SDSU CS 549 Spring 2024 Machine Learning Lecture 6: PCA

IRFAN KHAN

New Student Assistant/Grader

Student Assistant/Grader: Manas Gandhi

Email: mgandhi4512@sdsu.edu

Office location: GMCS 549

Office Hours: TBA

References

SDSU CS549 Lecture Notes by Prof Yang Xu, Spring 2023. Some updated slides used here

Coursera machine learning course by Dr Andrew Ng, Oct 2023

Outline

What is principal component analysis (PCA)? Why do we need it?

Intuitive picture of PCs

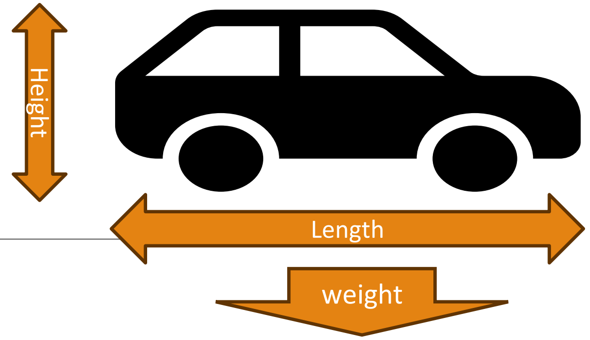
Algebraic definition and derivation of PCs

Applications of PCA

What is Principal Component Analysis (PCA)?

- Principal Component Analysis (PCA) is an unsupervised machine learning technique.
- In PCA, the goal is to transform the original features of a dataset into a new set of uncorrelated features called principal components.
- The principal components are ordered by the amount of variance they capture in the data. PCA is commonly used for dimensionality reduction, **visualization**, and noise reduction in datasets.
- Unlike supervised learning, PCA doesn't rely on labeled output for training.
- It analyzes the inherent structure of the data to find patterns and reduce the dimensionality.
- However, PCA can be combined with supervised learning techniques in some applications, such as feature extraction or preprocessing before applying a supervised algorithm

Example (Car Size and Weight)



	Weight (1000s of Lbs)	Length (feet)	Height (feet)	……
Car 1	2.0	12	5	
Car 2	6.0	16	6	
Car 3	4.0	14	4.5	
Car 4	5.0	15	5.5	
Car 5	4.5	14	5	
Car 6	3.5	13	4.5	
….				

- Questions:
  - How to visualize?
  - Which features are correlated?
  - Which features are most significant to describe the data?

Which features are most significant to describe the data?

- Number of Features ⇔ Columns of the Sample Data Matrix *(m x n)*
  - m is the number of samples

"**Significant**" feature/"dimension of data" should:

- Have higher resolution/variability → data points *spread out* across the dimension as much as possible
- Be independent → avoid redundancy

PCA Machine Learning

- PCA transforms sample data into a new coordinate system in which:
  - The dimensions are *orthogonal* (guarantee ***independence***) and are
  - *ranked* according to the variance of data along them (so that more ***informative*** dimensions, along which the data spread out more, occur first)

Outline

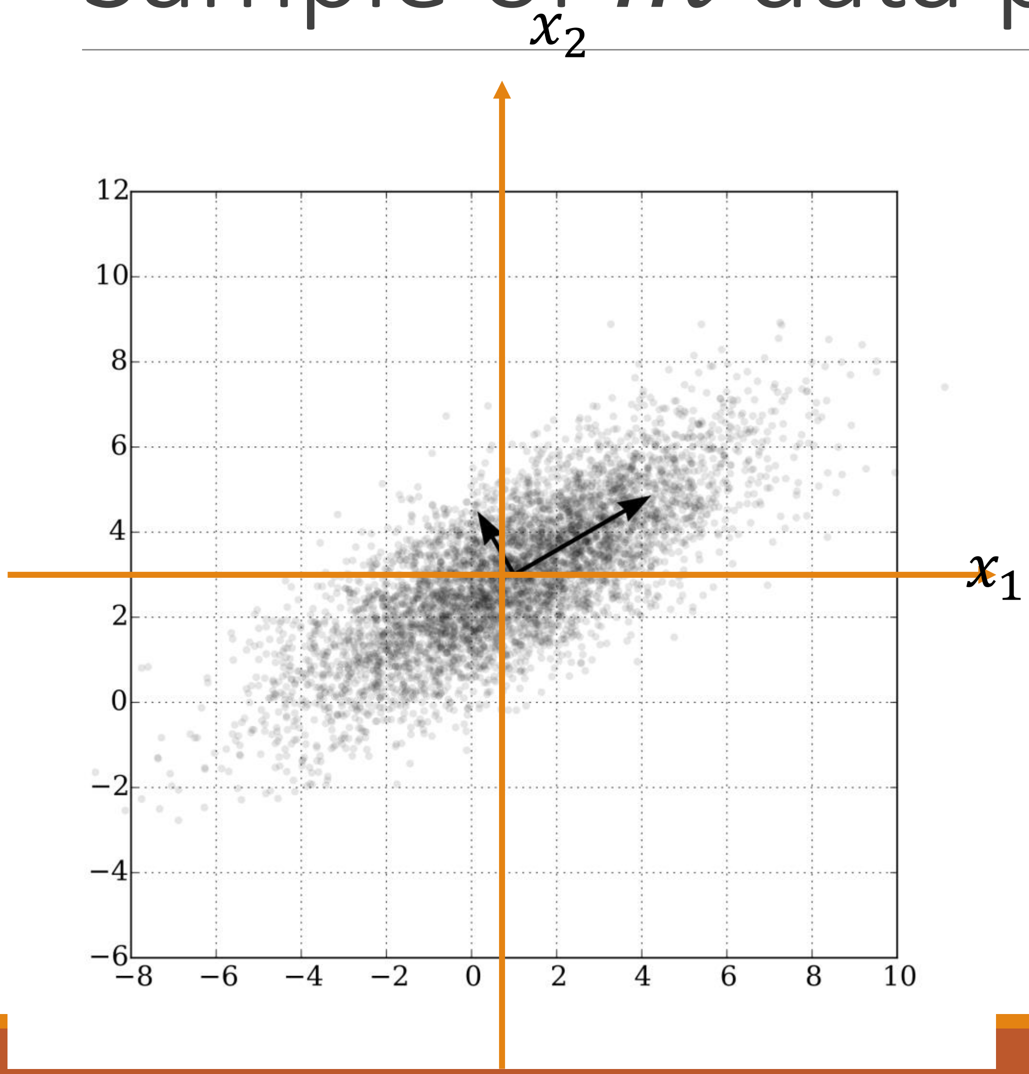
What is principal component analysis (PCA)? Why do we need it?

Intuitive picture of PCs

Algebraic definition and derivation of PCs

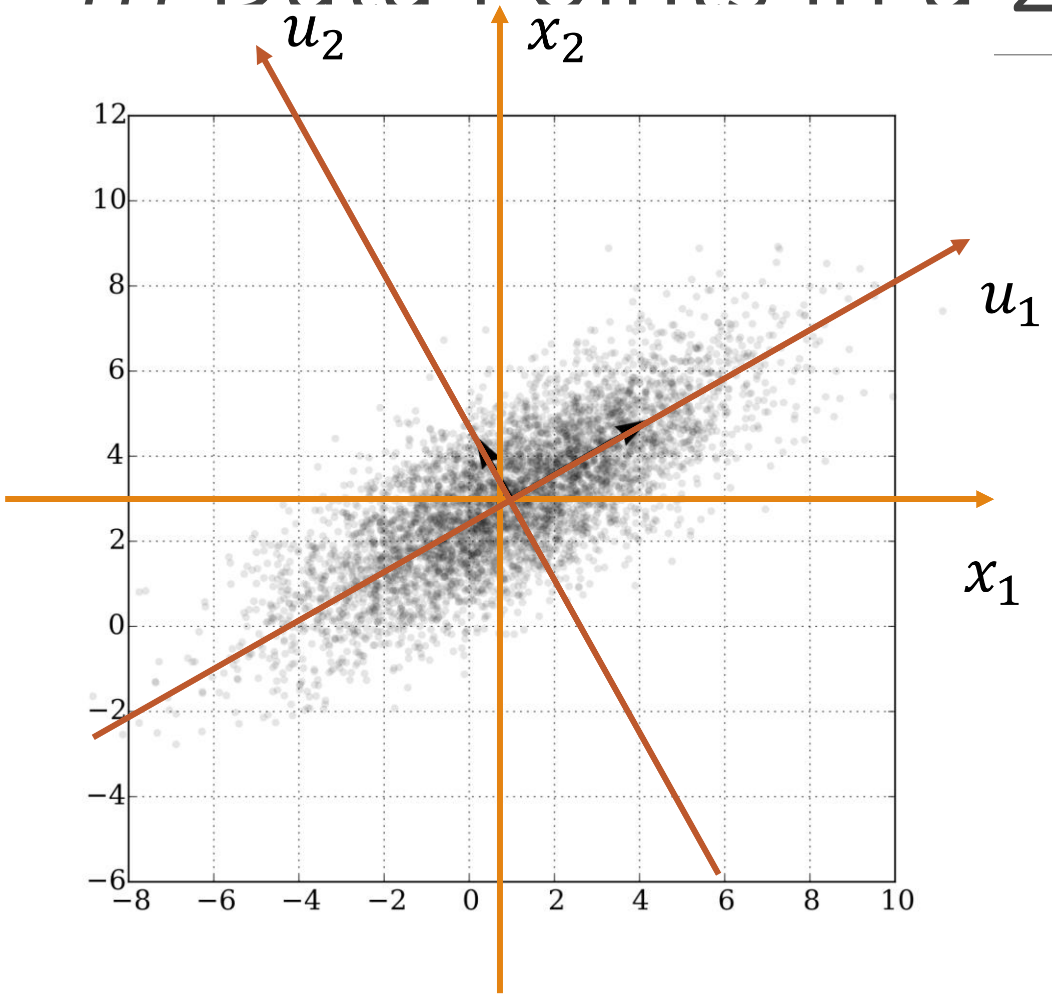
Applications of PCA

Sample of m data points in 2-D space



- 2-D Space => 2 features
- **Goal**: to account for the variation in data points with as few dimensions as possible.
- If we are to use only one dimension to describe the data, which one do we choose?
  - x_1 or x_2?
  - Can we do better - How about other dimensions?

m Data Points in a 2D-Space



u_1 and u_2 seem to better represent the data than x_1 and x_2, as they describe the main "**directions**" of data

Continuing with Our Car Example Using Only Two Features

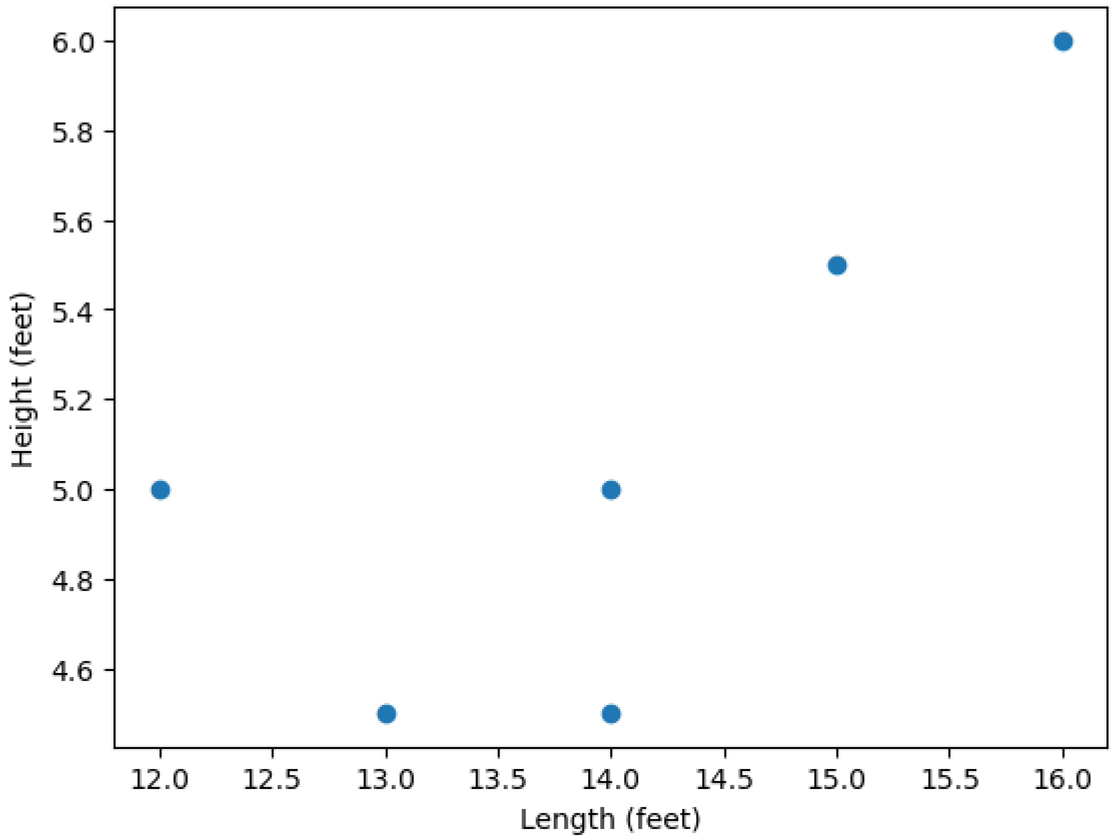
	Length (feet) **x1**	Height (feet) **x2**
Car 1	12	5
Car 2	16	6
Car 3	14	4.5
Car 4	15	5.5
Car 5	14	5
Car 6	13	4.5
….		

Mean x_1: $\mu_1 = \frac{1}{\mathrm{m}}\sum_{i=1}^{m} x_{i1} = 14.0$

Std Dev x_1:$\sigma_1 = \frac{1}{\mathrm{m}-1}\sum_{i=1}^{m}(x_{i1} - \mu_i)^2 = 1.41$

Mean x_2: $\mu_2 = 5.08$

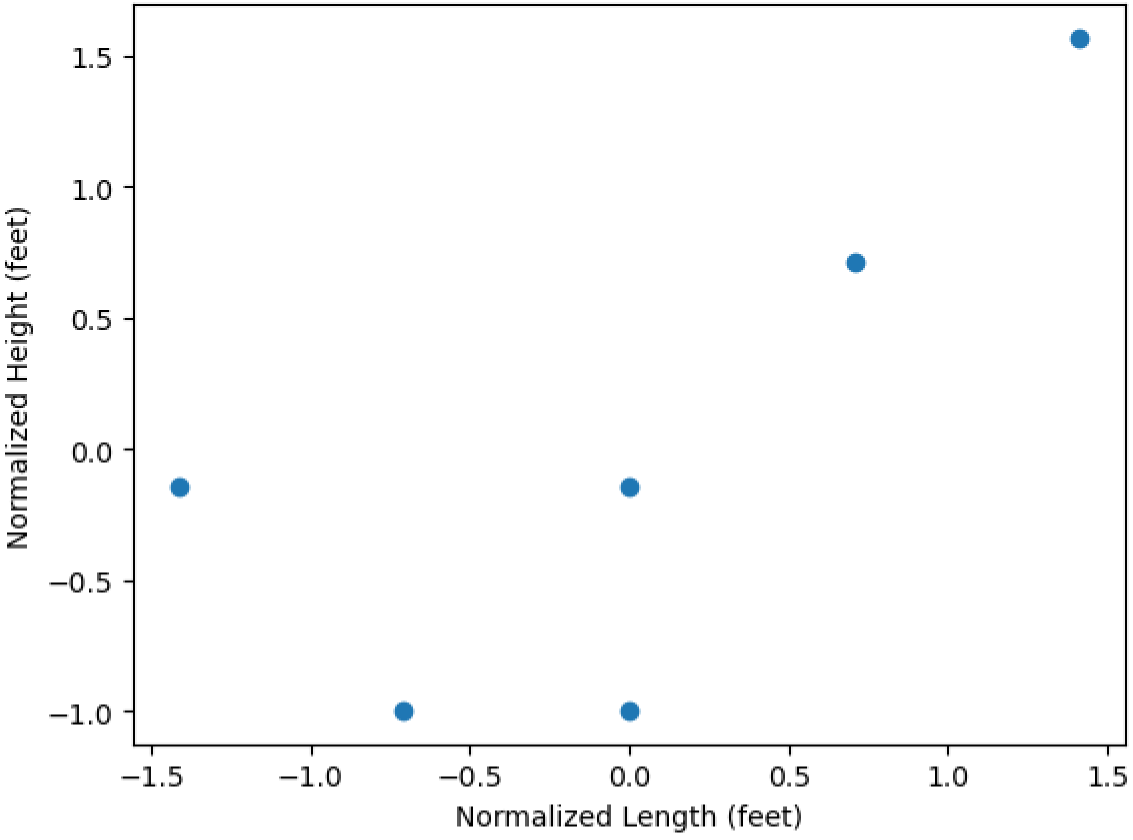
Std Dev x_2:$\sigma_2 = 0.58$



Normalizing the Data

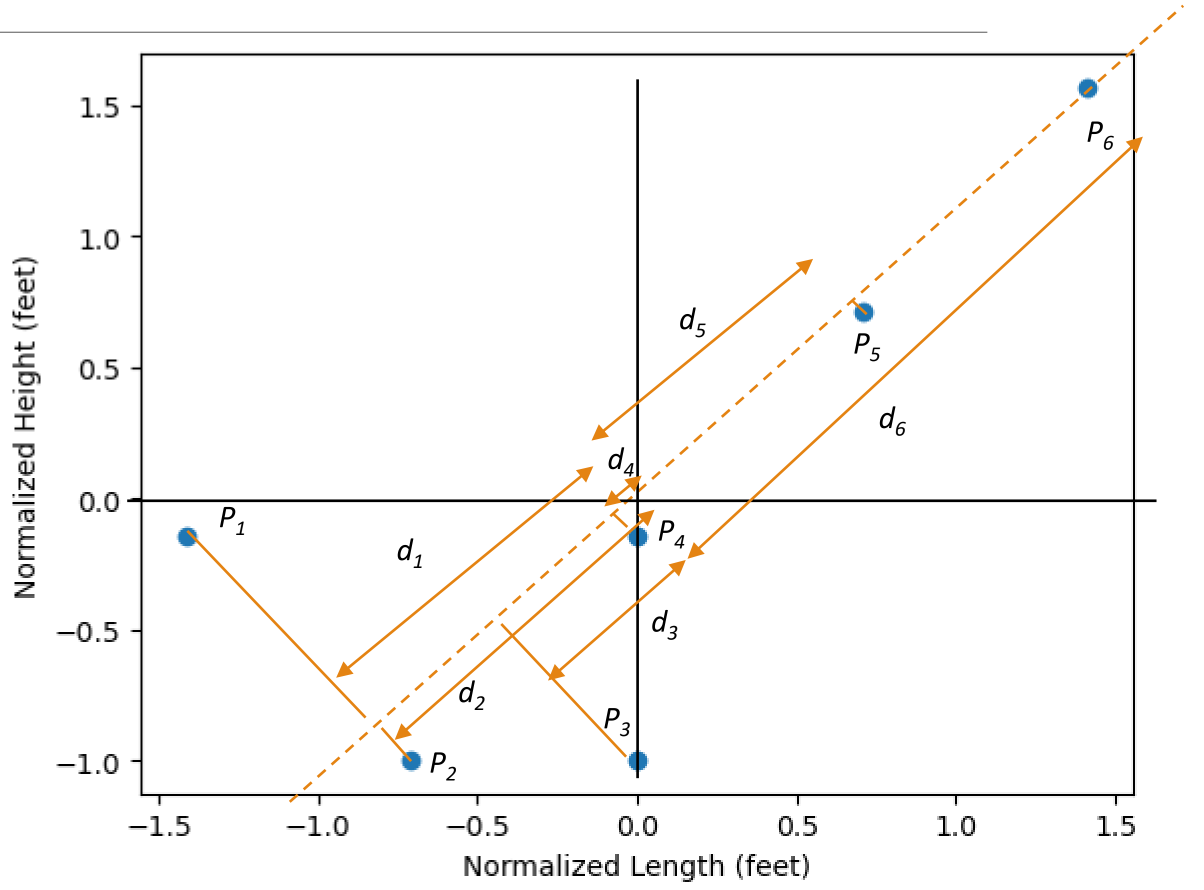
$$\widetilde{x_1} = \frac{x_1 - \mu_1}{\sigma_1} \qquad \widetilde{x_2} = \frac{x_1 - \mu_2}{\sigma_2}$$

	Normalized Length (feet) ($\widetilde{x1}$)	Normalized Height (feet) ($\widetilde{x2}$)
Car 1	-1.41	-0.14
Car 2	1.41	1.57
Car 3	0	-1.0
Car 4	0.71	0.71
Car 5	0	-0.14
Car 6	-0.71	-1.0

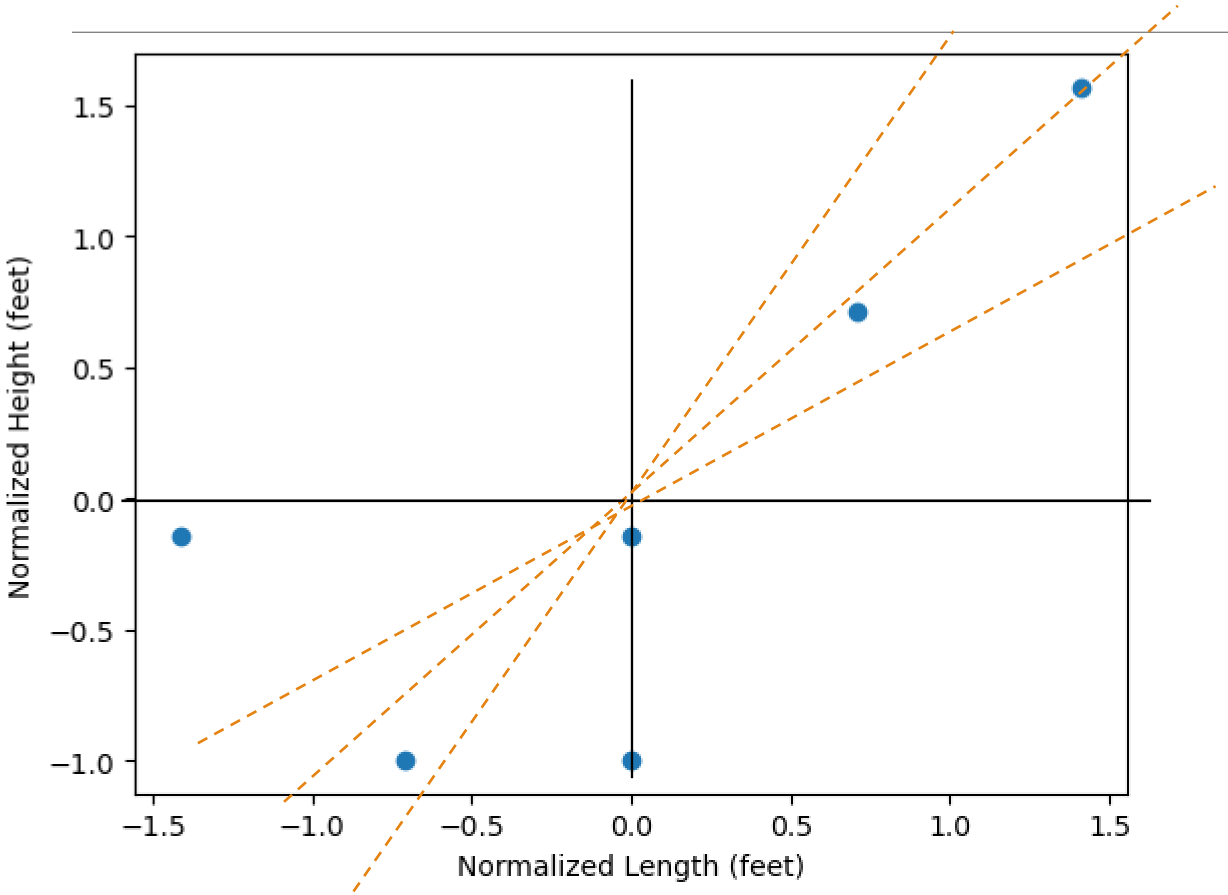


Quantifying Variance Along an Axis (Dimension)

- Consider the new axis drawn on the scatter plot
- For point P_1, d_1 is the projection on the new axis
- Similarly for other points
- Total variance along this dimension represented by the sum of squares of all the projections:
- SS = $\sum_{i=1}^{m} d_i^2$
- Our objective function to maximize is SS



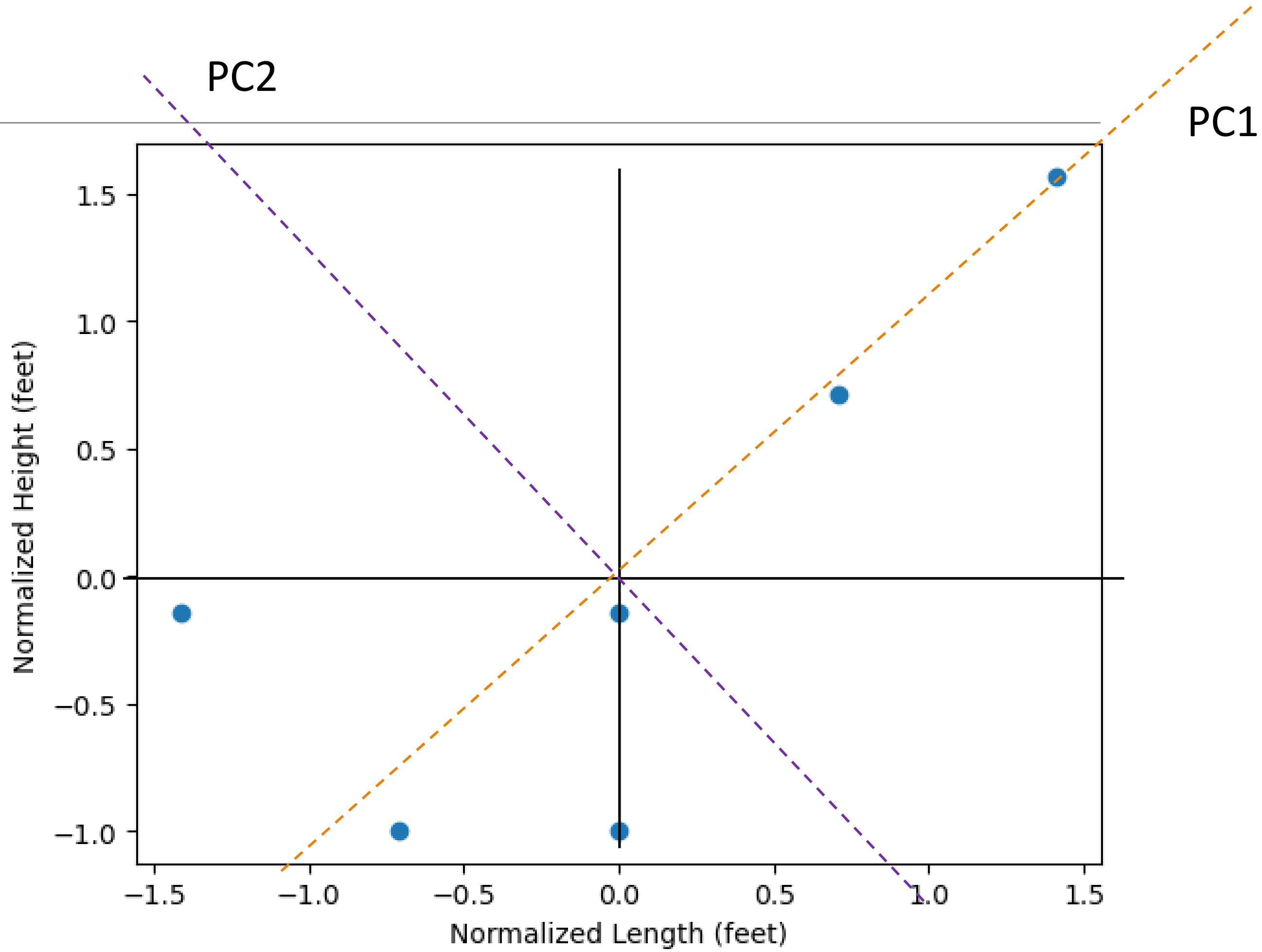
Maximize the Sum of Square of Projections



- Which Axis should we choose?
  - One that maximizes *SS: PC1*
- Let a unit vector, $\vec{u}_1$, represent PC1
- In our example this turns out to be $\begin{bmatrix} -0.71 \\ 0.71 \end{bmatrix}$
- For each normalized sample data point, the transformed data point, say z_1, along PC1 is given by the dot product of $\vec{u}_1$ with $\tilde{\boldsymbol{x}} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}$
- $z_1 = -.71 * \tilde{x}_1 + 0.71 * \tilde{x}_2$

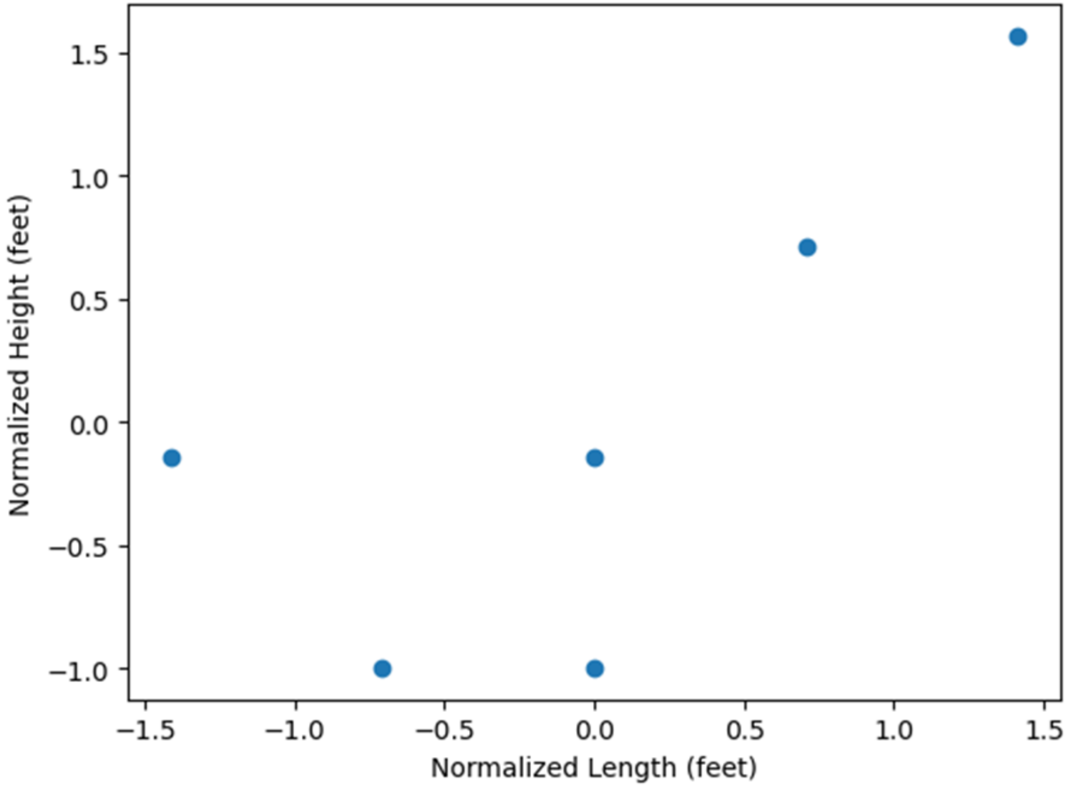
PC1 and PC2

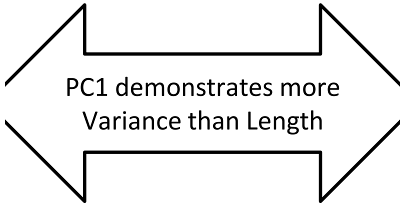
- The second principal component (PC2) is **perpendicular** to PC1
- Let a unit vector, $\vec{u}_2$, represent PC2
- In our example this turns out to be $\begin{bmatrix}-0.71\\-0.71\end{bmatrix}$
- Note that since $\vec{u}_1$ and $\vec{u}_2$ are perpendicular, their dot product must be *0*.
- For each normalized sample data point, the transformed data point, say z_2, along PC2 is given by the dot product of $\vec{u}_2$ with $\tilde{x} = \begin{bmatrix}\tilde{x}_1\\\tilde{x}_2\end{bmatrix}$
  - $z_2 = -.71 * \tilde{x}_1 - 0.71 * \tilde{x}_2$
- Projections on PC1 have a larger variance than on PC2
  - So $SS_1 > SS_2$



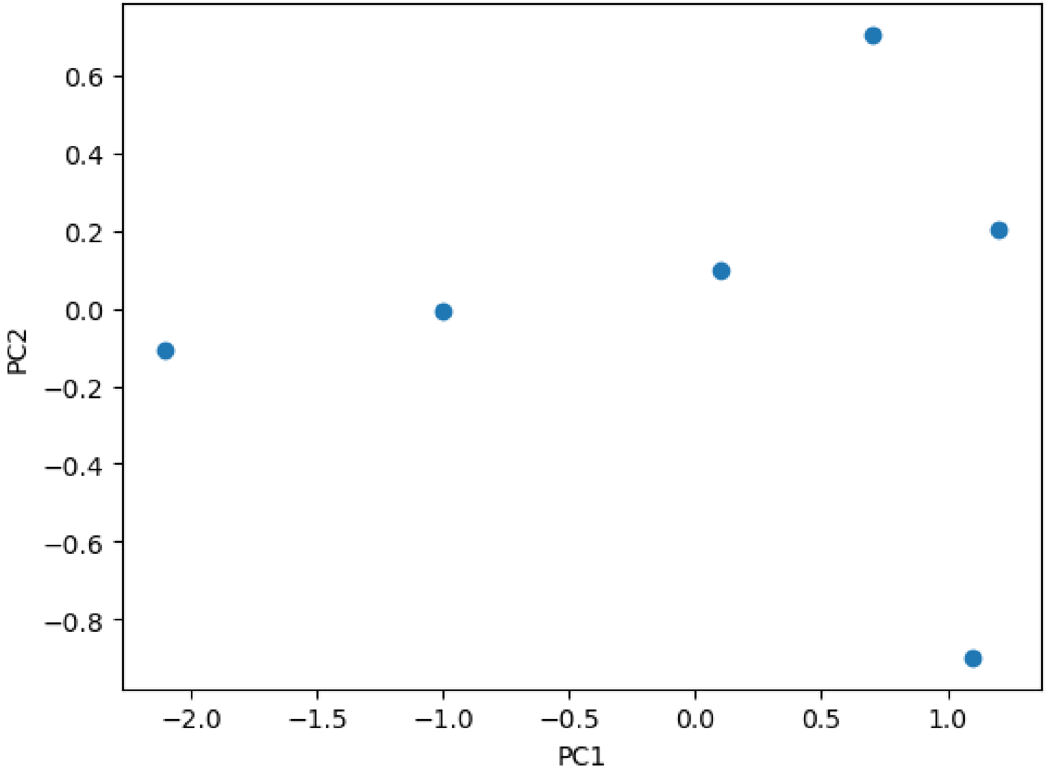
Rotate onto the new Axes (basis)

Original Normalized Data





Represent data using principal components



Outline

What is principal component analysis (PCA)? Why do we need it?

Intuitive picture of PCs

Algebraic definition and derivation of PCs

Applications of PCA

How to compute principal components?

Represent the **principal component (PC)** via a unit length vector $\vec{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$

The <u>length of projection</u> of $\tilde{x}^{(i)} = \begin{pmatrix} \tilde{x}_1^{(i)} \\ \tilde{x}_2^{(i)} \end{pmatrix}$ along $\vec{u}$ is the *inner product*: $\vec{u}^T \tilde{x}^{(i)} = u_1 \tilde{x}_1^{(i)} + u_2 \tilde{x}_2^{(i)}$

Then the sum of square of projections is:

$$SS = \sum_i (\vec{u}^T \tilde{x}^{(i)})^2 = \sum_i (u_1 \tilde{x}_1^{(i)} + u_2 \tilde{x}_2^{(i)})^2 = \sum_i \underline{u_1^2 \tilde{x}_1^{(i)2} + 2u_1 u_2 \tilde{x}_1^{(i)} \tilde{x}_2^{(i)} + u_2^2 \tilde{x}_2^{(i)2}}$$

Equivalent

Arrange normalized sample data in a matrix: $\tilde{X} = \underbrace{\begin{bmatrix} \tilde{x}_1^{(1)} & \tilde{x}_2^{(1)} \\ .. & .. \\ \tilde{x}_1^{(m)} & \tilde{x}_2^{(m)} \end{bmatrix}}_{\text{Normalized data}}$

m is the number of samples
and in general we will have n features
here n is 2. So the matrix is *m x n*

$$\Rightarrow \quad \underline{SS = \vec{u}^T (\tilde{X}^T \tilde{X}) \vec{u}}$$

Maximize mean $SS \Rightarrow$ maximize $\frac{1}{m} SS = \vec{u}^T (\frac{1}{m} \tilde{X}^T \tilde{X}) \vec{u}$

Under constraint $||\vec{u}|| = 1$

How to compute principal components (cont.)

A typical constrained optimization problem:

Maximize $\frac{1}{m}SS = \vec{u}^T(\frac{1}{m}\tilde{X}^T\tilde{X})\vec{u}$, with $||\vec{u}|| = 1$

Measures how much x_1 and x_2 are related (move in the same direction)

variance of x_1

covariance between x_1 and x_2

$$\frac{1}{m}\tilde{X}^T\tilde{X} = \frac{1}{m}\begin{bmatrix} \frac{x_1^{(1)}-\mu_1}{\sigma_1} .. & \frac{x_1^{(m)}-\mu_1}{\sigma_1} \\ \frac{x_2^{(1)}-\mu_2 ..}{\sigma_2} & \frac{x_2^{(m)}-\mu_2}{2} \end{bmatrix} \begin{bmatrix} \frac{x_1^{(1)}-\mu_1}{\sigma_1} & \frac{x_2^{(1)}-\mu_2}{\sigma_2} \\ \cdots & \cdots \\ \frac{x_1^{(m)}-\mu_1}{\sigma_1} & \frac{x_2^{(m)}-\mu_2}{\sigma_2} \end{bmatrix} = \frac{1}{n}\underbrace{\begin{bmatrix} \boxed{\sum \frac{(x_1^{(i)}-\mu_1)^2}{\sigma_1^2}} & \boxed{\sum \frac{(x_1^{(i)}-\mu_1)(x_2^{(i)}-\mu_2)}{\sigma_1\sigma_2}} \\ \boxed{\sum \frac{(x_1^{(i)}-\mu_1)(x_2^{(i)}-\mu_2)}{\sigma_1\sigma_2}} & \boxed{\sum \frac{(x_2^{(i)}-\mu_2)^2}{\sigma_2^2}} \end{bmatrix}}_{2\times 2 \text{ covariance matrix for n=2 (in general } n \times n)}$$

variance of x_2

2×2 covariance matrix for n=2 (in general *n x n*)

Sometimes denoted as Σ

To solve this maximization problem, use the ***Lagrange Multiplier*** method:

maximize (unconstrained): $L = \vec{u}^T\left(\frac{1}{m}\tilde{X}^T\tilde{X}\right)\vec{u} - \boldsymbol{\lambda}(\vec{u}^T\vec{u} - 1)$

$$\frac{\partial L}{\partial \vec{u}} = \left(\frac{1}{m}\tilde{X}^T\tilde{X}\right)\vec{u} - \boldsymbol{\lambda}\vec{u} = 0 \implies \left(\frac{1}{m}\tilde{X}^T\tilde{X}\right)\vec{u} = \boldsymbol{\lambda}\vec{u}$$

Solution $\vec{u}$ is the principal ***eigenvector*** of $\frac{1}{m}\tilde{X}^T\tilde{X}$

The maximized *SS* is the corresponding eigenvalue, representing variance along the Eigen-Vector

Linear Algebra Review

"Eigen" is the Dutch word meaning "my"

Eigenvectors $\approx$ "my vectors"

Eigenvectors and **eigenvalues**

For square matrix A and column vector $\vec{v}$, if

$$A\vec{v} = \lambda\vec{v}$$

Then $\vec{v}$ is the eigenvector (or characteristic vector) of A, and λ is the eigenvalue (or characteristic value)

Meanings:

A is a linear transformation. Almost all vectors change direction when they are transformed by A (i.e., multiplied by A)

Certain exceptional vectors $\vec{v}$ are in the same direction of $A\vec{v}$. Those are eigenvectors.
Transform an eigenvector by A, the resulting vector $A\vec{v}$ is a number λ times the original $\vec{v}$.

Linear algebra reviews (cont.)

Spectral theorem:

If the $n \times n$ matrix A is *symmetric* ($A^T = A$), then A is orthogonally diagonalizable and has only real eigenvalues.

In other words, there exist real numbers $\lambda_1, \cdots, \lambda_n$ (***eigenvalues***) and orthogonal, non-zero real vectors $\vec{v}_1, \cdots, \vec{v}_n$ (***eigenvectors***) such that for each $i = 1,2, \cdots, n$:

$$A\vec{v}_i = \lambda_i \vec{v}_i$$

If $\widetilde{\boldsymbol{X}}$ is a $\mathrm{m} \times n$ matrix, then the $n\times n$ matrix $\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\widetilde{\boldsymbol{X}}$ and the $m\times m$ matrix $\widetilde{\boldsymbol{X}}\,\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}$ are both symmetric.

The matrices $\frac{1}{m}\widetilde{\boldsymbol{X}}\,\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}$ and $\frac{1}{m}\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\widetilde{\boldsymbol{X}}$ share the same **nonzero** eigenvalues.

Proof: let $\vec{v}$ be a nonzero eigenvector of $\frac{1}{m}\widetilde{\boldsymbol{X}}\,\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}$ with eigenvalue $\lambda \neq 0$

$$\left(\frac{1}{m}\widetilde{\boldsymbol{X}}\,\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\right)\vec{v} = \lambda\vec{v}$$

Multiply both sides with $\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}$ on the left:

$$\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\left(\frac{1}{m}\widetilde{\boldsymbol{X}}\,\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\right)\vec{u} = \widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\lambda\vec{v} \Rightarrow \left(\frac{1}{m}\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\widetilde{\boldsymbol{X}}\right)(\underline{\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\vec{v}}) = \lambda(\underline{\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\vec{v}})$$

eigenvector

Usage:

if *m* and *n* are very different, say, X is 2 X 500

Then the quick way to find the eigenvalues of the 500×500 matrix X^TX is to first find the 2 eigenvalues of the 2×2 matrix XX^T.

The other 498 eigenvalues of X^TX are all zero!

Eigenvalues and eigenvectors of covariance matrix

When X is m x 2

There are 2 eigenvectors and eigenvalues

$$\Sigma = \frac{1}{n}\tilde{X}^T\tilde{X} = \begin{bmatrix} Var(x_1) & Cov(x_1, x_2) \\ Cov(x_1, x_2) & Var(x_2) \end{bmatrix}$$ is a symmetric 2×2 matrix $\Rightarrow$

$\Sigma u_1 = \lambda_1 u_1$ $\Sigma u_2 = \lambda_2 u_2$ Let $\lambda_1 > \lambda_2$

Total variance in data $T = \lambda_1 + \lambda_2$ PC1 u_1 accounts for $\frac{\lambda_1}{T}$ of the total variance PC2 u_2 accounts for $\frac{\lambda_2}{T}$

What if X is $m \times n$ → high dimensional data with *n* features

There are *n* eigenvectors and eigenvalues

$$\Sigma = \frac{1}{n}\tilde{X}^T\tilde{X} = \begin{bmatrix} Var(x_1) & Cov(x_1, x_2) \dots Cov(x_1, x_n) \\ Cov(x_1, x_2) & Var(x_2) \quad \cdots Cov(x_2, x_n) \\ \vdots & \vdots \quad \ddots \quad \vdots \\ Cov(x_1, x_n) & Cov(x_2, x_n) \cdots \quad Var(x_n) \end{bmatrix}$$ is a symmetric $n \times n$ matrix $\Rightarrow$

$\Sigma u_1 = \lambda_1 u_1$ $\Sigma u_2 = \lambda_2 u_2$ $\cdots$ $\Sigma u_n = \lambda_n u_n$

Let $\lambda_1 > \lambda_2 > \cdots > \lambda_n$

Total variance in data $T = \lambda_1 + \lambda_2 + \cdots + \lambda_n$ PC1 u_1 accounts for $\frac{\lambda_1}{T}$ of the total variance PC2 u_2 accounts for $\frac{\lambda_2}{T}$

$\cdots$ PC*n* u_n accounts for $\frac{\lambda_n}{T}$

Finding *eigenvalues* and *eigenvectors* in Python

Use the `linalg` module of `numpy`

```
In [2]: from numpy import linalg as LA
        import numpy as np

In [3]: X = np.array([[10,11,8,3,2,1], [6,4,5,3,2.8,1]])

In [6]: w, v = LA.eig(np.dot(X, X.T))

In [7]: print(w)
        print(v)

[386.37312455   7.46687545]
[[ 0.87715849 -0.48020099]
 [ 0.48020099  0.87715849]]
```

What's next? Reduce to *k* dimensions

X is $m\ x\ n$ $\quad \Sigma = \frac{1}{m}\tilde{X}^T\tilde{X}$ is $m \times m$ $\quad$ eigenvectors and eigenvalues: λ_i, u_i

Select the top *k* eigenvectors $u_1, u_2, \cdots, u_k$, with top *k* eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_k$

$\boldsymbol{U} = [\boldsymbol{u_1}\ \boldsymbol{u_2}\ \ldots \boldsymbol{uk}]$ Span to a k-dimensional subspace ⇒ Project data point $\tilde{\boldsymbol{x}}^{(i)}$ into this subspace

$n\ x\ k$ matrix

$$\boldsymbol{y}^{(i)} = \underbrace{\tilde{\boldsymbol{x}}^{(i)}}_{1\ x\ n}\boldsymbol{U} = \underbrace{[y_1{}^{(i)}\ y_2{}^{(i)} \ldots . y_k{}^{(i)}]}_{1\ x\ k}$$

Project *m* examples into new subspace

$$\underbrace{\boldsymbol{Y}}_{m\ x\ k} = \underbrace{\tilde{\boldsymbol{X}}}_{m\ x\ n}\boldsymbol{U} \text{ ———— } n\ x\ k$$

Reduce *n*-dimensional data to *k*-dimensional

Y are the coordinates of data in the new subspace

For visualization, k = 2 or 3

Reconstruction of Original Data from PCs

- Let $\widetilde{\boldsymbol{x}}$ be the original data and its projections on the first *k* PCs ($\boldsymbol{u_1},\boldsymbol{u_2},\ldots\boldsymbol{u_k}$) are given by:
  - $\boldsymbol{y_j} = \widetilde{\boldsymbol{x}}\,\boldsymbol{u}_j$ for *j= 1,2,..k*
- Reconstructed data $\widetilde{\boldsymbol{x}'}$ is given by:
  - $\widetilde{\boldsymbol{x}'} = \sum_{j=1}^{k} \boldsymbol{y}_j . \boldsymbol{u}_j^{\top}$
- If the full set of PCs are used, then the reconstruction will be perfect, i.e., exactly the same as the original image without losing any information.
- If a subset (e.g., top *k* PCs) is used, then the reconstruction will cause some information loss.
- This information loss can be measured by the Euclidean distance between the original data $\widetilde{\boldsymbol{x}}$ and the reconstructed data $\widetilde{\boldsymbol{x}'}$. Larger distance indicates higher information loss.

Covariance matrix of the transformed data

Y is the transformed data

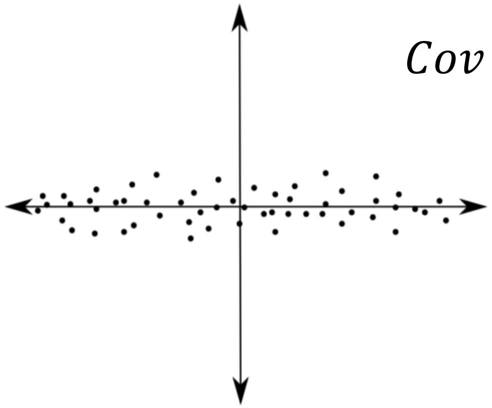
Its covariance matrix: $\Sigma = \frac{1}{m} Y^T Y \quad = \frac{1}{m} (\tilde{X} U)^T (\tilde{X} U) = \frac{1}{m} U^T \tilde{X}^T \tilde{X} U \quad = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_k \end{bmatrix}$

A diagonal matrix with eigenvalues as diagonal elements

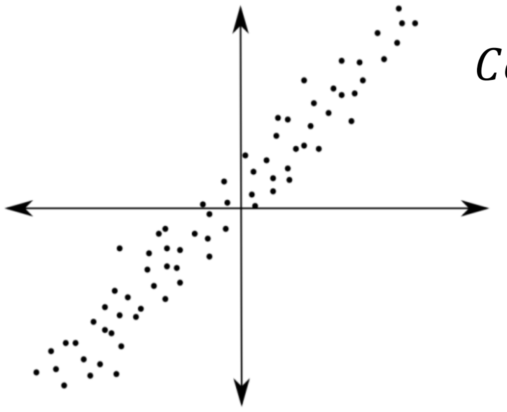
Therefore, all covariances $Var(y_i, y_j)$ are **zero**! → All dimensions are independent

---

Covariance matrix and the shape of data



$$Cov = \begin{bmatrix} 95 & 1 \\ 1 & 5 \end{bmatrix}$$



$$Cov = \begin{bmatrix} 50 & 40 \\ 40 & 50 \end{bmatrix}$$

Summary: procedures of performing PCA

Normalize data: $X \to \tilde{X}, m \times n$

Find the eigenvectors and eigenvalues of the covariance matrix $\frac{1}{m}\tilde{X}^T\tilde{X}, u_1, u_2, \cdots, u_n, \lambda_1 > \lambda_2 > \cdots > \lambda_n$

Observe whether a small number of the λ_i are much bigger than all the others, e.g., first *k* elements.

- If yes, a dimension reduction is proper: *n*-dim → *k*-dim

Interpret the principal components:

- Which ones are the most important?

Applications of PCA

- *Better interpretation of features*

Case 1: Sibley's Bird Database of North American birds

	Length, x_1	**Wingspan, x_2**	**Weight, x_3**
Bird 1	4	10	6
Bird 2	5	11	4
Bird 3	6	8	5
…	…	…	…

$$\lambda_1 = 1626.5 \qquad \lambda_2 = 129.0 \qquad \lambda_3 = 7.1$$

$$u_1 = \begin{bmatrix} 0.22 \\ 0.41 \\ 0.88 \end{bmatrix} \qquad u_2 = \begin{bmatrix} 0.25 \\ 0.85 \\ -0.46 \end{bmatrix} \qquad u_3 = \begin{bmatrix} 0.94 \\ -0.32 \\ -0.08 \end{bmatrix}$$

X is $100\ x\ 3$

$$\Sigma = \begin{bmatrix} 91.4 & 171.9 & 298.0 \\ & 373.9 & 545.2 \\ & & 1297.3 \end{bmatrix}$$

$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} = 92.3\% \qquad \frac{\lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} = 7.3\%$$

PC1 = 0.22 Length + 0.41 Wingspan + 0.88 Weight

It characterizes the "size"

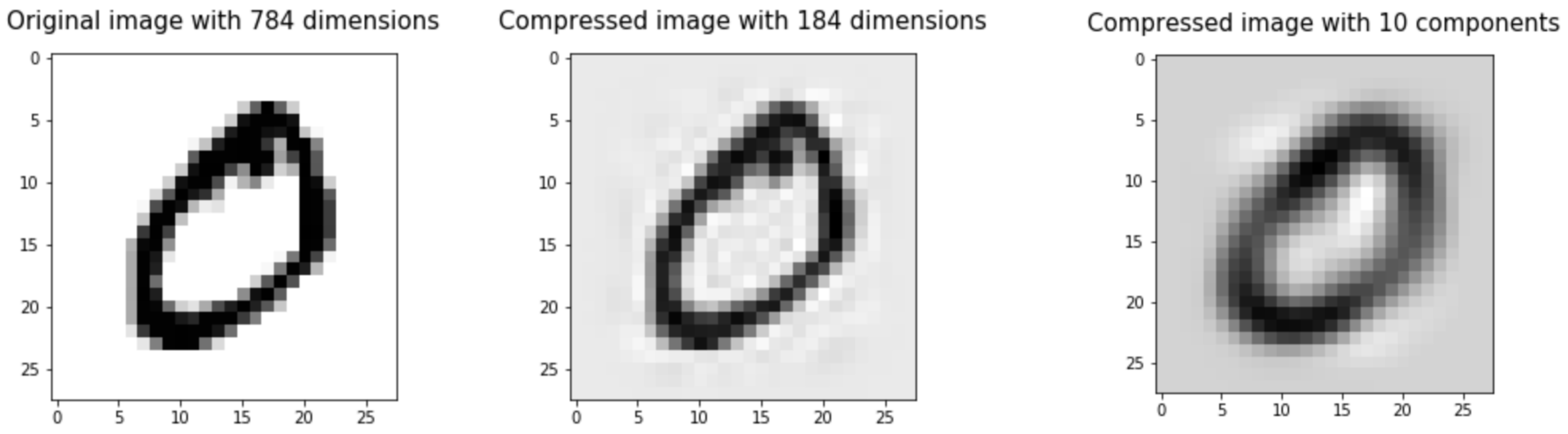
PC2 = 0.25 Length + 0.85 Wingspan – 0.46 Weight

It characterizes the "stoutness"

Applications of PCA

Data compression

- Original data: $X \in \mathbb{R}^n$
- Use the top k PCs to approximate: $X \cong X^k$



Source: https://towardsdatascience.com/image-compression-using-principal-component-analysis-pca-253f26740a9f

Applications of PCA: Eigenface

A training set of *m* face images. Each image is 100×100

Flatten each image to column vector and stack them → X of shape $m\ x\ 10000$ (normalized)

Compute the eigenvectors and eigenvalues of the covariance matrix $\frac{1}{m}X^TX$, of shape 10000×10000

- Each eigenvector is of shape 10000×1
- Reshape it back to 100×100 → It can be seen as a face image: "eigenface"

These 10000 eigenfaces can be used to represent new faces

- Project a new image vector to a selected subset of eigenfaces
- The projections can be used to identify the new face
- In practice, choosing **100 to 150** eigenfaces are sufficient

Example of the first eigenvector (PC1)
From Jauregui (2012)

Assignment

- Assignment uses sign-language digits data – provided in a numpy array
- The original data can be obtained from https://www.kaggle.com/ardamavi/sign-language-digits-dataset (preprocessed) or https://github.com/ardamavi/Sign-Language-Digits-Dataset (raw).