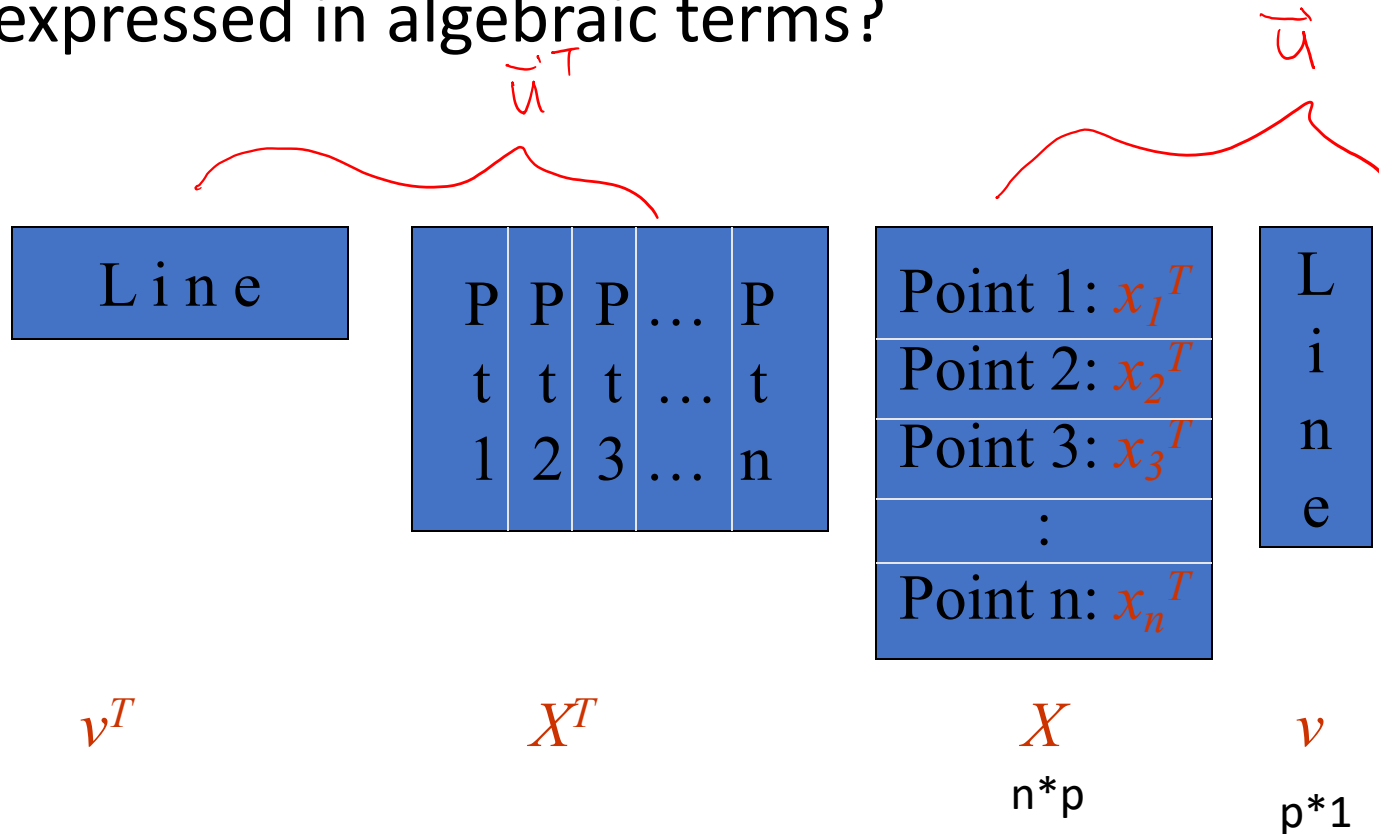
$$\max \left\{ \sum_{i=1}^n u_i^2 \right\} =$$

$$= [u_1, u_2, u_3, \dots, u_n] \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \vec{u}^T \vec{u} \quad \vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

Here $\vec{u} =$ vector $\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} x_1^T v \\ x_2^T v \\ \vdots \\ x_n^T v \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} v = \sum_{n \times p} v_{p \times 1}$

Algebraic Interpretation – 1D

- How is the sum of squares of projection lengths expressed in algebraic terms?



Algebraic Interpretation – 1D

- How is the sum of squares of projection lengths expressed in algebraic terms?

$$\max(\underbrace{v^T X^T X v}), \text{ subject to } v^T v = 1$$

Algebraic Interpretation – 1D

- Rewriting this:

$$\max(v^T X^T X v), \text{ subject to } v^T v = 1$$

$$v^T X^T X v = \lambda = \lambda v^T v = v^T (\lambda v)$$

$$\Leftrightarrow v^T (X^T X v - \lambda v) = 0$$

Algebraic Interpretation – 1D

- Rewriting this:

$$\max(\mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v}), \text{ subject to } \mathbf{v}^T \mathbf{v} = 1$$

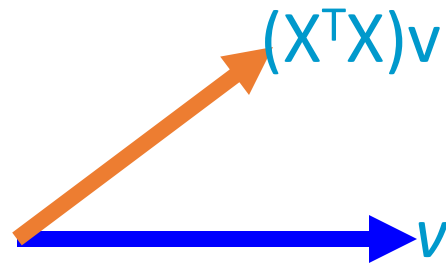
$$\mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v} = \lambda = \lambda \mathbf{v}^T \mathbf{v} = \mathbf{v}^T (\lambda \mathbf{v})$$

$$\Leftrightarrow \mathbf{v}^T (\mathbf{X}^T \mathbf{X} \mathbf{v} - \lambda \mathbf{v}) = 0$$


- Show that the maximum value of $\mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v}$ is obtained for those vectors / **directions** satisfying $\mathbf{X}^T \mathbf{X} \mathbf{v} = \lambda \mathbf{v}$
- So, find the largest λ and associated \mathbf{v} such that the matrix $\mathbf{X}^T \mathbf{X}$ when applied to \mathbf{v} , yields a new vector which is in the same direction as \mathbf{v} , only scaled by a factor λ .

Algebraic Interpretation – 1D

- $(X^T X)v$ points in some other direction (different from v) in general



→ If v is an eigenvector and λ is corresponding eigenvalue

$$X^T X v = \lambda v$$


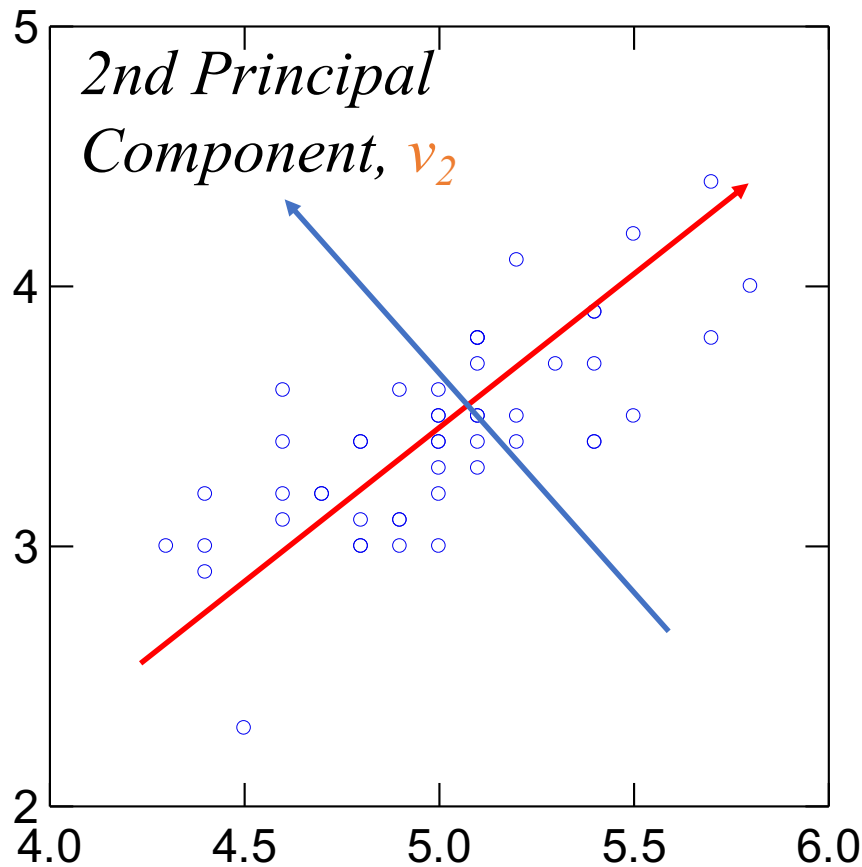
So, find the largest λ and associated v such that the matrix $X^T X$ when applied to v , yields a new vector which is in the same direction as v , only scaled by a factor λ .

Algebraic Interpretation – beyond 1D

- For matrices of the form (**symmetric**) $X^T X$
 - **All eigenvalues are non-negative**
 - See Handout-1 “linear algebra review” / Page 18,19,20
 - $\lambda_1 \dots \lambda_p$ are the eigenvalues, ordering from large to small,
 - *i.e. Ordered by the PC’s importance*

PCA Eigenvectors → Principal Components

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_p \quad [PCA]$$



PCA (k=1) : How the sum of squares of projection lengths relates to **VARIANCE** ?

size of one sample x 's projection on vector v

$$\rightarrow u = x^T v = v^T x$$

- In a new coordinate system with v as axis, u is the position of sample x on this axis

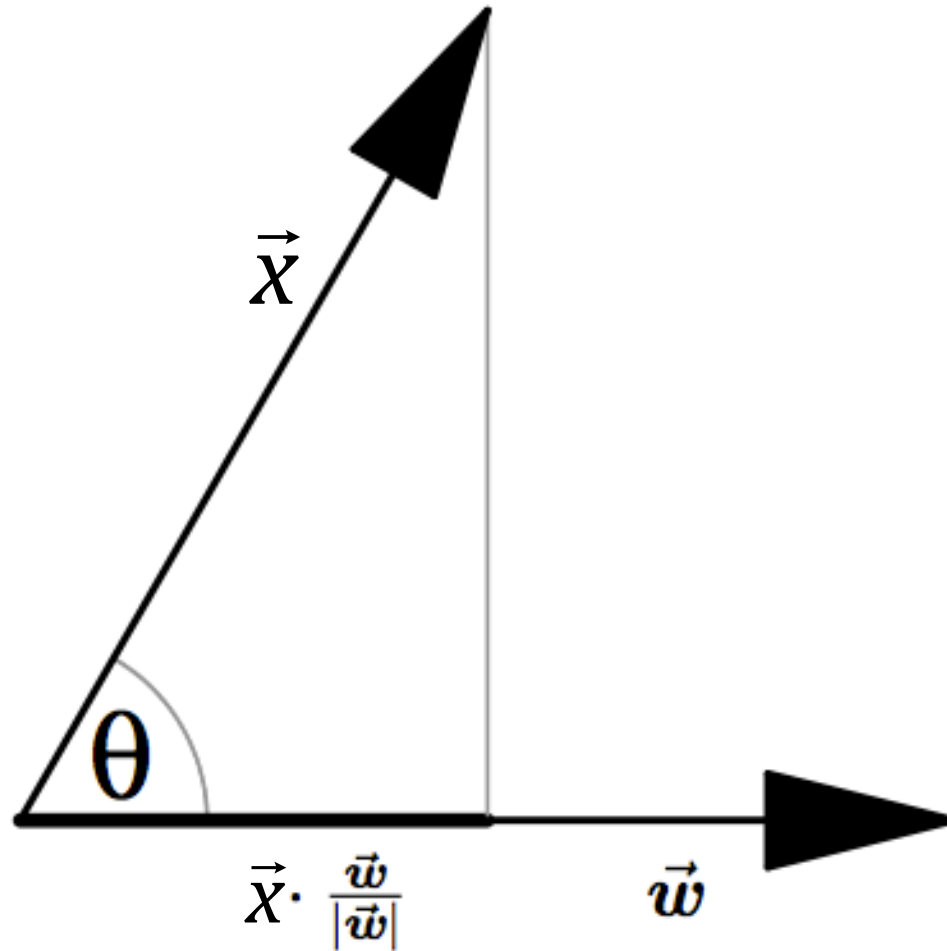
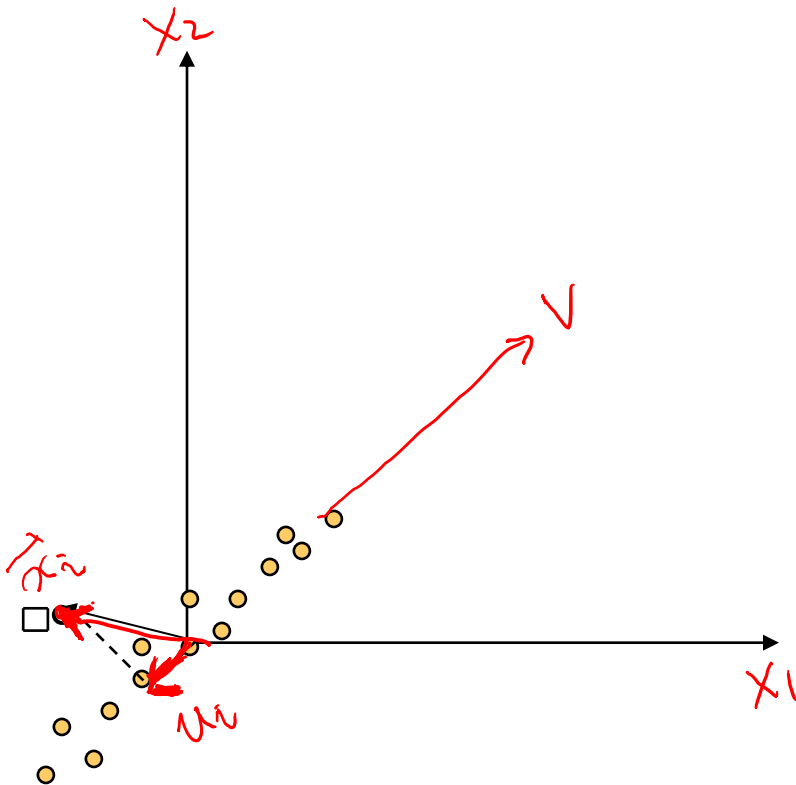


Figure 1: The dot product is fundamentally a projection.

PCA (k=1) : How the sum of squares of projection lengths relates to **VARIANCE** ?

Consider the **variation** along direction \mathbf{v} considering all of the points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$:

→ The variance of all positions $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n\}$



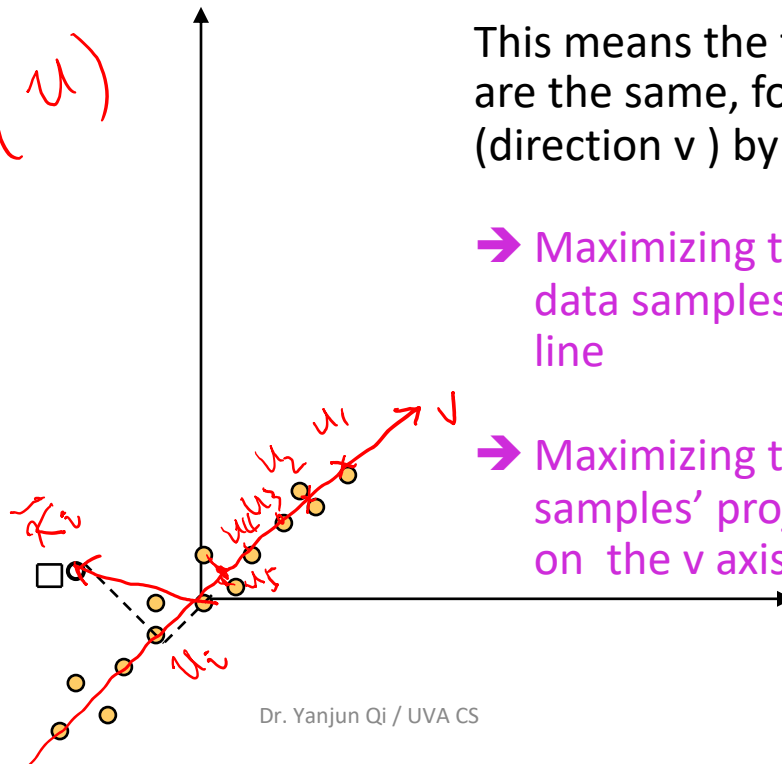
convert \mathbf{x}_i onto \mathbf{V} coordinate
→ $\mathbf{u}_i = (\mathbf{x}_i)^T \mathbf{V}$

How the sum of squares of projection lengths relates to **VARIANCE** ?

$$\text{Var}(u) = \sum_{u_i} (u_i - \mu)^2 P(u = u_i) = \sum_{u_i} (u_i)^2$$

Assuming centered data matrix

$\underset{v}{\text{argmax}} \sum u_i^2$
 $= \underset{v}{\text{argmax}} \text{Variance}(u)$



This means the following two objectives are the same, for finding a line (direction v) by

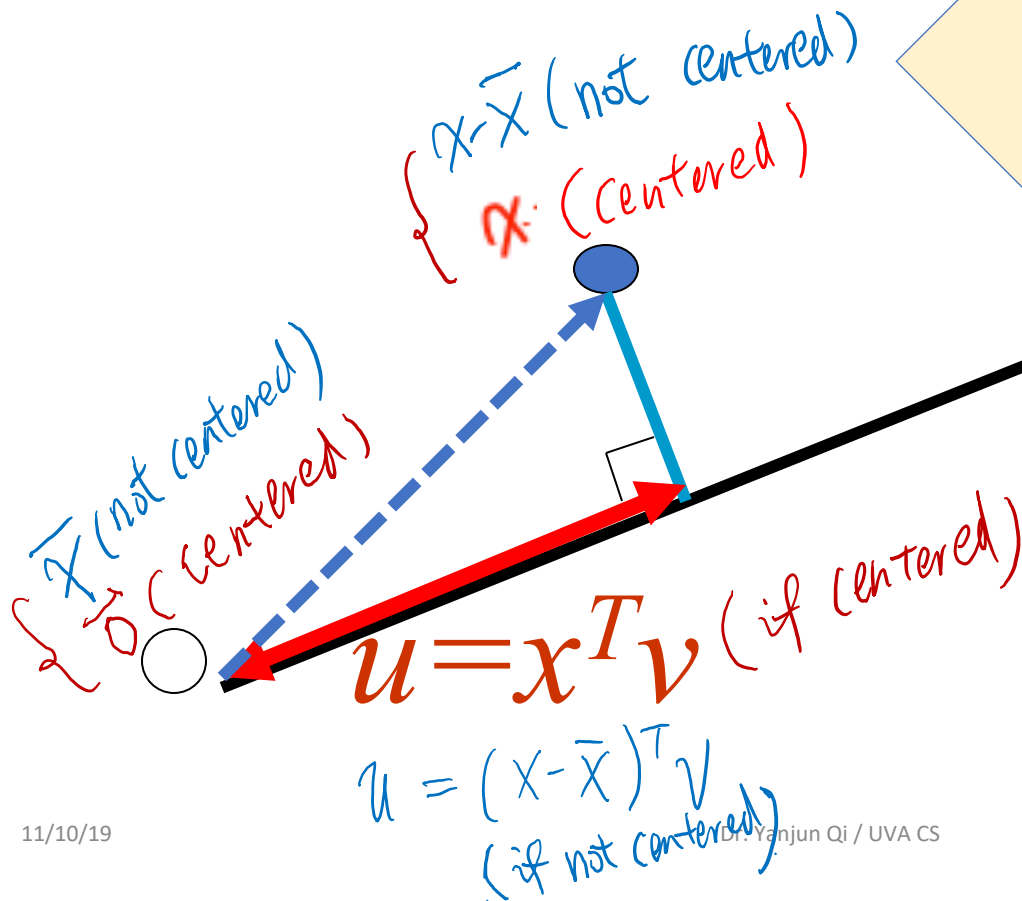
- ➔ Maximizing the sum of squares of data samples' projections on that v line
- ➔ Maximizing the variance of data samples' projected representations on the v axis

Centered Vs. Not Centered Formulation

Center $X_{n \times p}$ vs.
not-centered $X_{n \times p}$

subject to $\mathbf{v}^T \mathbf{v} = 1$

\mathbf{x} : $p \times 1$ vector
 \mathbf{v} : $p \times 1$ vector
 u : 1×1 scalar



Today

- Dimensionality Reduction (unsupervised) with Principal Components Analysis (PCA)
 - Review of eigenvalue, eigenvector
 - How to project samples into a line capturing the variation of the whole dataset → Eigenvector / Eigenvalue of covariance matrix
 - PCA for dimension reduction
 - Eigenface → PCA for face recognition

Applications

- Uses:

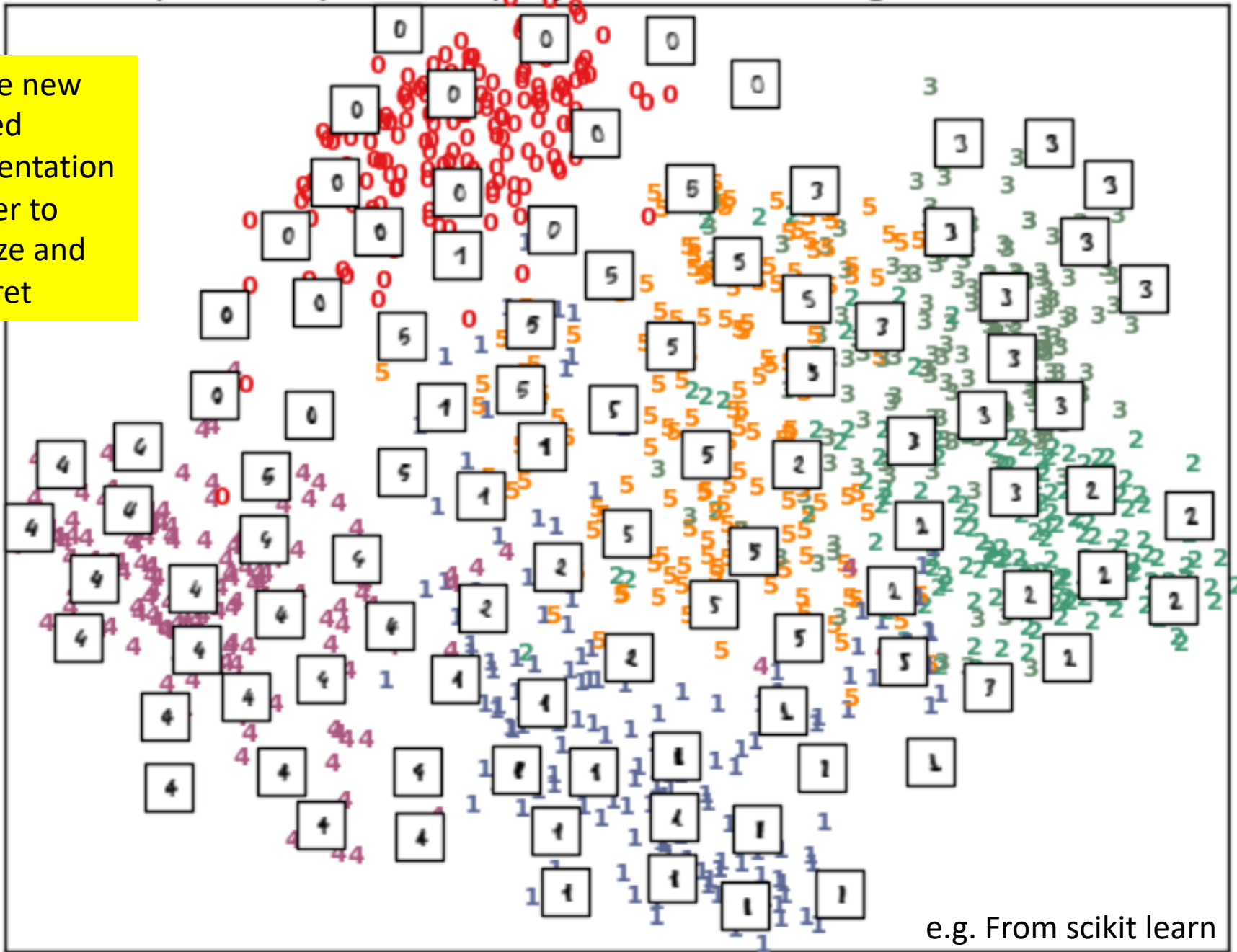
- Data Visualization
- Data Reduction
- Data Classification
- Trend Analysis
- Factor Analysis
- Noise Reduction

- Examples:

- How many unique “sub-sets” are in the sample?
- How are they similar / different?
- What are the underlying factors that influence the samples?
- How to best present what is “interesting”?
- Which “sub-set” does this new sample rightfully belong?
-

Principal Components projection of the digits (time 0.02s)

e.g. the new reduced representation is easier to visualize and interpret



e.g. From scikit learn

Interpretation of PCA

From p original coordinates: x_1, x_2, \dots, x_p :

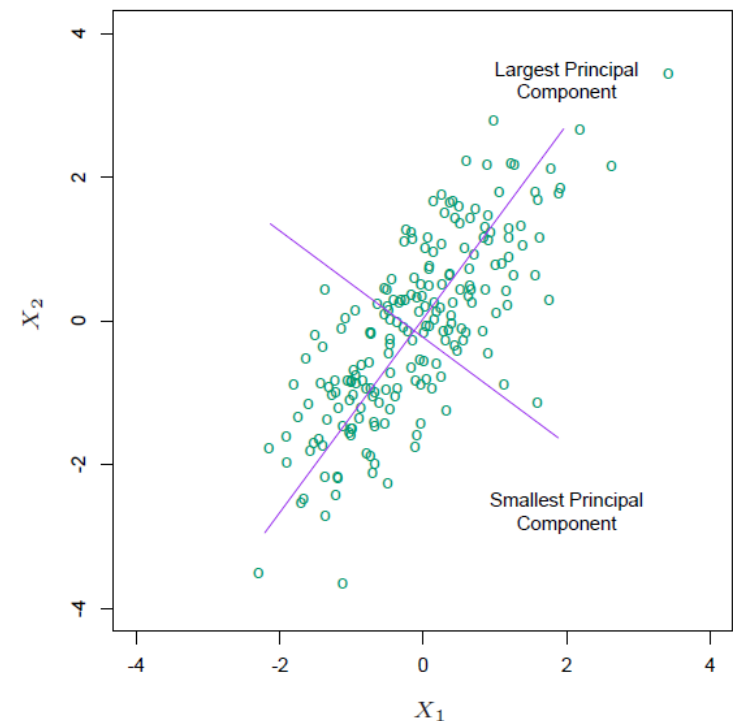
Produce k new coordinates : v_1, v_2, \dots, v_k :

$$v_1 = a_{11}x_1 + a_{12}x_2 + \dots + a_{1p}x_p$$

$$v_2 = a_{21}x_1 + a_{22}x_2 + \dots + a_{2p}x_p$$

...

When $p=2$



Interpretation of PCA

v_k 's are Principal Components

such that:

v_k are uncorrelated (orthogonal) from each other

v_1 explains as much as possible of original variance in data set

v_2 explains as much as possible of remaining variance

etc.

v_k : k^{th} PC retains the k^{th} greatest fraction of the variation in the samples

$$\text{Var}(u^k) = \sum_{i=1}^n (u^k_i)^2 = v_k^T X^T X v_k$$

- The new variables (PCs) have a variance equal to their corresponding eigenvalue, since

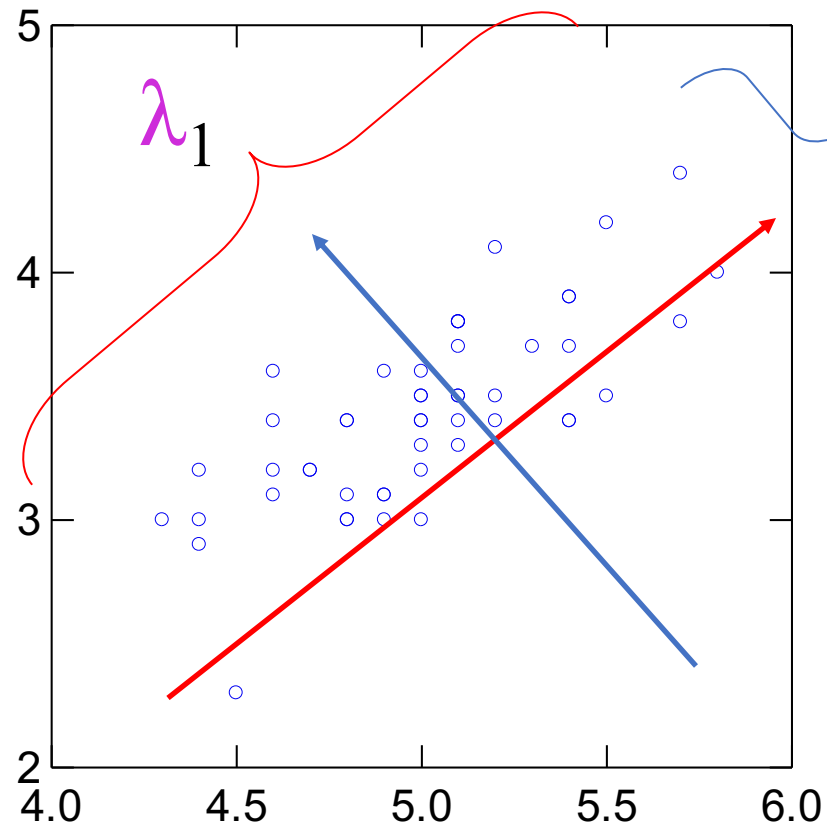
$$\text{Var}(u^k) = v_k^T X^T X v_k = v_k^T \lambda_k v_k = \lambda_k v_k^T v_k = \lambda_k$$

for all $k=1\dots p$

- Small $\lambda_k \Leftrightarrow$ small variance \Leftrightarrow data change little in the direction of component v_k

PCA is useful for finding new, more informative, uncorrelated features; it reduces dimensionality by rejecting low variance features

PCA Eigenvalues



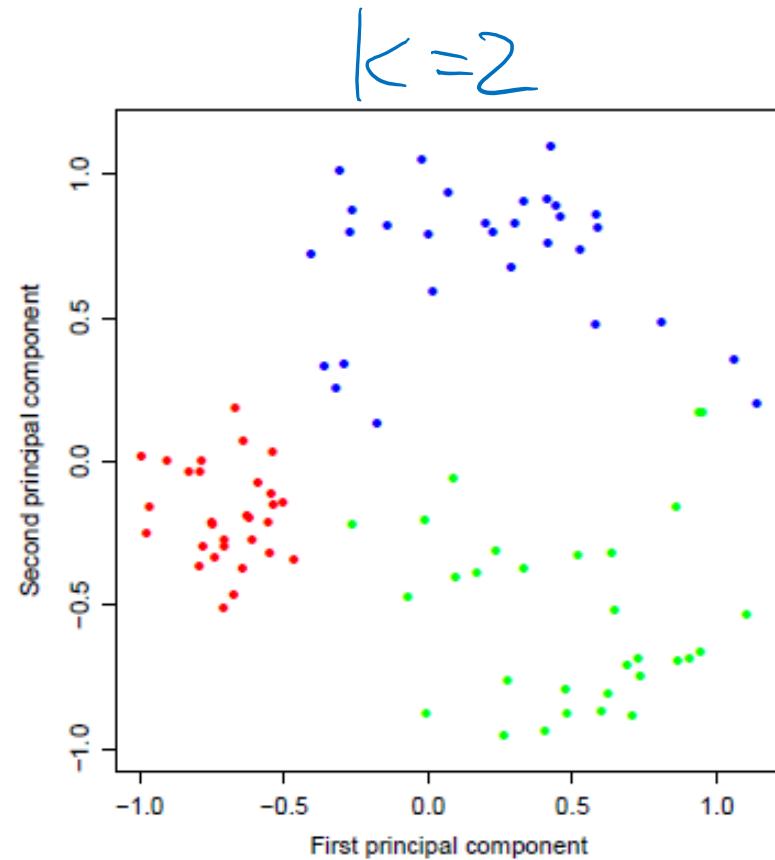
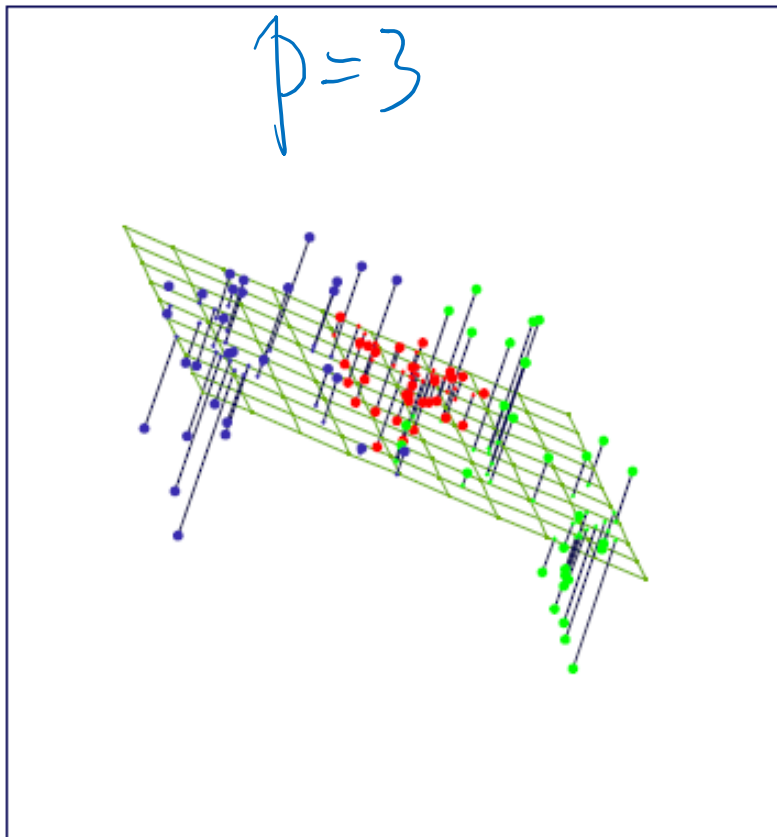
λ_2 = Variance
in this
2nd PC direction

PCA Summary until now

- Rotates multivariate dataset into a new configuration which is easier to interpret
- PCA is useful for finding new, more informative, uncorrelated features; it reduces dimensionality by rejecting low variance features

- ✓ PCA compresses (i.e. perform projection) the data points by only using the top few eigenvectors.
- ✓ This corresponds to choosing a “linear subspace” represent points on a line, plane, or “hyper-plane”

PCA for dimension reduction
e.g. $p=3 \rightarrow$ (pick top $k=2$ PCs)

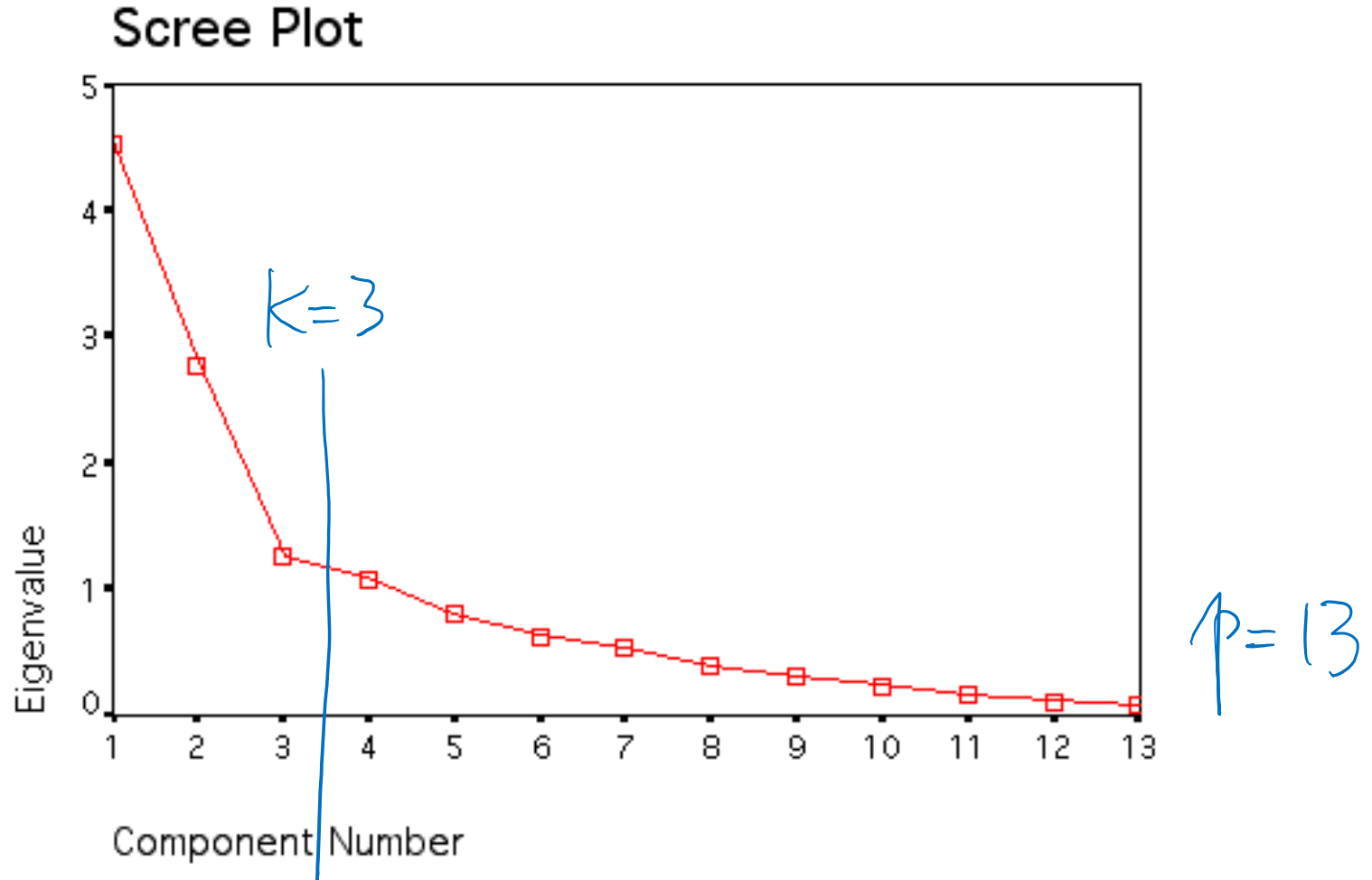


corresponds to choosing a
“2D linear plane”

How many components to keep?

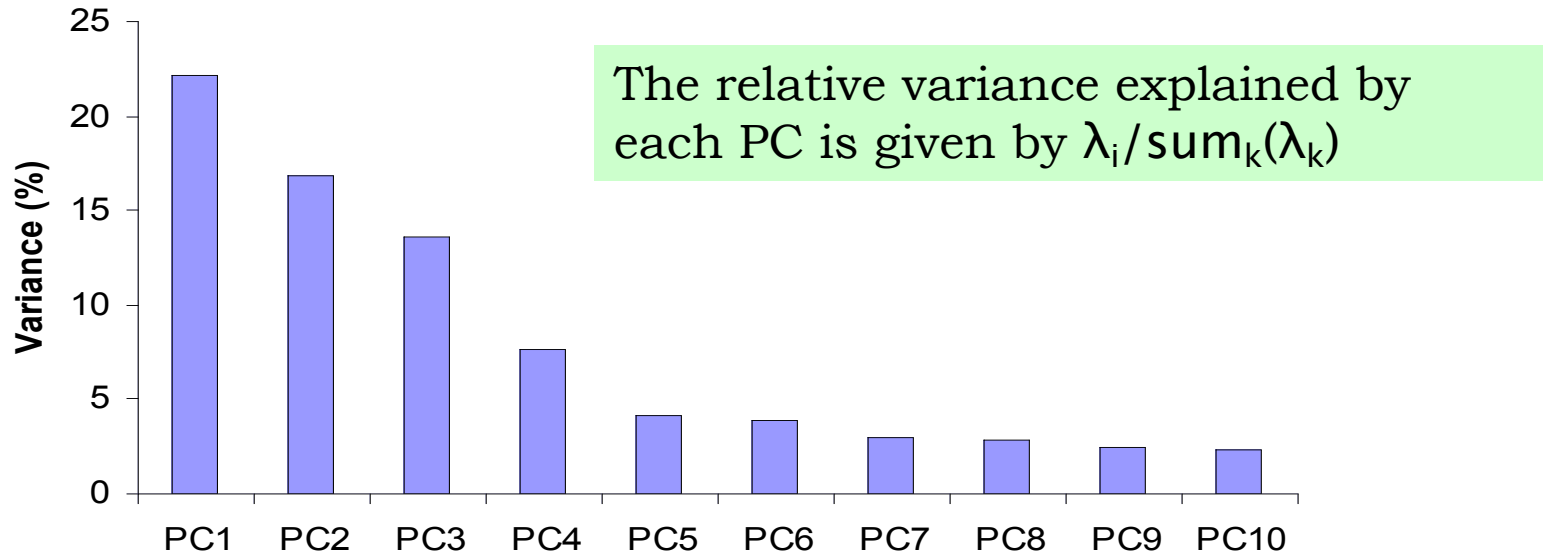
- **I. Variance:** Enough PCs to have a cumulative variance explained by the PCs that is >50-70%
- **II. Scree plot:** represents the ability of PCs to explain the variation in data, e.g. keep PCs with eigenvalues >1

e.g. check eigenvalue (I)



e.g. check percentage of kept variance

Can *ignore* the components of lesser significance.



You do *lose some information*, but if the eigenvalues are small, you don't lose much

- p dimensions in original data
- Calculate p eigenvectors and eigenvalues
- choose only the first k eigenvectors, by keep enough variance
- final projected data set has only k dimensions

Why to Reduce Dimension?

- PCA as a general dimensionality reduction technique
- Preserves most of variance with a much more compact representation
 - Lower storage requirements (eigenvectors + a few numbers (k) per sample)
 - Faster matching (since matching within a lower-dim)

(1) Limitations of PCA

- PCA is not effective for some datasets.
- For example, if the data is a set of strings
- $(1,0,0,0,\dots), (0,1,0,0,\dots), \dots, (0,0,0,\dots,1)$ then the eigenvalues do not fall off as PCA requires.

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

$$\text{eigenvalue} = [1, 1, 1]$$

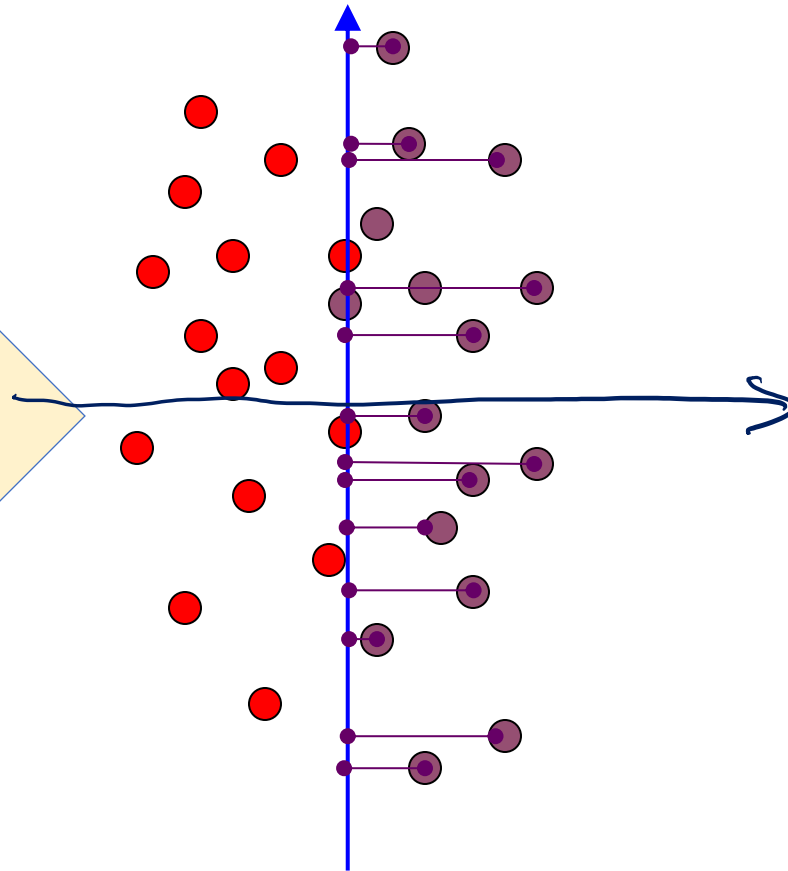
(2) PCA and Discrimination

- The direction of maximum variance is not always good for classification ([Example 1](#))

For this case:

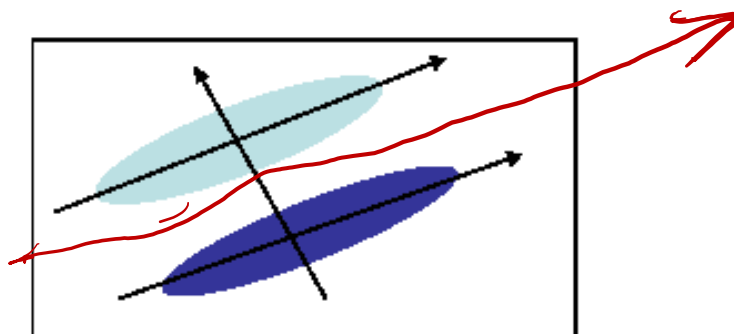
+ Ideal for capturing global variance !

+ Not ideal for discrimination

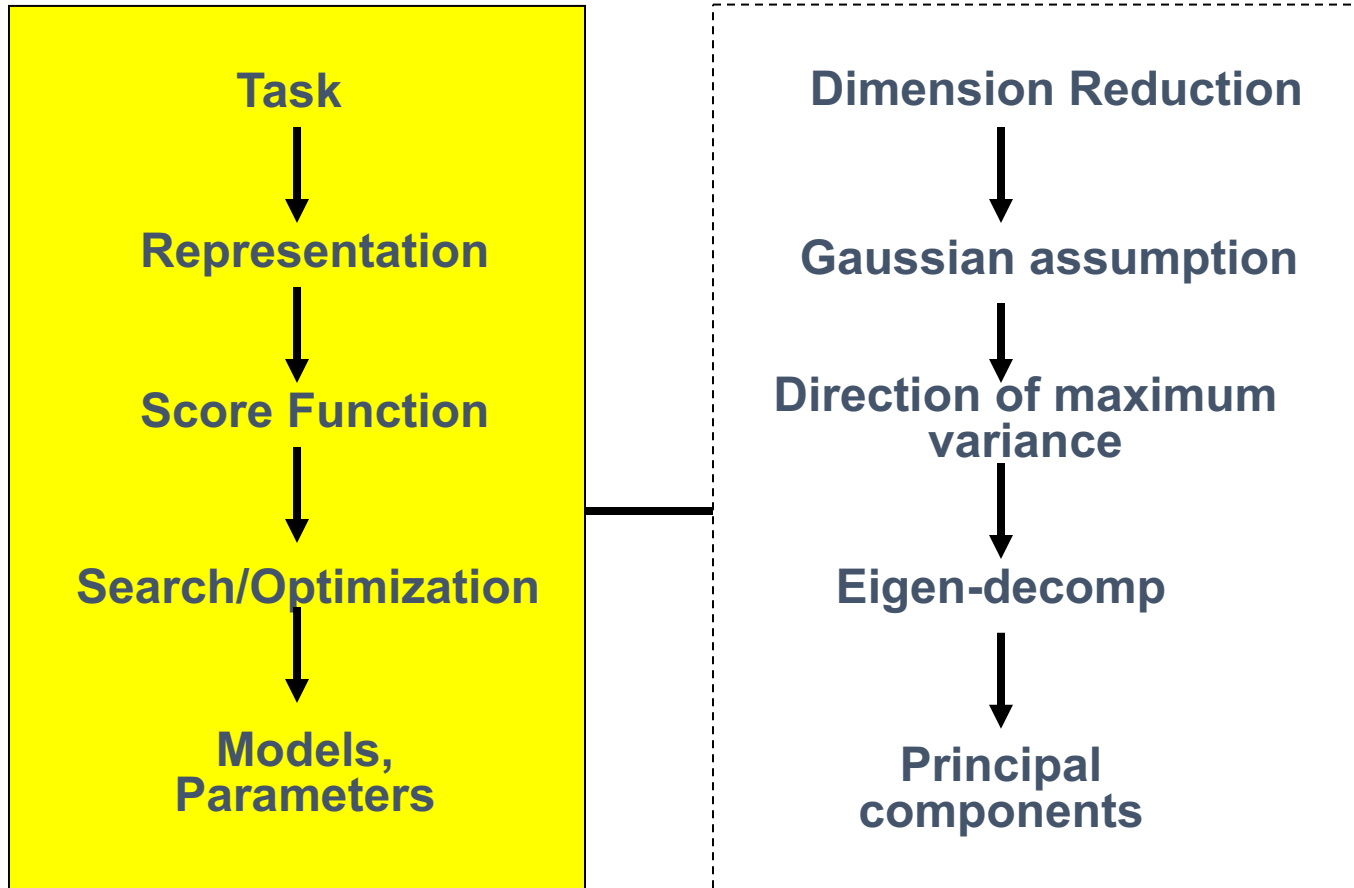


PCA and Discrimination

- PCA may not find the best directions for discriminating between two classes. (Example 2)
- Example:
 - suppose the two classes have 2D Gaussian densities as ellipsoids.
 - 1st eigenvector is best for representing the probabilities / overall data trend
 - 2nd eigenvector is best for discrimination.



Principal Component Analysis



References

- ❑ Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.
- ❑ Dr. S. Narasimhan's PCA lectures
- ❑ Prof. Derek Hoiem's eigenface lecture

Algebraic Review

- How many eigenvectors are there?
- For Real **Symmetric Matrices**
 - except in degenerate cases when eigenvalues repeat, there are p eigenvectors
 - u_1, \dots, u_p are the eigenvectors*
 - $\lambda_1 \dots \lambda_p$ are the eigenvalues, large to small, ordered by its value*
 - all eigenvectors are mutually orthogonal and therefore form a new basis space
 - Eigenvectors for distinct eigenvalues are mutually orthogonal
 - Eigenvectors corresponding to the same eigenvalue have the property that any linear combination is also an eigenvector with the same eigenvalue; one can then find as many orthogonal eigenvectors as the number of repeats of the eigenvalue.

Today

- Dimensionality Reduction (unsupervised) with Principal Components Analysis (PCA)
 - Review of eigenvalue, eigenvector
 - How to project samples into a line capturing the variation of the whole dataset → Eigenvector / Eigenvalue of covariance matrix
 - PCA for dimension reduction
 - Eigenface → PCA for face recognition

Example 1: Application to image, e.g. a task of face recognition

1. Treat pixels as a vector



2. Recognize face by 1-nearest neighbor



$\mathbf{y}_1 \dots \mathbf{y}_n$

A face-image database of totally n different people

$$k = \operatorname{argmin}_k \left\| \mathbf{y}_k^T - \mathbf{x} \right\|$$

Example 1: the space of all face images

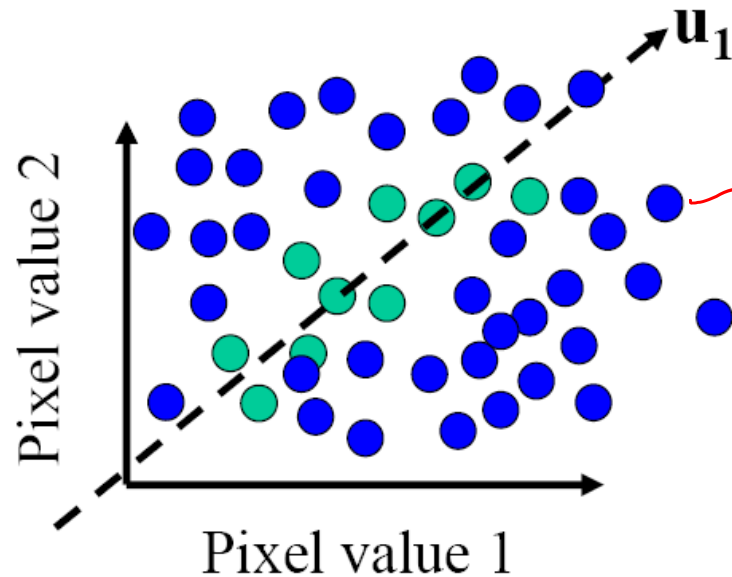
- When viewed as vectors of pixel values, face images are extremely high-dimensional
 - 100x100 image = 10,000 dimensions
 - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

$$p = 10,000$$



Example 1: The space of all face images

- Eigenface idea: construct a low-dimensional linear subspace that best explains the variation in the set of face images



We don't have blue samples in our face-image set

- A face image
- A (non-face) image

Example 1: Application to Faces, e.g. Eigenfaces (PCA on face images)

1. Compute covariance matrix of face images
2. Compute the principal components (“eigenfaces”)
 - K eigenvectors with largest eigenvalues
3. Represent all face images in the dataset as linear combinations of eigenfaces
 - Perform nearest neighbors on these projected low-d coefficients

Example 1: Application to Faces

Training
images



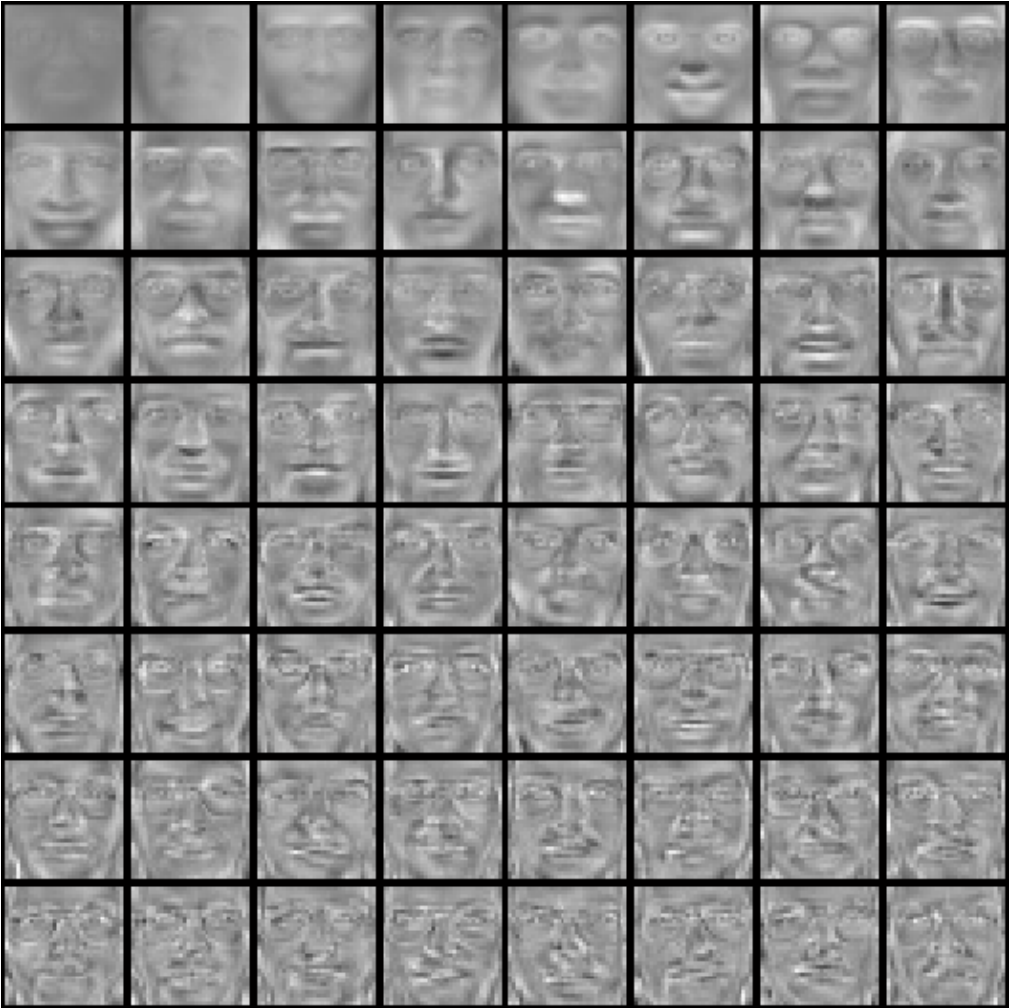
Example 1: Eigenfaces example

$$C = (X - \bar{X})^T (X - \bar{X})$$

Top eigenvectors: u_1, \dots, u_k

$k=64$

Mean: μ



$$\bar{X} = \mu = \frac{1}{N} \sum_{k=1}^N x_k$$

Example 1: Visualization of eigenfaces

$p = 10,000$

Principal component (eigenvector) u_k



u_1 u_2

$[k=9] u_9$

$\mu + 3\sigma_k u_k$

$\text{var}(X^T u_k) = \lambda_k$
 $\sigma_k = \sqrt{\lambda_k}$



$\mu - 3\sigma_k u_k$



Example 1: Representation and reconstruction of original \mathbf{x}

- Face \mathbf{x} in “face space” coordinates:



$$\mathbf{x} \rightarrow [\mathbf{u}_1^T (\mathbf{x} - \mu), \dots, \mathbf{u}_k^T (\mathbf{x} - \mu)]$$

$x^T v = v^T x$

$$g = [w_1, \dots, w_k]$$

New representation

Remarkably few eigenvector terms are needed to give a fair likeness of most people's faces.

→ subtract the mean along each dimension, in order to center the original axis system at the centroid of all data points

Representation and reconstruction

- Face \mathbf{x} in “face space” coordinates:



$$\mathbf{x} \rightarrow [\mathbf{u}_1^T (\mathbf{x} - \mu), \dots, \mathbf{u}_k^T (\mathbf{x} - \mu)]$$

$$= w_1, \dots, w_k$$

← New representation

- Reconstruction:

$$\Rightarrow \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \quad \text{reconstruction error}$$



=



+



$$\hat{\mathbf{x}}$$

=

$$\mu$$

$$+ w_1 u_1 + w_2 u_2 + w_3 u_3 + w_4 u_4 + \dots + w_k u_k$$

A human face may be considered to be a linear combination of these **standardized eigen faces**

Assuming centered data

⑥

$\vec{u}_1, \vec{u}_2, \dots, \vec{u}_k$ Top k Eigen Vectors,

⑦

each \vec{u}_i is $p \times 1$ column vector

⑧

$\vec{g}_i = \begin{bmatrix} \vec{x}_i^T \vec{u}_1 & \vec{x}_i^T \vec{u}_2 & \dots & \vec{x}_i^T \vec{u}_k \end{bmatrix}^T$ $p \times 1$ column vector

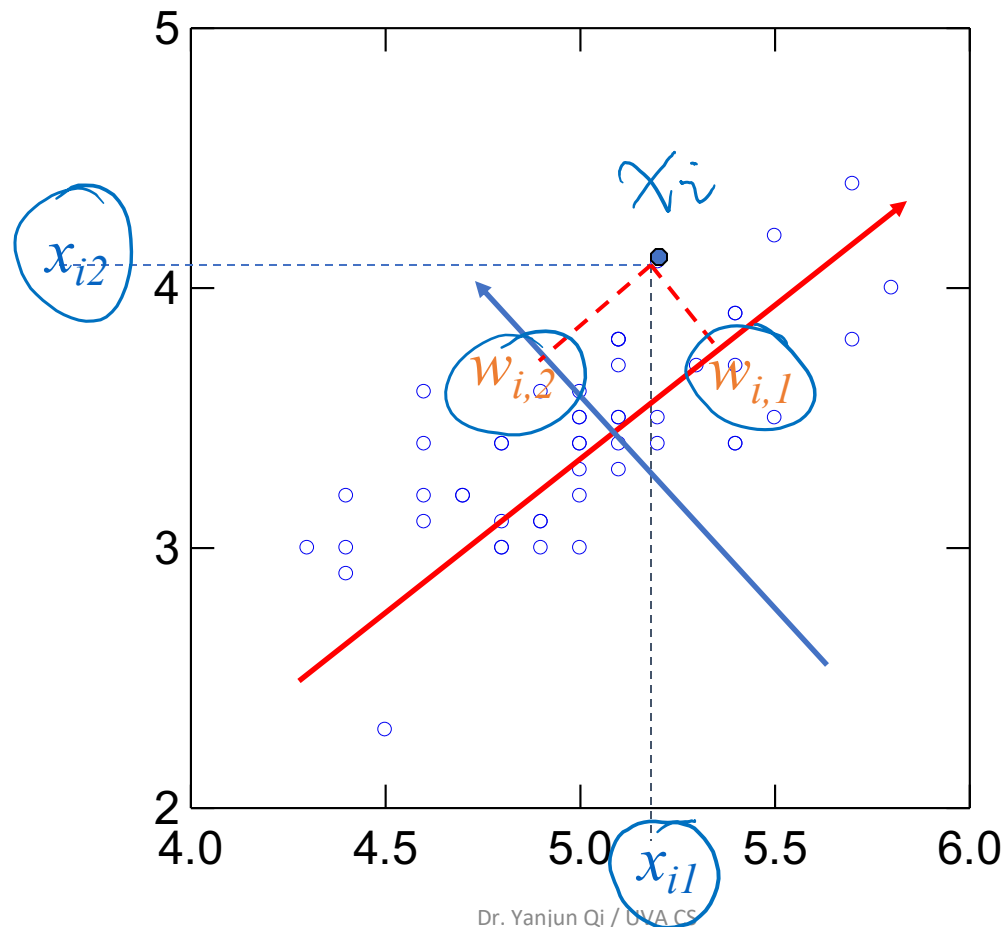
$$\vec{x}_i^T = \vec{g}_i^T \begin{bmatrix} \vec{u}_1^T \\ \vec{u}_2^T \\ \vdots \\ \vec{u}_k^T \end{bmatrix} \quad k \times p$$

$$= \begin{bmatrix} \vec{x}_i^T \vec{u}_1 & \vec{x}_i^T \vec{u}_2 & \dots & \vec{x}_i^T \vec{u}_k \end{bmatrix}$$

$$\begin{bmatrix} \vec{u}_1^T \\ \vec{u}_2^T \\ \vdots \\ \vec{u}_k^T \end{bmatrix}$$

$$= \vec{x}_i^T \sum_{i=1}^k \vec{u}_i \vec{u}_i^T$$

New representation in the lower-dim PC space



original
 $x_i \rightarrow [x_{i1}, x_{i2}]$
 \Downarrow
 $g_i \rightarrow [w_{i1}, w_{i2}]$
projected
 $\left[x_i^T u_1, x_i^T u_2 \right]$

Key Property of Eigenspace Representation

Given

- 2 images \hat{x}_1, \hat{x}_2 that are used to construct the Eigenspace
- \hat{g}_1 is the eigenspace projection of image \hat{x}_1
- \hat{g}_2 is the eigenspace projection of image \hat{x}_2

Then,

$$\| \hat{g}_2 - \hat{g}_1 \| \approx \| \hat{x}_2 - \hat{x}_1 \|$$

That is, distance in Eigenspace is approximately equal to the distance between two original images.

Classify / Recognition with eigenfaces

Step 1: Process labeled training images

- Find mean $\boldsymbol{\mu}$ and covariance matrix
- Find k principal components (i.e. eigenvectors of $\boldsymbol{\Sigma}$) $\rightarrow \mathbf{u}_1, \dots, \mathbf{u}_k$
- Project each training image \mathbf{x}_i onto subspace spanned by the **top** principal components:
 $(w_{i1}, \dots, w_{ik}) = (\mathbf{u}_1^T(\mathbf{x}_i - \boldsymbol{\mu}), \dots, \mathbf{u}_k^T(\mathbf{x}_i - \boldsymbol{\mu}))$

Classify / Recognition with eigenfaces

Step 2: Nearest neighbor based face classification

Given a novel image \mathbf{x}

- Project onto k PC's subspace:
 $(w_1, \dots, w_k) = (\mathbf{u}_1^T(\mathbf{x} - \boldsymbol{\mu}), \dots, \mathbf{u}_k^T(\mathbf{x} - \boldsymbol{\mu}))$
- **Optional**: check reconstruction error $\mathbf{x} - \hat{\mathbf{x}}$ to determine whether the image is really a face
- Classify as closest training face(s) in the lower k -dimensional subspace

Is this a face or not?

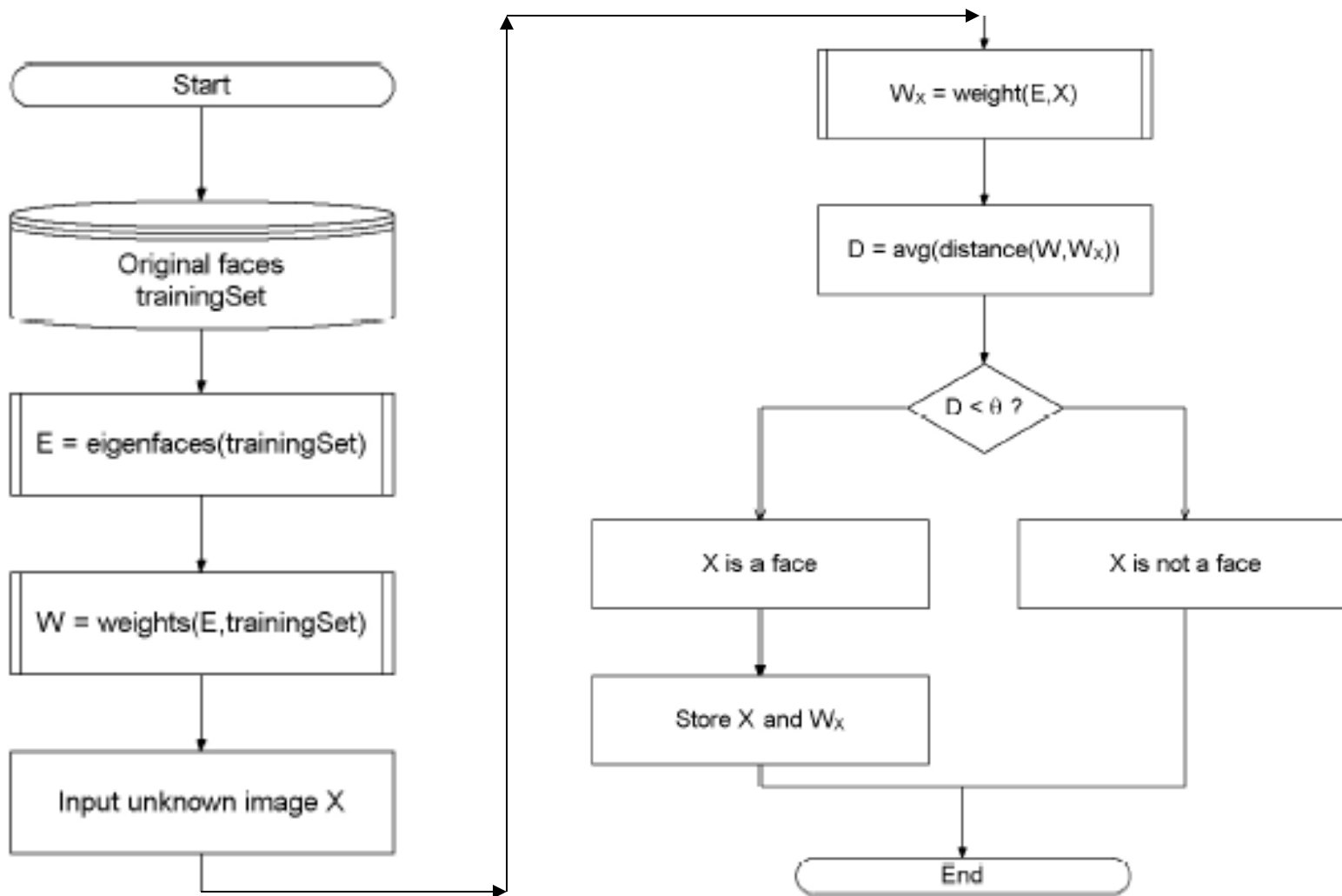


Figure 1: High-level functioning principle of the eigenface-based facial recognition algorithm

Example 2: e.g. Handwritten Digits

- 16 x 16 gray scale
- Total 658 such 3's
- 130 is shown
- Image $x_i : \mathbb{R}^{256}$
- Compute principal components

$p=256$

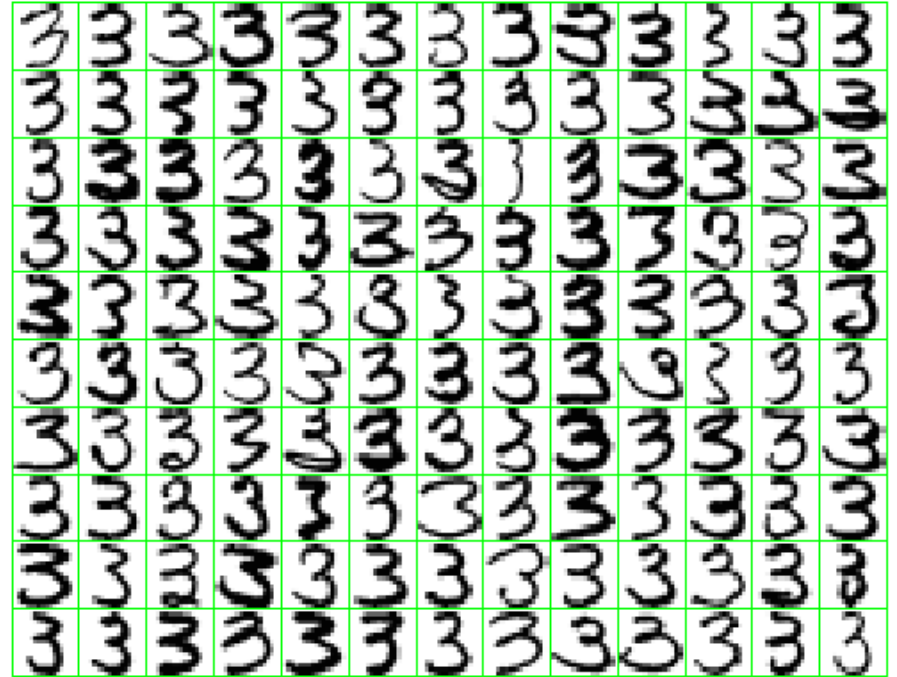


FIGURE 14.22. A sample of 130 handwritten 3's shows a variety of writing styles.

$$\hat{X} = \mu$$

u_1

u_2

$k=2$

$$x = \mu + w_1 u_1 + w_2 u_2$$

$$\mathbf{x} \rightarrow [\mathbf{u}_1^T (\mathbf{x} - \mu), \dots, \mathbf{u}_k^T (\mathbf{x} - \mu)]$$

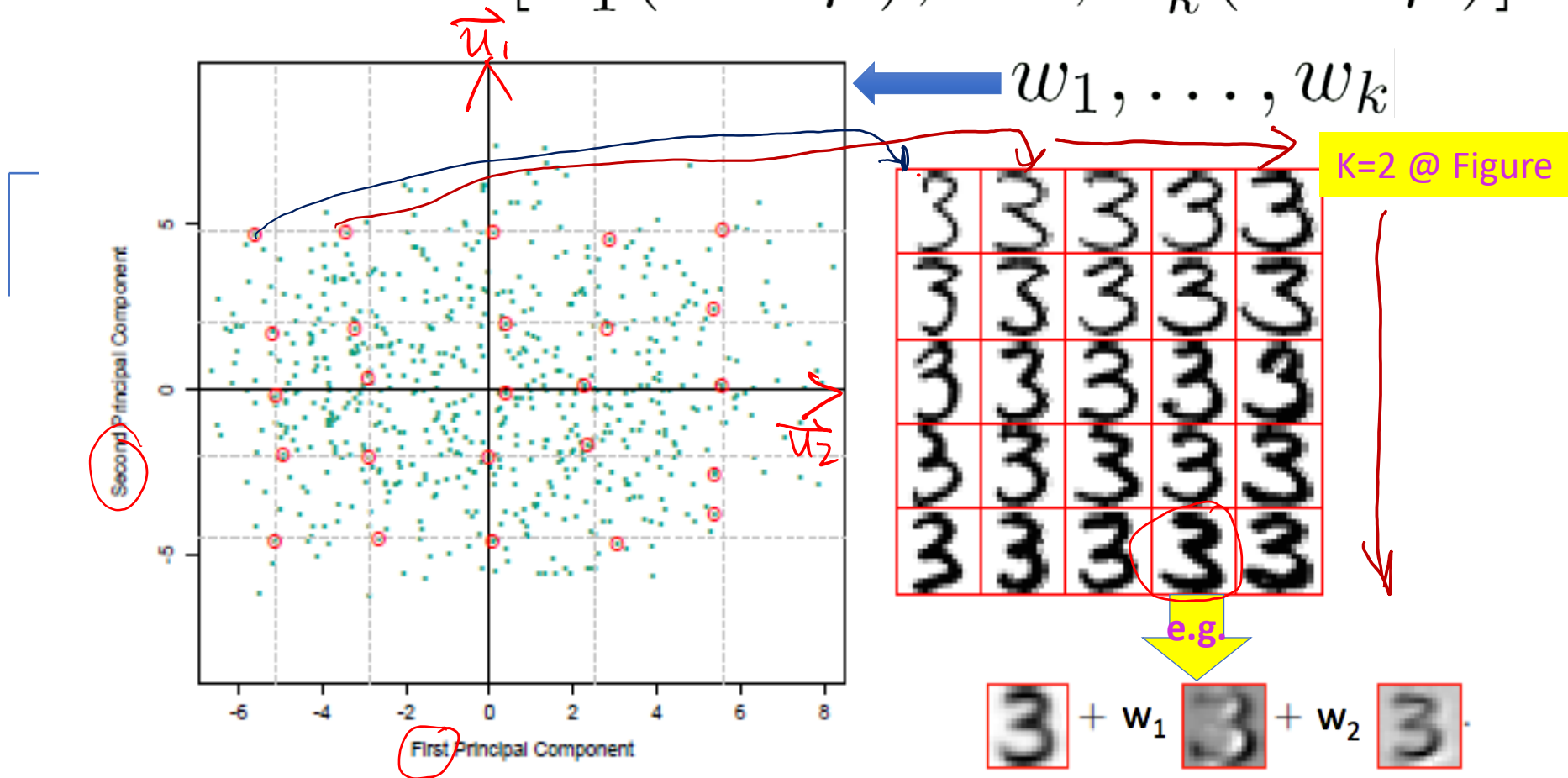
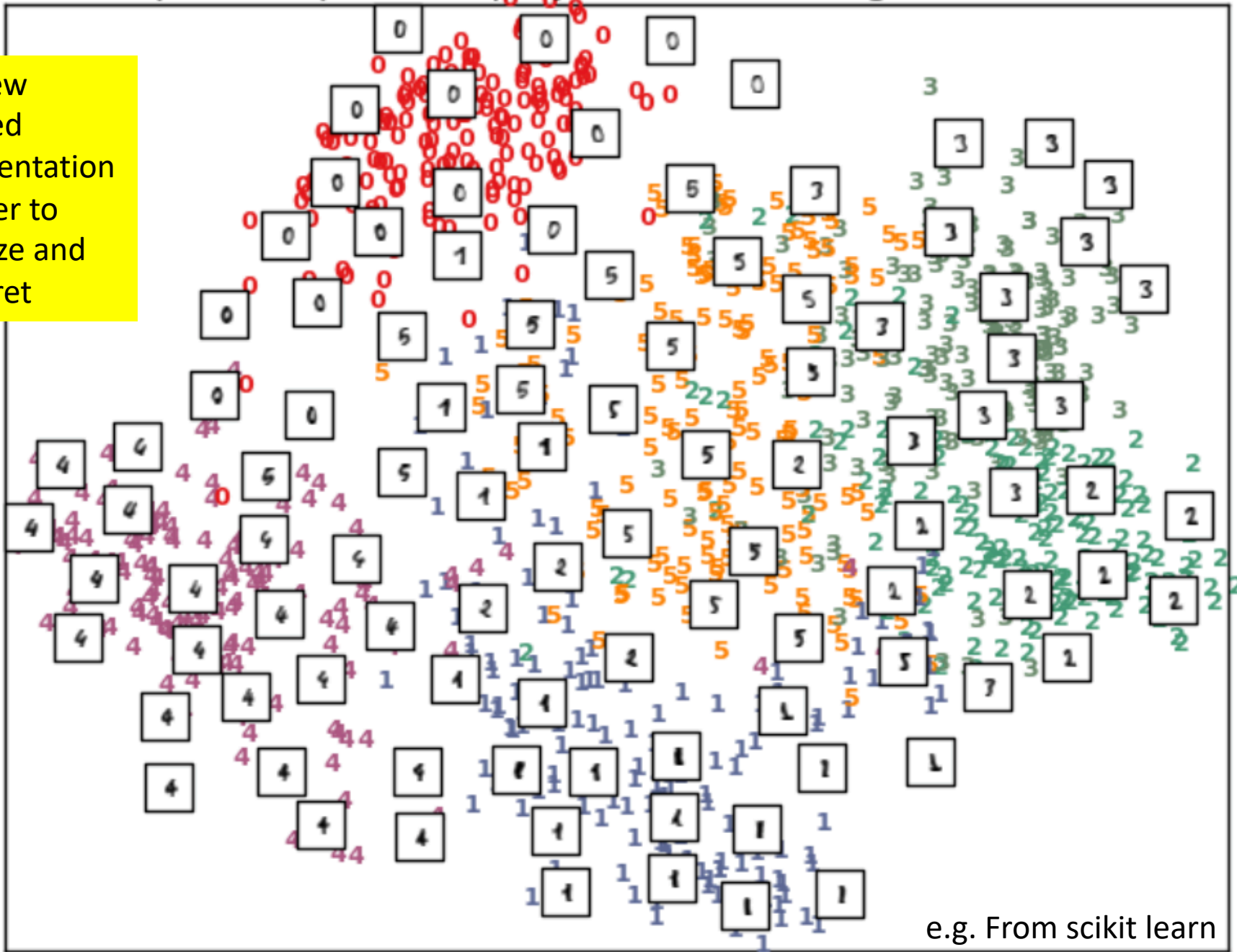


FIGURE 14.23. (Left panel:) the first two principal components of the handwritten threes. The circled points are the closest projected images to the vertices of a grid, defined by the marginal quantiles of the principal components. (Right panel:) The images corresponding to the circled points. These show the nature of the first two principal components.

Principal Components projection of the digits (time 0.02s)

The new reduced representation is easier to visualize and interpret



e.g. From scikit learn



Extra: A 2D Numerical Example

PCA Example –STEP 1

- **Subtract the mean** from each of the data dimensions.
- Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and co-variance values are not affected by the mean value.

PCA Example –STEP 1

DATA: (p=2)

<u>x1</u>	<u>x2</u>
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

ZERO MEAN DATA:

<u>x1</u>	<u>x2</u>
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

PCA Example –STEP 2

- Calculate the covariance matrix

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

- since the non-diagonal elements in this covariance matrix are positive, we should expect that the x1 and x2 variable increase together.

PCA Example –STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

$$\text{eigenvalues} = \begin{pmatrix} 1.28402771 \\ .0490833989 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

PCA Example –STEP 3

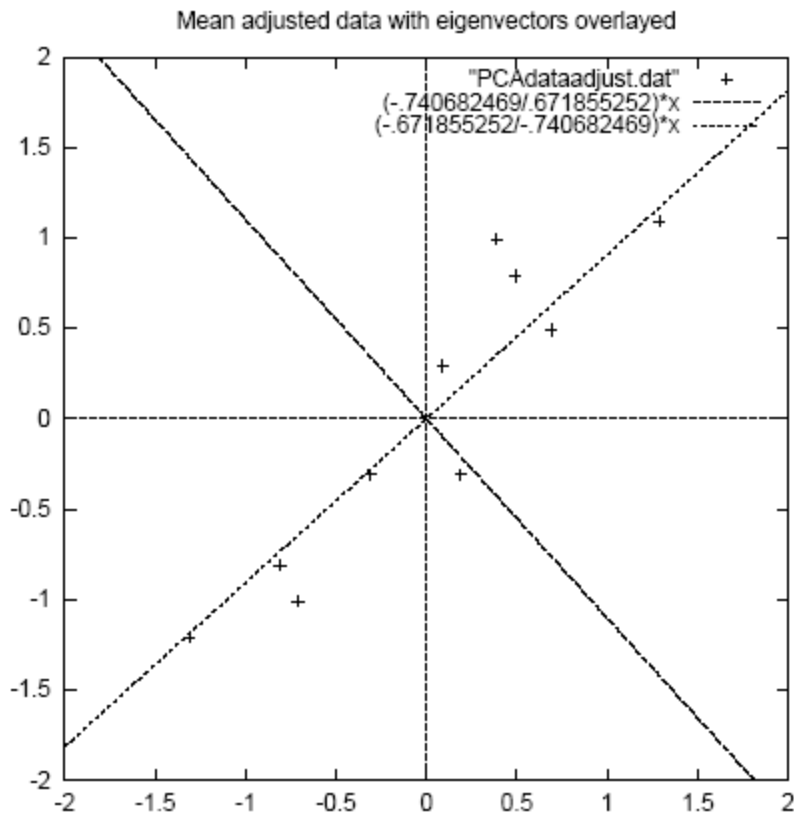


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlaid on top.

- eigenvectors are plotted as diagonal dotted lines on the plot.

- Note they are perpendicular to each other.

- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.

- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

PCA Example –STEP 4

- Reduce dimensionality and form *feature vector*

the eigenvector with the *highest* eigenvalue is the *principle component* of the data set.

In our example, the eigenvector with the largest eigenvalue was the one that pointed down the middle of the data.

Once eigenvectors are found from the covariance matrix, the next step is to **order them by eigenvalue**, highest to lowest. This gives you the components in order of significance.

PCA Example –STEP 4

- Feature Vector

$$\text{FeatureVector} = (\text{eig}_1 \text{ eig}_2 \text{ eig}_3 \dots \text{eig}_n)$$

We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

Now, if you like, you can decide to *ignore the components of lesser significance*.

You do *lose some information*, but if the eigenvalues are small, you don't lose much

PCA Example –STEP 5

- Deriving the new data

FinalData = RowFeatureVector x RowZeroMeanData

RowFeatureVector is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top

RowZeroMeanData is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension.

PCA Example –STEP 5

FinalData transpose: dimensions
along columns

w1	w2
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

PCA Example –STEP 5

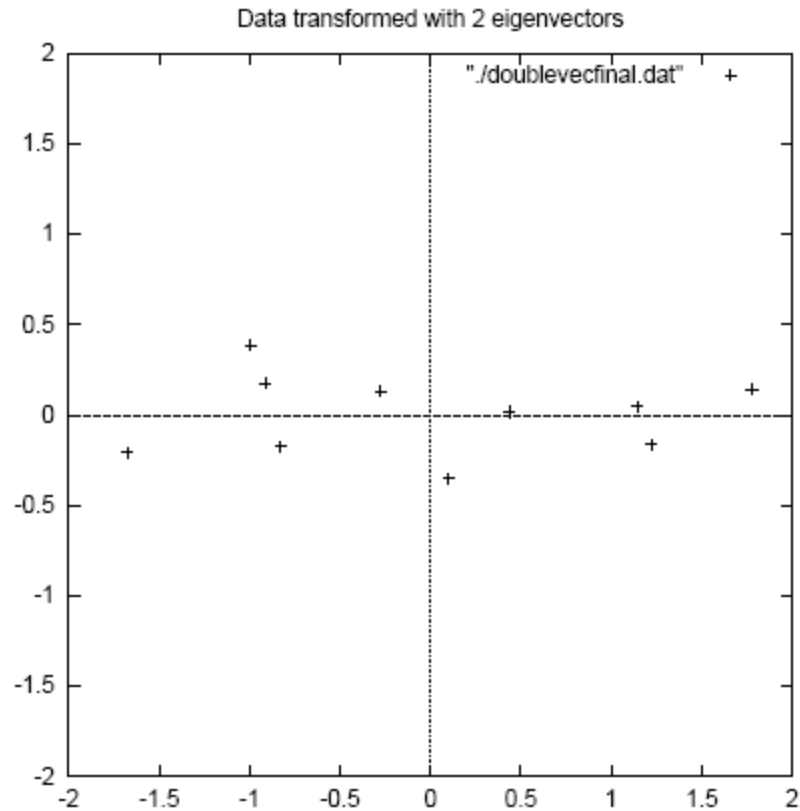


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

Reconstruction of original Data

- If we reduced the dimensionality, obviously, when reconstructing the data we would lose those dimensions we chose to discard.
- In our example let us assume that we considered only the w_1 dimension...

Reconstruction of original Data

w1

-0.827970186
1.77758033
-0.992197494
-0.274210416
-1.67580142
-0.912949103
.0991094375
1.14457216
.438046137
1.22382056

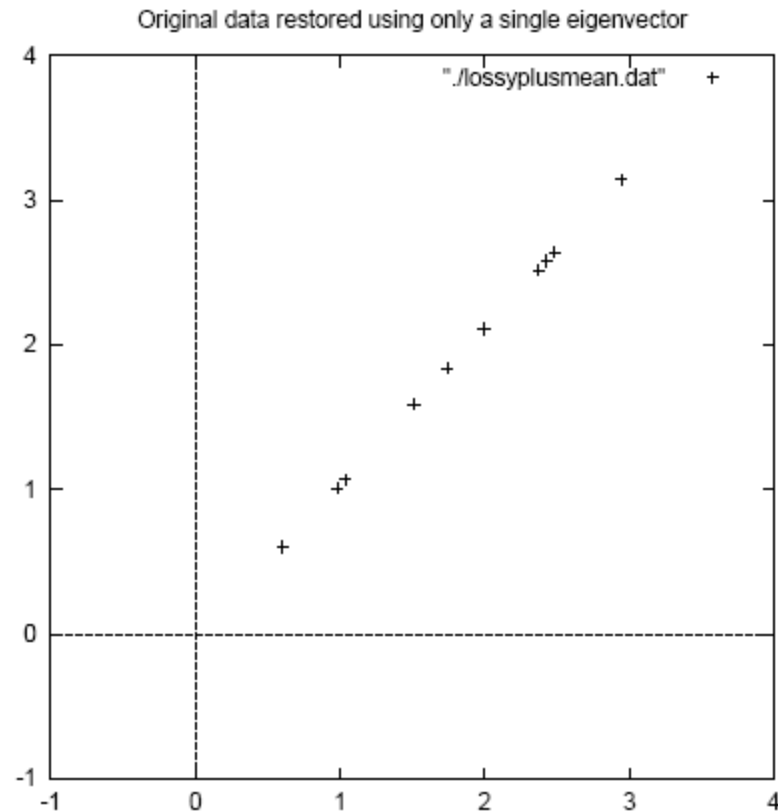


Figure 3.5: The reconstruction from the data that was derived using only a single eigenvector