CMPSC 448: Machine Learning

Lecture 14. Principal Component Analysis

Rui Zhang

Fall 2021

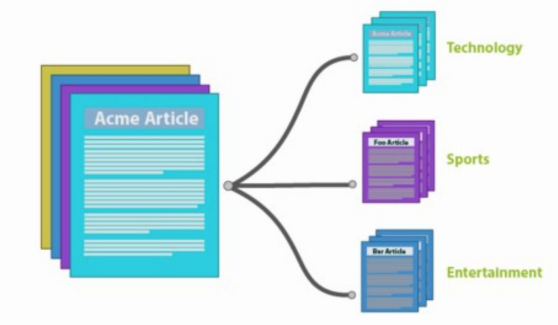
PennState

Outline

- Dimensionality Reduction
- Principal Component Analysis (PCA)
  - View I: Maximum Variance and View II: Minimum Reconstruction Error
  - only one principal component and general form with arbitrary number of principal component
- Decide number of principal components
- Limitations of PCA
- Autoencoders

Big and High-dimensional data

High dimensionality = lot of features



Document classification

- Features per document = thousands of words/unigrams millions of bigrams, contextual information

Netflix rating matrix (recommender system)

- 480189 users x 17770 movies

Big and High-dimensional data

- Large number of training data (big)
  - We already learned about SGD for dealing with big number of training data
- Huge number of features (high dimensional)
  - How about huge number of features?
    Useful to learn lower dimensional representations of the data.

Representation Learning

Input: raw feature vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

Goal: learn a useful/informative feature transformation $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}^k$

Examples:

- High-dimensional feature mapping in kernel methods
- Centering training data $\boldsymbol{x}_i \longrightarrow (\boldsymbol{x}_i - \boldsymbol{\mu})$, where $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$
- Standardization: all features have mean **0** and unit variance
- Feature (representation) learning in deep neural networks
- If $k \ll d$, **dimensionality reduction**

Principal Component Analysis - Overview

PCA is an **unsupervised** technique for extracting hidden low-rank structure from high-dimensional data.

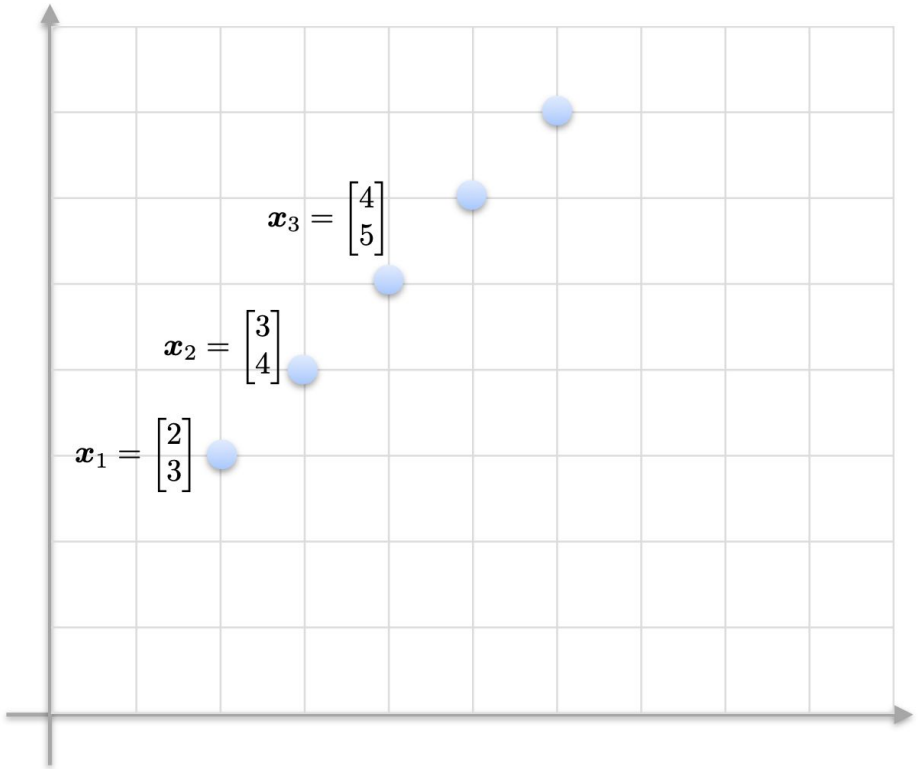
PCA performs **dimensionality reduction** by an orthogonal linear projection or transformation of the data onto a **lower dimensional subspace that preserves most of information in data**.

PCA is about getting rid of "uninformative information" by replacing Redundant/Duplicated/Noisy features with a few new features that adequately summarize information contained in the original feature space

Principal Component Analysis - Applications

- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)
- Further processing by machine learning algorithms

Dimensionality Reduction

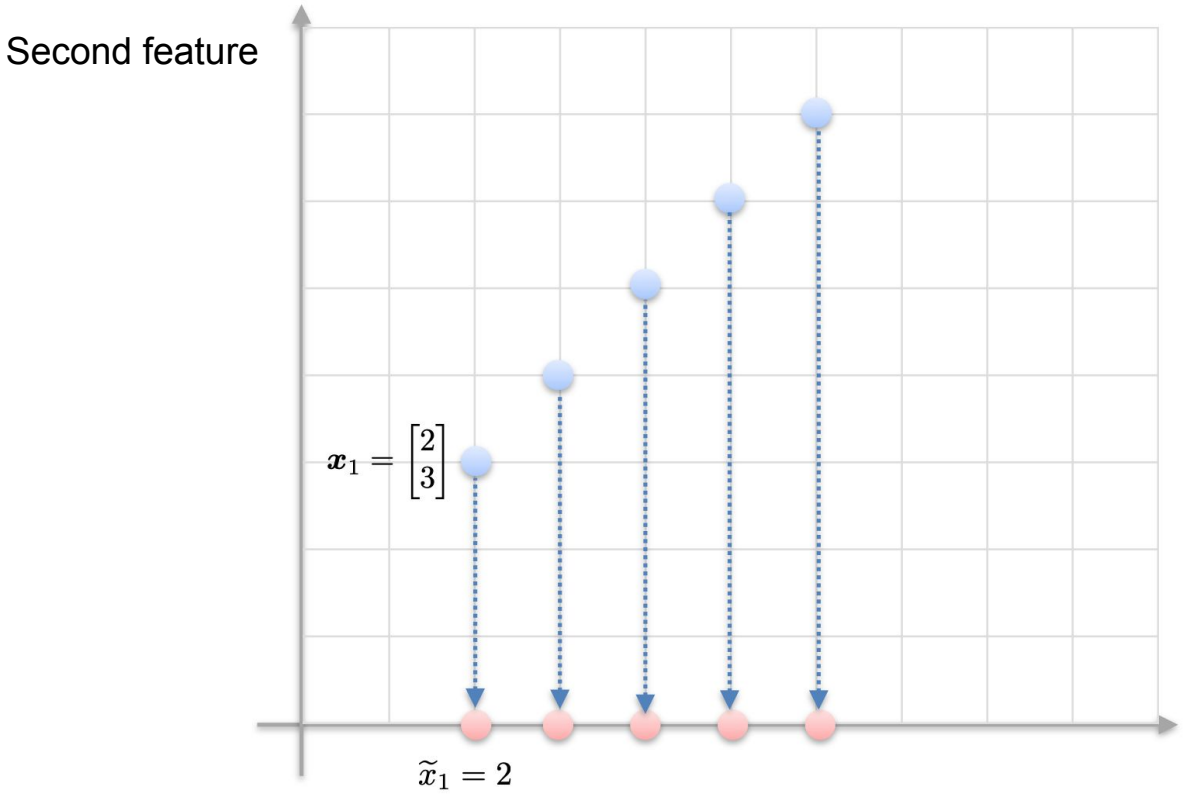


Assume you have 5 data points each with two features

But, you can only store 5 floating point numbers!

Any proposal?

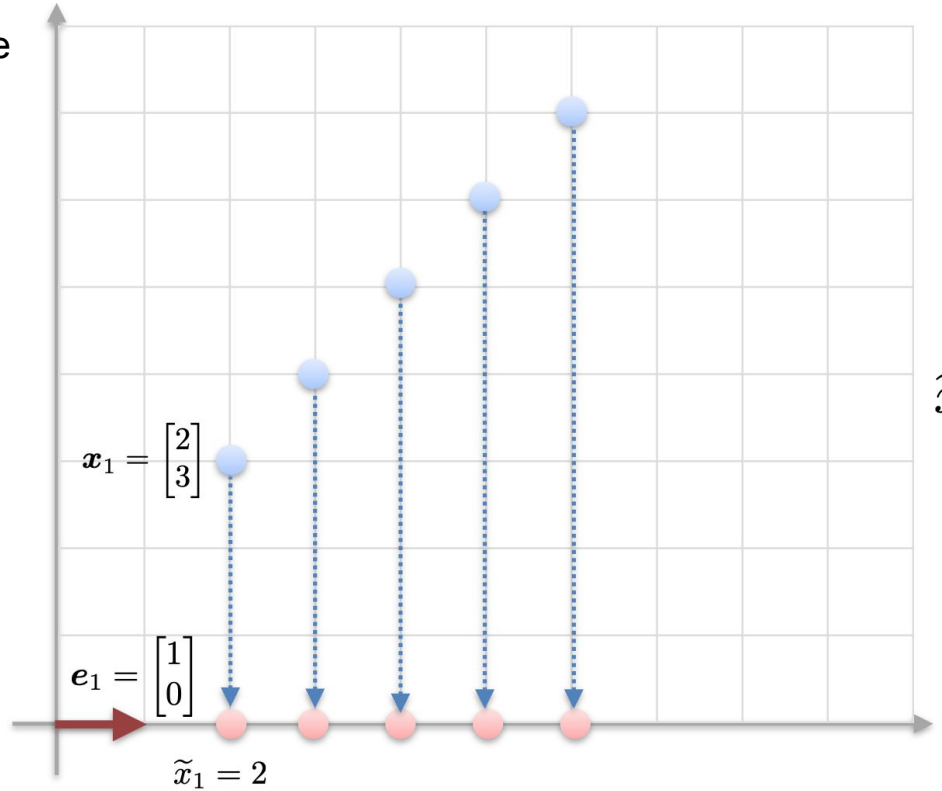
Dimensionality Reduction



Let's get rid of second feature!

The second feature becomes zero and we just drop it!

Dimensionality Reduction via Projection

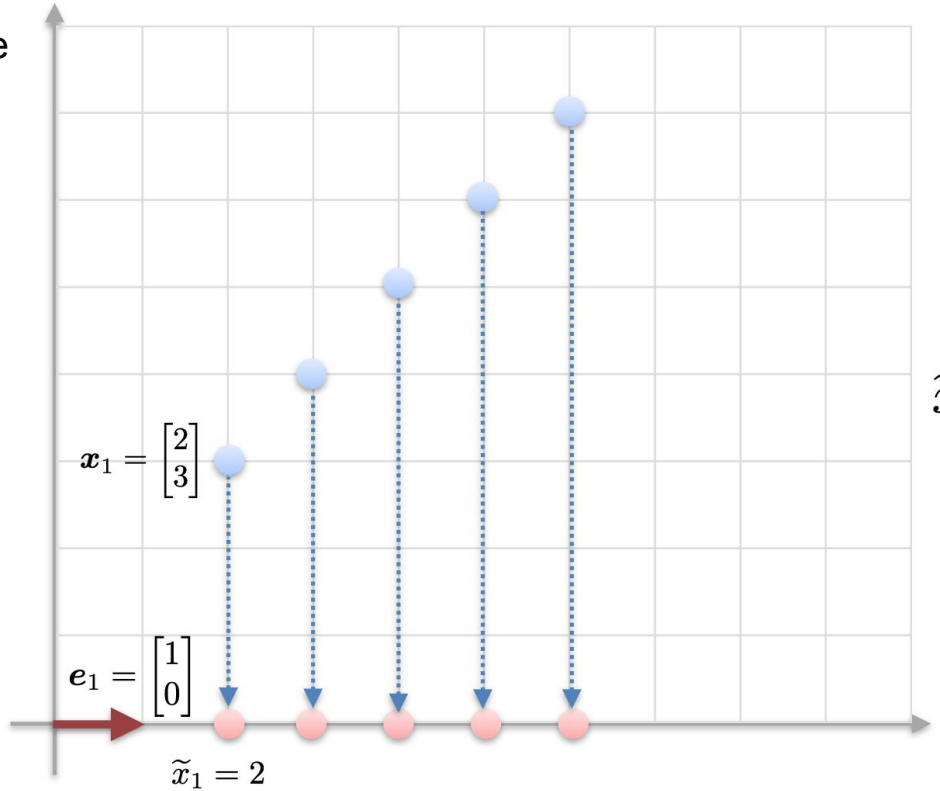


Dropping the second feature is same is projecting the point on the horizontal axes with normal vector:

$$\boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\widetilde{x}_1 = \boldsymbol{e}_1^\top \boldsymbol{x}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = 2$$

Dimensionality Reduction via Projection



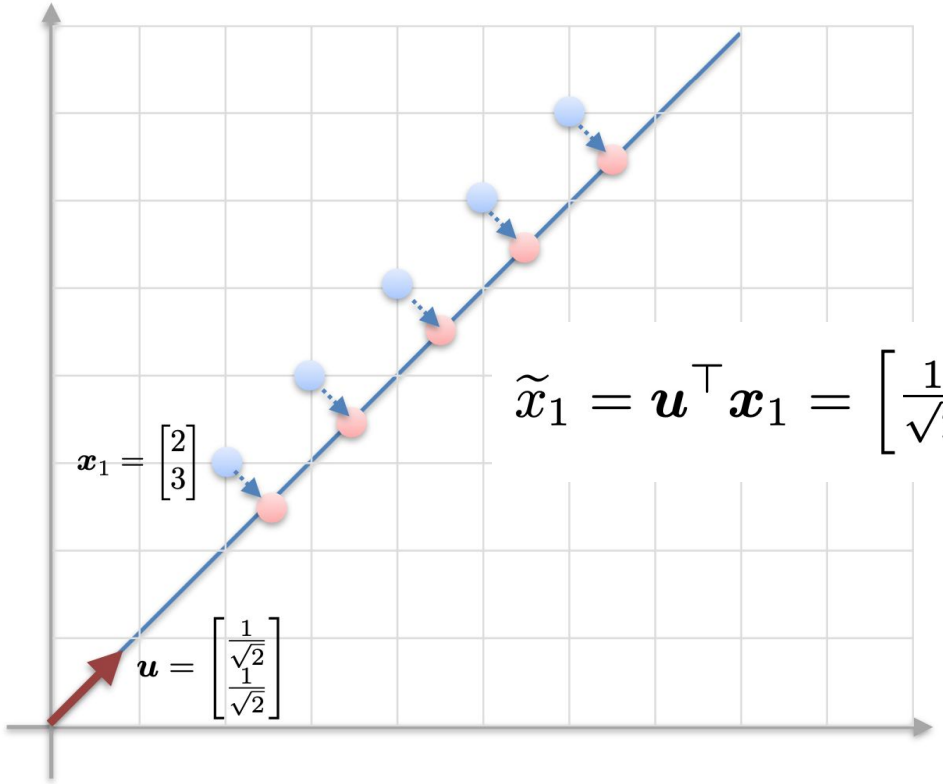
Dropping the second feature is same is projecting the point on the horizontal axes with normal vector:

$$\boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\widetilde{x}_1 = \boldsymbol{e}_1^\top \boldsymbol{x}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = 2$$

This is BAD :-(We lost all the information of the second feature!

Dimensionality Reduction via Projection



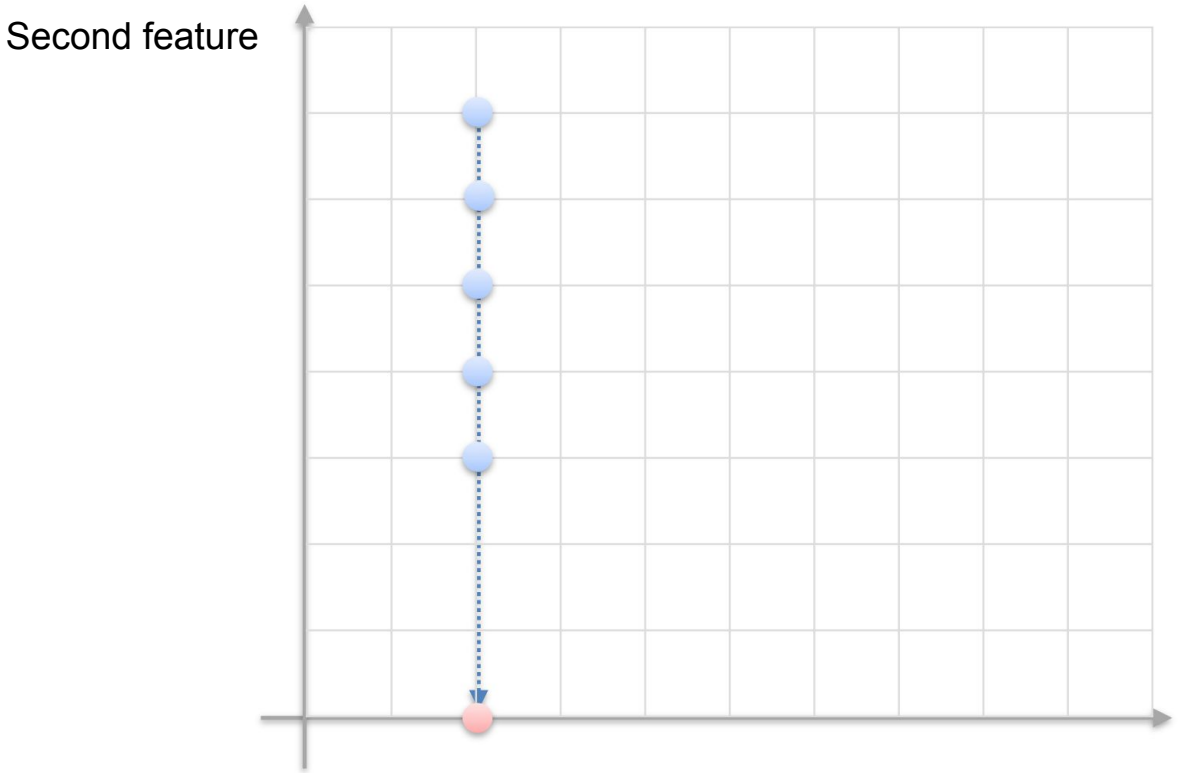
$$\widetilde{x}_1 = \boldsymbol{u}^\top \boldsymbol{x}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \frac{2}{\sqrt{2}} + \frac{3}{\sqrt{2}}$$

Let's be smart and pick another line with normal vector:

$$\boldsymbol{u} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

This is BETTER :-) At least we add the features together rather than fully ignoring one of them!

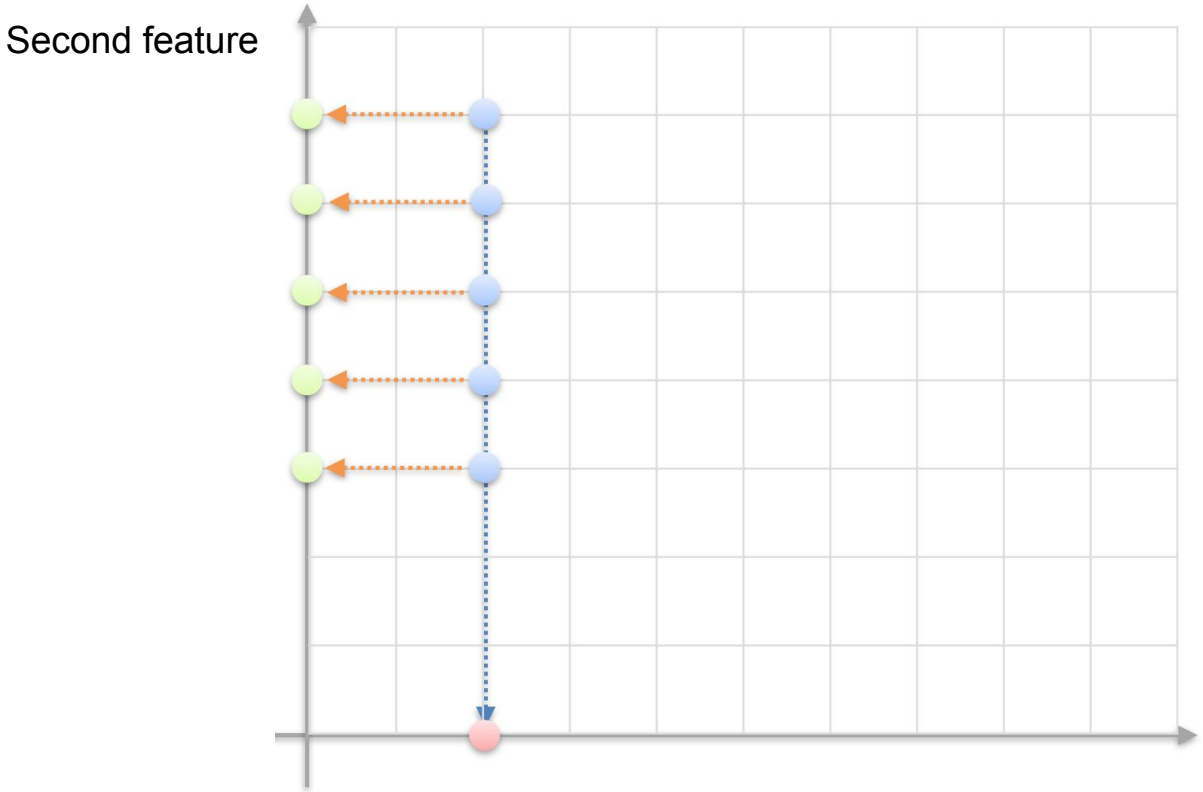
Dimensionality Reduction via Projection



This is TERRIBLE :-(

All the points collapse into a single number!

Dimensionality Reduction via Projection

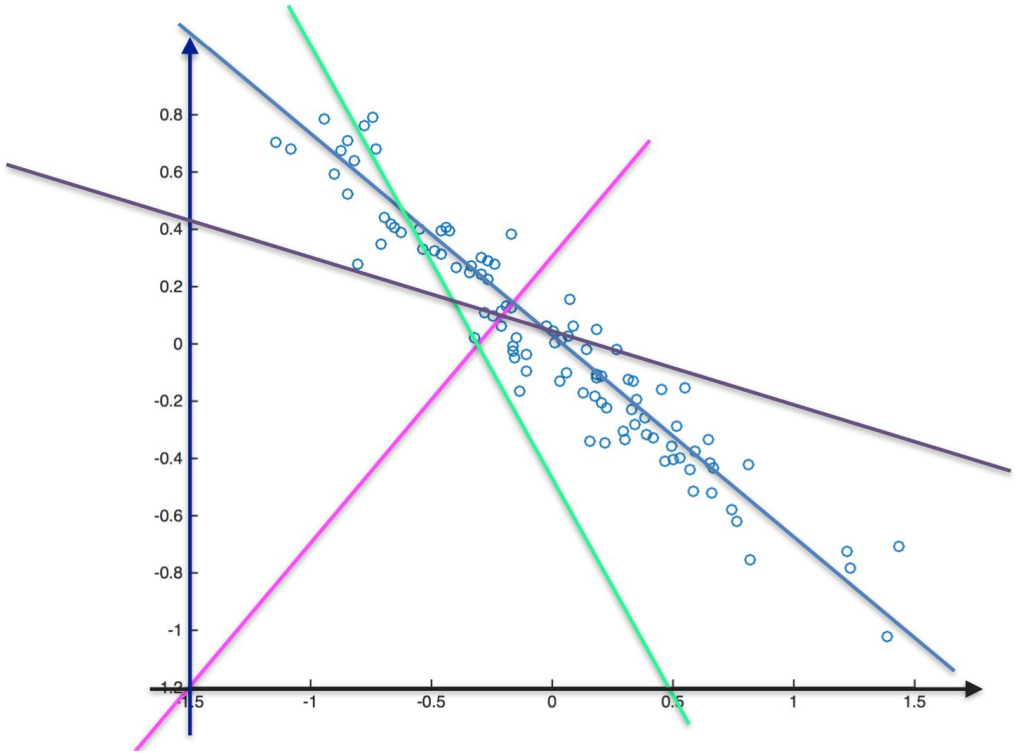


Would be perfect if we projected onto vertical axes!

Note that the first feature is same of all samples!

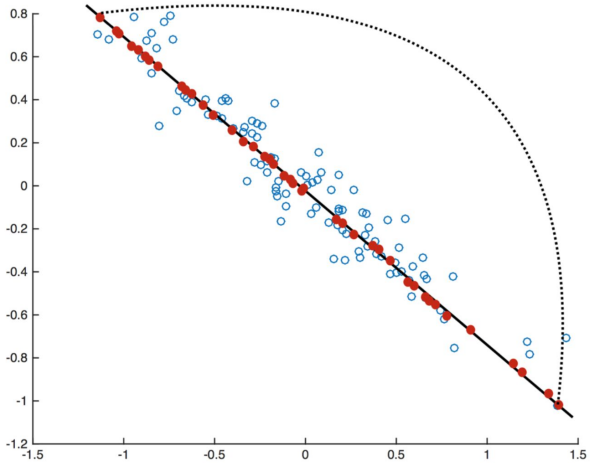
Dimensionality Reduction

Question: which direction (line) you choose to project the following 2-d data onto a 1-d subspace (i.e., a line)?



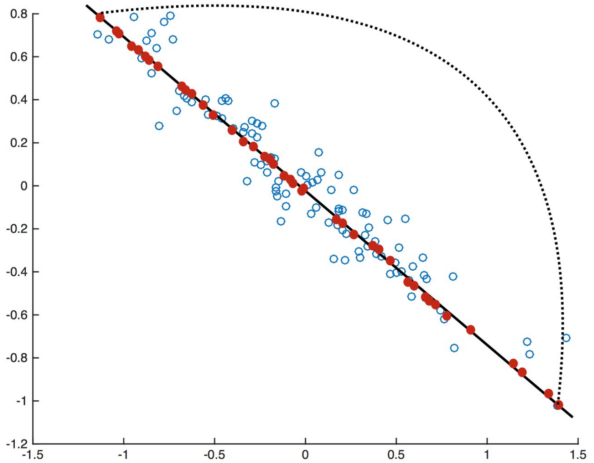
Two interpretations of PCA

View I: **Maximum Variance**. PCA finds a subspace such that projections onto the subspace capture maximum variance in the data.

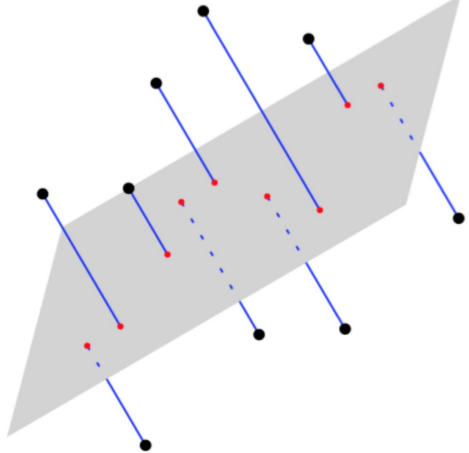


Two interpretations of PCA

View I: **Maximum Variance**. PCA finds a subspace such that projections onto the subspace capture maximum variance in the data.

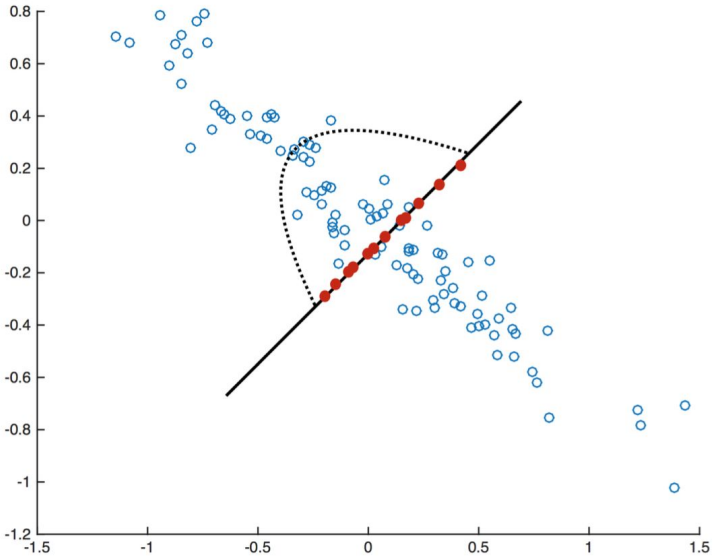


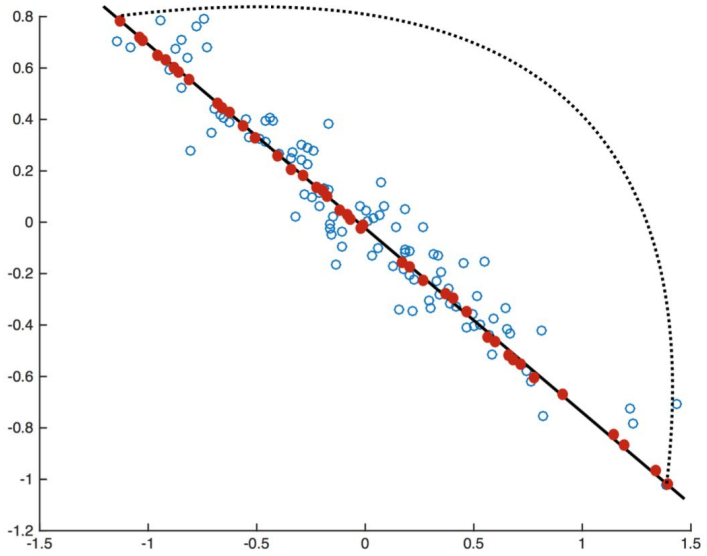
View II: **Minimum Reconstruction Error**. PCA finds a subspace that projection onto the subspace yields minimum the Mean squared error (MSE) reconstruction.



View I: Maximum Variance

Basic idea: Pick the directions along which data is maximally spread (or variance is high).

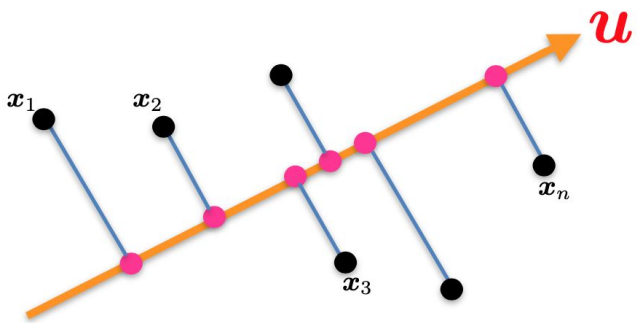




Here we would like to pick the second option as the spread of the points across the chosen direction is larger in the second figure.

How to formulate it?

We want to pick the direction (line) along which the variance of projected points is largest.



$$x_1 \longrightarrow \langle u, x_1 \rangle u$$

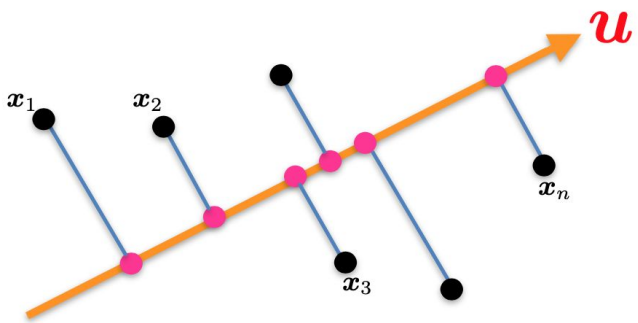
$$x_2 \longrightarrow \langle u, x_2 \rangle u$$

$$\vdots$$

$$x_n \longrightarrow \langle u, x_n \rangle u$$

How to formulate it?

We want to pick the direction (line) along which the variance of projected points is largest.



$$\boldsymbol{x}_1 \longrightarrow \langle \boldsymbol{u}, \boldsymbol{x}_1 \rangle \boldsymbol{u}$$

$$\boldsymbol{x}_2 \longrightarrow \langle \boldsymbol{u}, \boldsymbol{x}_2 \rangle \boldsymbol{u}$$

$$\vdots$$

$$\boldsymbol{x}_n \longrightarrow \langle \boldsymbol{u}, \boldsymbol{x}_n \rangle \boldsymbol{u}$$

We need to compute the variance of projected points (each point becomes a number after projection if we ignore the direction)

How to formulate it?

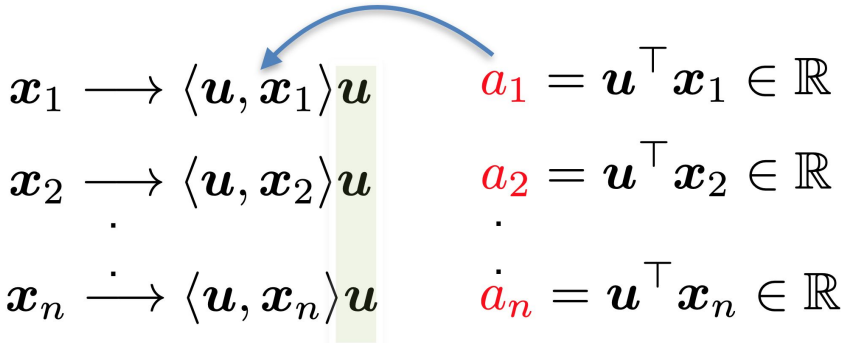
The variance of a set of numbers $a_1, a_2, \ldots, a_n$ is:

1. Subtract the mean from each value in the data (centering data). This gives you a measure of the distance of each value from the mean.
2. Square each of these distances (so that they are all positive values), and add all of the squares together.
3. Divide the sum of the squares by the number of values in the data set.

$$\frac{1}{n}\sum_{i=1}^{n}\left(a_i - \left(\frac{1}{n}\sum_{j=1}^{n} a_j\right)\right)^2$$

How to formulate it?

When we project data onto a line with unit vector $\boldsymbol{u}$, the value of each data point becomes a real number:



$$\begin{aligned} \boldsymbol{x}_1 &\longrightarrow \langle \boldsymbol{u}, \boldsymbol{x}_1 \rangle \boldsymbol{u} & a_1 &= \boldsymbol{u}^\top \boldsymbol{x}_1 \in \mathbb{R} \\ \boldsymbol{x}_2 &\longrightarrow \langle \boldsymbol{u}, \boldsymbol{x}_2 \rangle \boldsymbol{u} & a_2 &= \boldsymbol{u}^\top \boldsymbol{x}_2 \in \mathbb{R} \\ &\vdots & &\vdots \\ \boldsymbol{x}_n &\longrightarrow \langle \boldsymbol{u}, \boldsymbol{x}_n \rangle \boldsymbol{u} & a_n &= \boldsymbol{u}^\top \boldsymbol{x}_n \in \mathbb{R} \end{aligned}$$

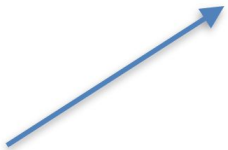
The variance of projected 1 dimensional points becomes:

$$\frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{u}^\top \boldsymbol{x}_i - \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{u}^\top \boldsymbol{x}_j \right)^2$$

How to formulate it?

That is we want to pick the direction that maximize the sample variance in the projected direction which is given by :

$$\arg \max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2^2=1} \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{u}^\top \boldsymbol{x_i} - \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{u}^\top \boldsymbol{x_j} \right)^2$$



The average of projected points

How to formulate it?

$$\arg \max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2^2=1} \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{u}^\top \boldsymbol{x_i} - \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{u}^\top \boldsymbol{x_j} \right)^2$$

How to formulate it?

$$\arg \max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2^2=1} \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{u}^\top \boldsymbol{x_i} - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{u}^\top \boldsymbol{x_j}\right)^2$$

$$\equiv \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{u}^\top\left(\boldsymbol{x_i} - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{x_j}\right)\right)^2$$

$$\boldsymbol{\mu} = (1/n)\sum_{i=1}^{n}\boldsymbol{x}_i$$

$$\equiv \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{u}^\top(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top\boldsymbol{u}$$

Sample Covariance

$$\equiv \boldsymbol{u}^\top\left(\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top\right)\boldsymbol{u}$$

Computing the Covariance

	Hours(H)	*Mark(M)*
Data	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42

H	M	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54

How to formulate it?

$$\arg \max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2^2=1} \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{u}^\top \boldsymbol{x_i} - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{u}^\top \boldsymbol{x_j}\right)^2$$

$$\arg \max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2^2=1} \boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X}\right)\boldsymbol{u}$$

$\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is the centered data matrix (each row is a single data point minus average of all points $\boldsymbol{x}_i - \boldsymbol{\mu}$)

$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{d\times d}$ is called the covariance matrix of data points)

Sample Covariance Matrix

$$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{d\times d}$$

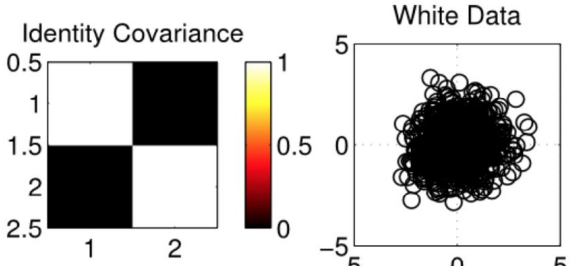
is called the covariance matrix of data points after centering the data points

The $(i,j)\text{th}$ entry of covariance matrix is the covariance between $i\text{th}$ and $j\text{th}$ features!

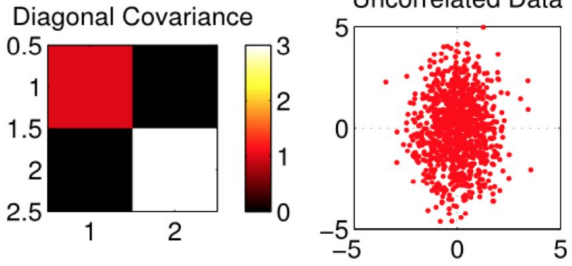
If all the features are mutually independent, then the covariance matrix would be diagonal. (If X and Y are independent variables, then their covariance is 0)

Covariance matrix

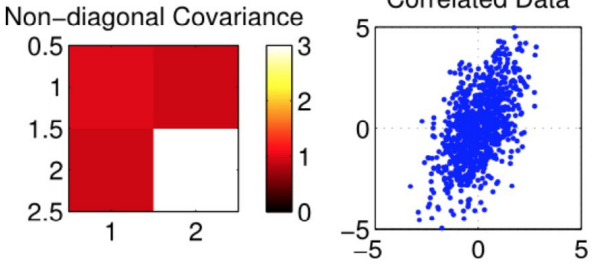
Covariance matrix for 2d Gaussian distribution with mean 0 and different covariance matrix!



Covariance matrix equal to the identity matrix, and the points drawn from the corresponding Gaussian distribution. The diagonal values are 1, indicating the data have variance of 1 along both of the dimensions. Additionally, the off-diagonal elements are zero, meaning that the two dimensions are uncorrelated.



The points that result from a diagonal, but not identity covariance matrix. The off-diagonal elements are still zero, indicating that the dimensions are uncorrelated. However, the variances along each dimension are not equal to one, and are not equal. This is demonstrated by the elongated distribution in red. The elongation is along the second dimension, as indicated by the larger value in the bottom-right (point) of the covariance matrix.



The points that result from a non-diagonal covariance matrix. Here the off-diagonal elements of covariance matrix have non-zero values, indicating a correlation between the dimensions. This correlation is reflected in the distribution of drawn datapoints (in blue). We can see that the primary axis along which the points are distributed is not along either of the dimensions, but a linear combination of the dimensions.

PCA by Maximizing Variance

We need to solve the following optimization problem to pick the direction that maximize the variance in the projected direction:

$$\arg \max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2^2=1} \boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X}\right)\boldsymbol{u}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is the **centered data matrix** (each row is a single data point minus average of all points).

PCA reduces feature space dimensionality by looking for linear correlation patterns between features

The answer of above optimization problem is the **leading eigenvector (eigenvector corresponding to the maximum eigenvalue) of the covariance matrix**

PCA by Maximizing Variance - Solution

Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

$$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$$

PCA by Maximizing Variance - Solution

Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

$$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$$

$$\boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X}\right) \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{u}$$

PCA by Maximizing Variance - Solution

Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

$$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$$

$$\boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X}\right) \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{u}$$

$$\boldsymbol{c} \coloneqq \boldsymbol{U}^\top \boldsymbol{u}$$

$$||\boldsymbol{c}||_2^2 = ||\boldsymbol{U}^\top \boldsymbol{u}||_2^2 = (\boldsymbol{U}^\top \boldsymbol{u})^\top \boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{u} = ||\boldsymbol{u}||_2^2 = 1$$

PCA by Maximizing Variance - Solution

Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

$$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$$

$$\boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X}\right) \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{u}$$

$$\boldsymbol{c} \coloneqq \boldsymbol{U}^\top \boldsymbol{u} \qquad = \boldsymbol{c}^\top \boldsymbol{\Sigma}\boldsymbol{c}$$

$$= \sum_{i=1}^{d} \lambda_i \, c_i^2$$

$$||\boldsymbol{c}||_2^2 = ||\boldsymbol{U}^\top \boldsymbol{u}||_2^2 = (\boldsymbol{U}^\top \boldsymbol{u})^\top \boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{u} = ||\boldsymbol{u}||_2^2 = 1$$

PCA by Maximizing Variance - Solution

Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

$$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$$

$$\boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X}\right) \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{u}$$

$$\boldsymbol{c} \coloneqq \boldsymbol{U}^\top \boldsymbol{u} \qquad = \boldsymbol{c}^\top \boldsymbol{\Sigma}\boldsymbol{c}$$

$$= \sum_{i=1}^{d} \lambda_i \, c_i^2 \qquad \sum_{i=1}^{d} c_i^2 = 1$$

$$||\boldsymbol{c}||_2^2 = ||\boldsymbol{U}^\top \boldsymbol{u}||_2^2 = (\boldsymbol{U}^\top \boldsymbol{u})^\top \boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{u} = ||\boldsymbol{u}||_2^2 = 1$$

PCA by Maximizing Variance - Solution

Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

$$\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$$

$$\boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X}\right) \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{u}$$

$$\boldsymbol{c} \coloneqq \boldsymbol{U}^\top \boldsymbol{u} \qquad = \boldsymbol{c}^\top \boldsymbol{\Sigma}\boldsymbol{c}$$

$$= \sum_{i=1}^{d} \lambda_i c_i^2 \qquad \sum_{i=1}^{d} c_i^2 = 1$$

$$c_1 = 1, c_2 = \cdots = c_d = 0$$

$$\boldsymbol{u} = \boldsymbol{u}_1$$

$$||\boldsymbol{c}||_2^2 = ||\boldsymbol{U}^\top \boldsymbol{u}||_2^2 = (\boldsymbol{U}^\top \boldsymbol{u})^\top \boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{u} = \boldsymbol{u}^\top \boldsymbol{u} = ||\boldsymbol{u}||_2^2 = 1$$

PCA solution ($k = 1$)

Input: data matrix $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

1. Calculate the **Sample Covariance** matrix

$$\boldsymbol{\mu} = (1/n) \sum_{i=1}^{n} \boldsymbol{x}_i \qquad \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top = \frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^d$$

Data Matrix after Centering

$$\boldsymbol{X} \in \mathbb{R}^{n \times d}$$

PCA solution ($k = 1$)

Input: data matrix $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

1. Calculate the **Sample Covariance** matrix

$$\boldsymbol{\mu} = (1/n) \sum_{i=1}^{n} \boldsymbol{x}_i \qquad \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top = \frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^d$$

Data Matrix after Centering

$$\boldsymbol{X} \in \mathbb{R}^{n \times d}$$

2. Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

PCA solution ($k = 1$)

Input: data matrix $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

1. Calculate the **Sample Covariance** matrix

$$\boldsymbol{\mu} = (1/n) \sum_{i=1}^{n} \boldsymbol{x}_i \qquad \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top = \frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^d$$

Data Matrix after Centering

$$\boldsymbol{X} \in \mathbb{R}^{n \times d}$$

2. Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

3. Compute the **the eigenvector** $\boldsymbol{u} \in \mathbb{R}^d$ corresponding to the **largest eigenvalue** of the sample covariance matrix

PCA solution ($k = 1$)

Input: data matrix $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

1. Calculate the **Sample Covariance** matrix

$$\boldsymbol{\mu} = (1/n) \sum_{i=1}^{n} \boldsymbol{x}_i \qquad \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top = \frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^d$$

Data Matrix after Centering

$$\boldsymbol{X} \in \mathbb{R}^{n \times d}$$

2. Recall that $\boldsymbol{A}^\top \boldsymbol{A}$ is a symmetric, positive semidefinite matrix, which means it has (1) non-negative real eigenvalues and (2) eigendecomposition with orthonormal eigenvectors.

3. Compute the **the eigenvector** $\boldsymbol{u} \in \mathbb{R}^d$ corresponding to the **largest eigenvalue** of the sample covariance matrix

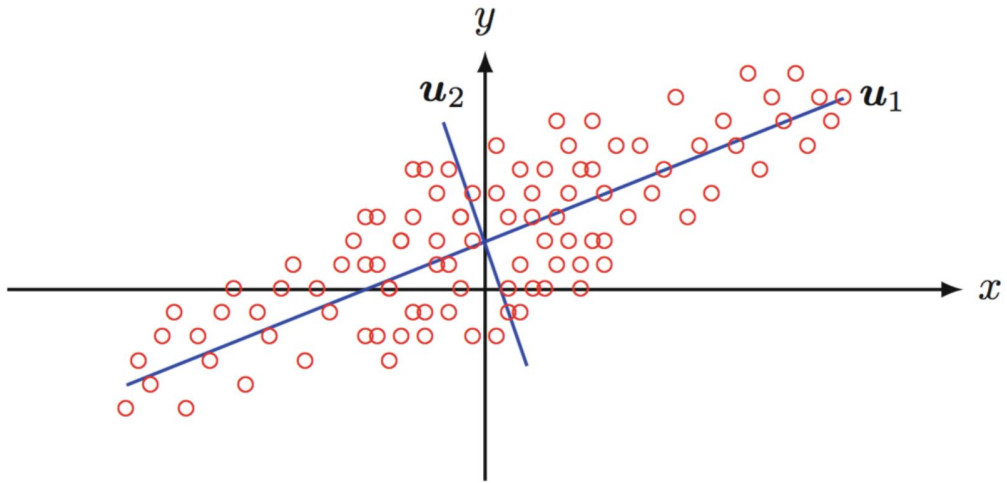
4. Map each data point to: $\boldsymbol{x}_i \mapsto \boldsymbol{u}^\top \boldsymbol{x}_i \in \mathbb{R}$

Example

$$\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^2 \Rightarrow \text{centering} \Rightarrow \frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{2\times 2}$$

1st Principle Component (eigenvector) of covariance matrix corresponding to largest eigenvalue

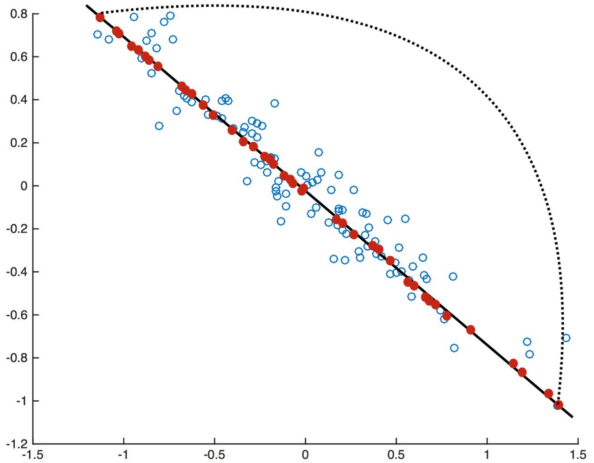
2nd Principle Component (eigenvector) of covariance matrix corresponding to 2nd largest eigenvalue



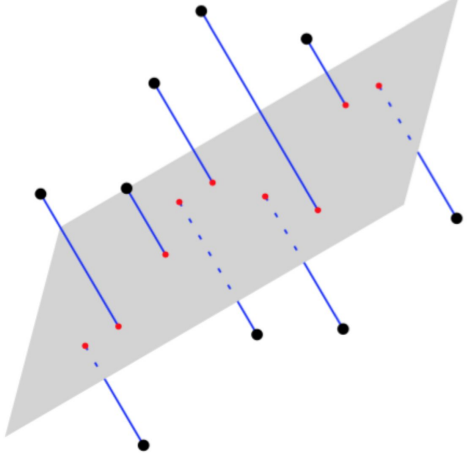
Example showing a two-dimensional data set and its two principal components.

Two interpretations of PCA

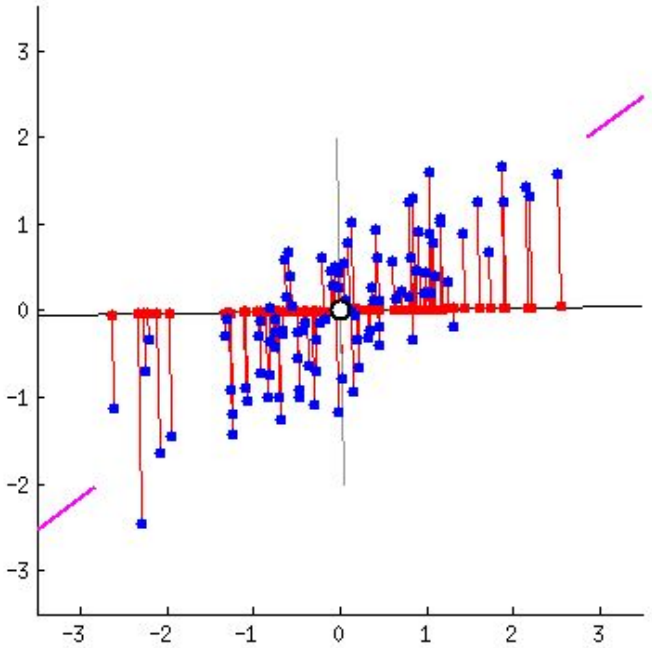
View I: **Maximum Variance**. PCA finds a subspace such that projections onto the subspace capture maximum variance in the data.



View II: **Minimum Reconstruction Error**. PCA finds a subspace that projection onto the subspace yields minimum MSE reconstruction.



Geometric Interpretation of two view equivalency



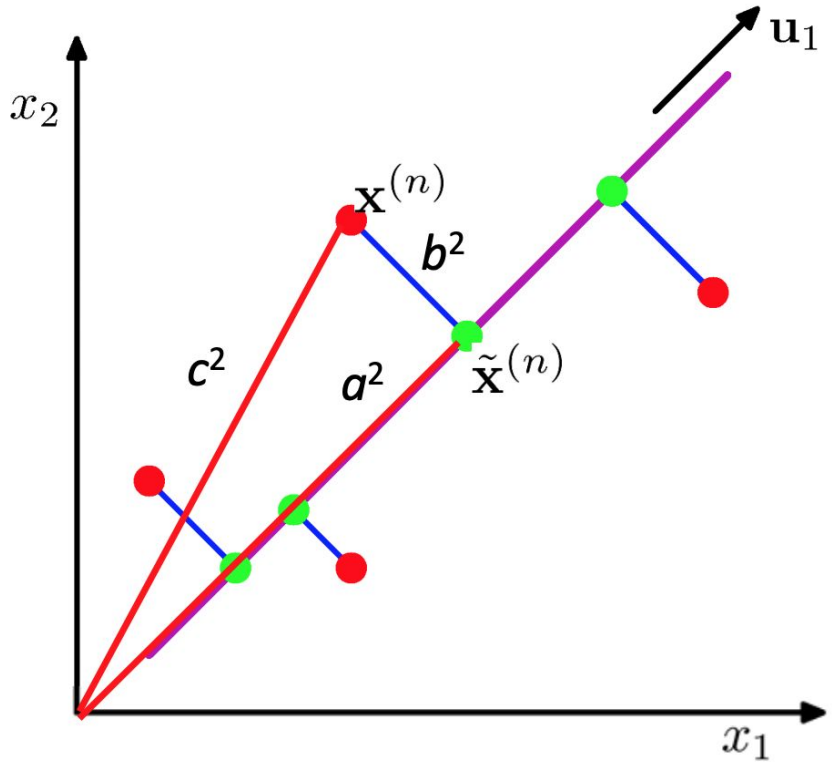
https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Geometric Interpretation of two view equivalency

- With mean at the origin

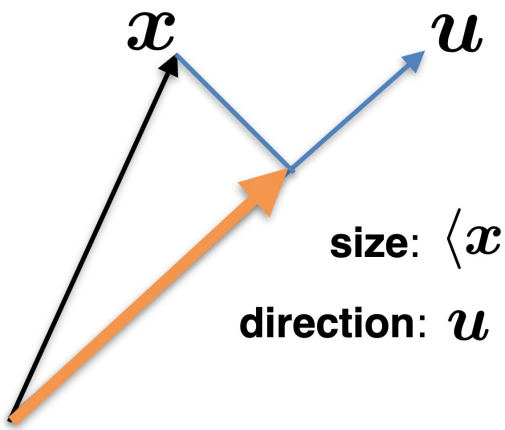
$$c_i^2 = a_i^2 + b_i^2$$

- With constant $\sum_i c_i^2$
  - Minimizing $\sum_i b_i^2$
  - Maximizes $\sum_i a_i^2$
  - And vice versa



Projection with $k = 1$

For a single dimensional mapping $k = 1$ (projection onto a subspace with rank one, or a single unit vector $\boldsymbol{u} \in \mathbb{R}^d, \|\boldsymbol{u}\|_2 = 1$):



size: $\langle \boldsymbol{x}, \boldsymbol{u} \rangle = \boldsymbol{x}^\top \boldsymbol{u}$

direction: $\boldsymbol{u}$

Projection of $\boldsymbol{x}$ onto $\boldsymbol{u}$ is:

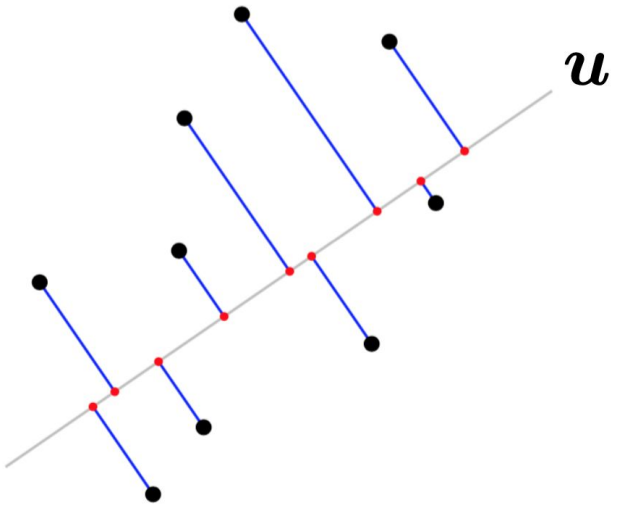
$$\langle \boldsymbol{x}, \boldsymbol{u} \rangle \boldsymbol{u} = (\boldsymbol{x}^\top \boldsymbol{u}) \boldsymbol{u} = \underbrace{(\boldsymbol{u}\boldsymbol{u}^\top)}_{d \times d \text{ matrix}} \boldsymbol{x}$$

$\Pi = \boldsymbol{u}\boldsymbol{u}^\top \in \mathbb{R}^{d \times d}$ is the projection onto the one dimensional subspace spanned by $\boldsymbol{u}$

Minimize Reconstruction Error with $k = 1$

Find $\boldsymbol{u} \in \mathbb{R}^d$ to minimize:

$$\frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{u}\boldsymbol{u}^\top \boldsymbol{x}_i\|_2^2$$



Minimize Reconstruction Error with $k = 1$

$$\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \boldsymbol{u}\boldsymbol{u}^\top \boldsymbol{x}_i\|_2^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2 - \boldsymbol{u}^\top \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^\top\right)\boldsymbol{u}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i\|_2^2 - \boldsymbol{u}^\top \left(\frac{1}{n}\boldsymbol{X}^\top\boldsymbol{X}\right)\boldsymbol{u}$$

Minimize Reconstruction Error with $k = 1$

$$\arg \min_{\boldsymbol{u} \in \mathbb{R}^d : \|\boldsymbol{u}\|_2 = 1} \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{u}\boldsymbol{u}^\top \boldsymbol{x}_i\|_2^2$$

$$= \arg \min_{\boldsymbol{u} \in \mathbb{R}^d : \|\boldsymbol{u}\|_2 = 1} \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i\|_2^2 - \boldsymbol{u}^\top \left(\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \right) \boldsymbol{u}$$

$$\arg \max_{\boldsymbol{u} \in \mathbb{R}^d : \|\boldsymbol{u}\|_2 = 1} \boldsymbol{u}^\top \left(\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \right) \boldsymbol{u}$$

Same optimization objective as View I (variance maximization)!

PCA solution ($k = 1$)

Objective ($k = 1$): both Variance Maximizer and Residual Squared Error Minimizer both resulted in:

$$\arg \max_{\boldsymbol{u} \in \mathbb{R}^d : \|\boldsymbol{u}\|_2 = 1} \boldsymbol{u}^\top \left(\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \right) \boldsymbol{u}$$

Solution: The leading eigenvector of $\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X}$ corresponding to largest eigenvalue!

Projection with General k

Input: raw feature vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ target dimensionality k

Goal: k- dimensional subspace represented by an orthonormal basis

$$\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k \in \mathbb{R}^d$$

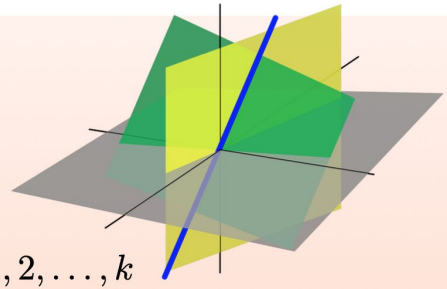
Projection with General k

Input: raw feature vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ target dimensionality k

Goal: k- dimensional subspace represented by an orthonormal basis

$$\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k \in \mathbb{R}^d$$

Recall. The set of vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k\} \in \mathbb{R}^d$ forms an **orthonormal** basis for **sub**space $\mathbb{R}^k$ if:



- $\text{span}\,(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k) = \mathbb{R}^k$
- Each vector has **unit** length: $\|\boldsymbol{u}_i\|_2 = 1, i = 1, 2, \ldots, k$
- They are **pairwise orthogonal**: $\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = \boldsymbol{u}_i^\top \boldsymbol{u}_j = 0, i \neq j$

  or

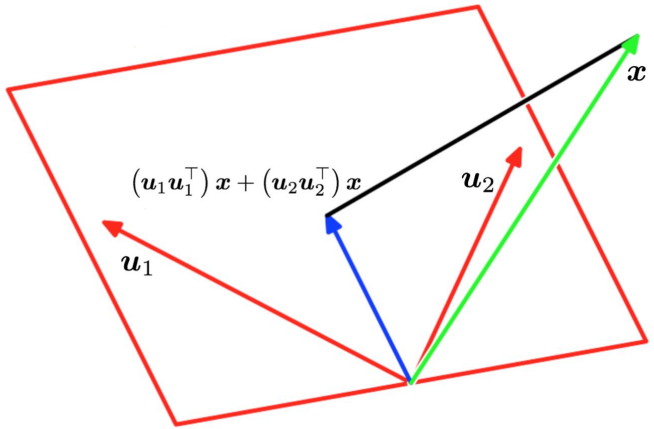
  $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$, where $\boldsymbol{U} = [\boldsymbol{u}_1 | \boldsymbol{u}_2 | \ldots | \boldsymbol{u}_k] \in \mathbb{R}^{d \times k}$

Note: every vector $\boldsymbol{x}$ can be written as: $\boldsymbol{x} = \sum_{i=1}^{k} \alpha_i \boldsymbol{u}_i$, where $\alpha_i = \langle \boldsymbol{x}, \boldsymbol{u}_i \rangle$

Projection with General k

Projection of a vector $\boldsymbol{x} \in \mathbb{R}^d$ to $\text{span}(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$ is computed by

$$\sum_{i=1}^{k} \langle \boldsymbol{u}_i, \boldsymbol{x} \rangle \, \boldsymbol{u}_i \in \mathbb{R}^d$$



Projection with General k

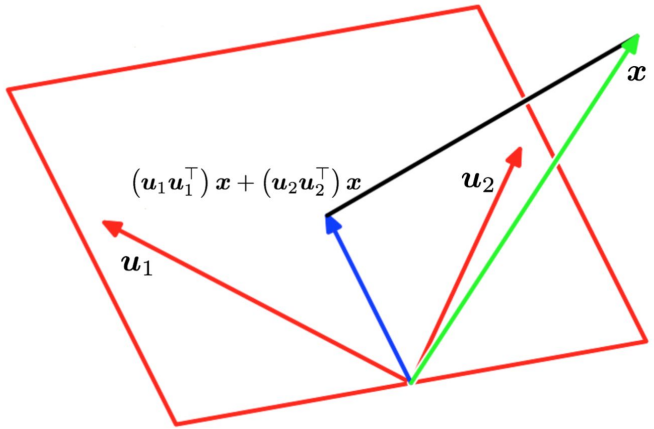
Projection of a vector $\boldsymbol{x} \in \mathbb{R}^d$ to $\text{span}(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$ is computed by

$$\left(\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\top \right) \boldsymbol{x} = \sum_{i=1}^{k} \langle \boldsymbol{u}_i, \boldsymbol{x} \rangle \, \boldsymbol{u}_i \in \mathbb{R}^d$$

Define $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k] \in \mathbb{R}^{d \times k}$ then,

Projection can be written as:

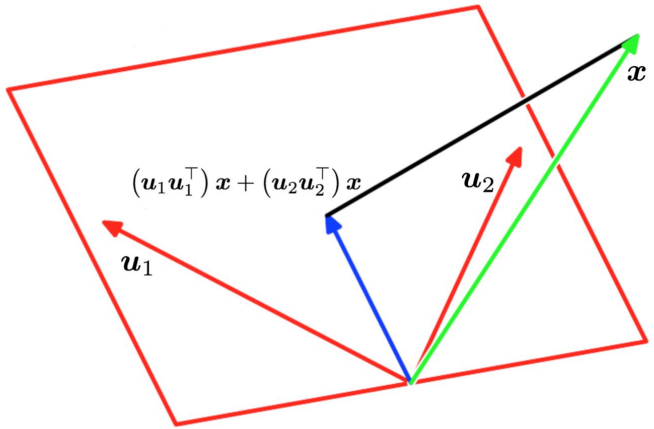
$$\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}$$



PCA with General k

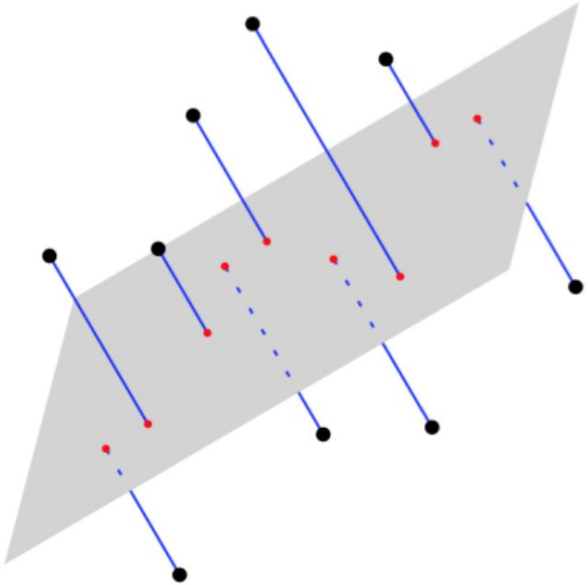
We can simplify the input in projected space as:

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} \langle \boldsymbol{u}_1, \boldsymbol{x} \rangle \\ \langle \boldsymbol{u}_2, \boldsymbol{x} \rangle \\ \vdots \\ \langle \boldsymbol{u}_k, \boldsymbol{x} \rangle \end{bmatrix} \in \mathbb{R}^k$$



View II: Minimizing Reconstruction Error

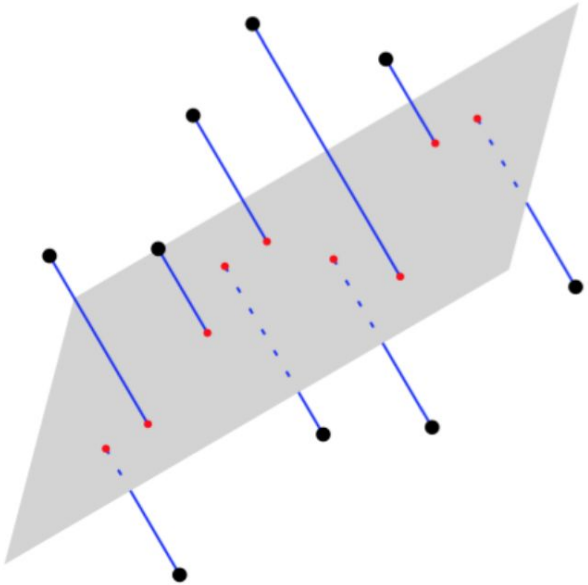
Another way of thinking is: which orthonormal basis to choose to project data such that the reconstruction error is minimized.



That is pick the orthonormal basis such that the residual distance from projected points is minimized!

View II: Minimizing Reconstruction Error

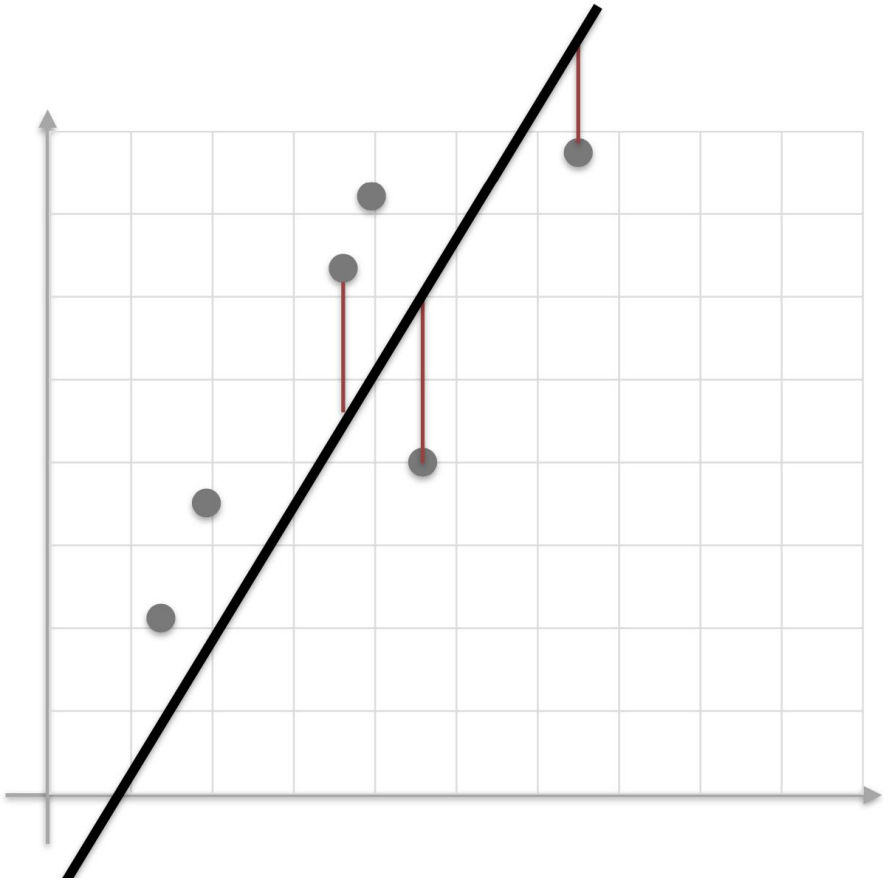
Another way of thinking is: which orthonormal basis to choose to project data such that the reconstruction error is minimized.



That is pick the orthonormal basis such that the residual distance from projected points is minimized!

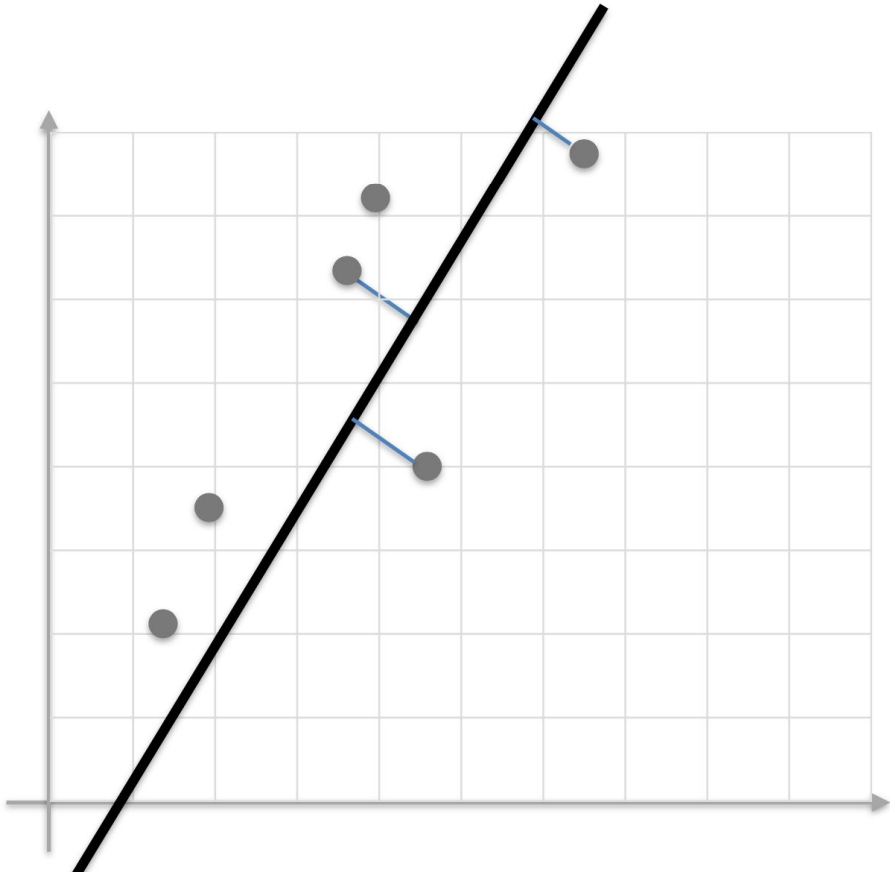
Wait! Is not this just the OLS regression?

PCA versus regression



In regression, we find a line that minimizes prediction error!

PCA versus regression



In PCA (unsupervised dimensionality reduction), we find a line that minimizes the perpendicular error (orthogonal distance)!

Minimizing Reconstruction Error with General k

We can generalize the idea to k dimensional subspaces by solving the following optimization problem to minimize reconstruction error:

$$\arg \min_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \frac{1}{n} \sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i \right\|_2^2$$

$$\arg \min_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \frac{1}{n} \sum_{i=1}^{n} \| \boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i \|_2^2$$

$$\begin{aligned}
||\boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i||_2^2 &= (\boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i)^\top (\boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i) \\
&= \boldsymbol{x}_i^\top \boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i + \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i \\
&= \boldsymbol{x}_i^\top \boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i + \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i \\
&= \boldsymbol{x}_i^\top \boldsymbol{x}_i - \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i
\end{aligned}$$

$$\arg \max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i$$

$$\arg \max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \frac{1}{n} \sum_{i=1}^{n} \mathrm{trace}(\boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i)$$

$$\arg \max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \frac{1}{n} \sum_{i=1}^{n} \mathrm{trace}(\boldsymbol{U}^\top \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{U})$$

$$\arg \max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \mathrm{trace}(\boldsymbol{U}^\top \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{U})$$

$$\arg \max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \mathrm{trace}(\boldsymbol{U}^\top (\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X}) \boldsymbol{U})$$

Maximizing Total Variance with General k

We can generalize the idea to k dimensional subspaces by solving the following optimization problem to minimize reconstruction error:

$$\arg \min_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \frac{1}{n} \sum_{i=1}^{n} \| \boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i \|_2^2$$

This is equivalent to this optimization problem

$$\arg \max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \operatorname{trace}(\boldsymbol{U}^\top (\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X}) \boldsymbol{U})$$

Maximizing Total Variance with General k

We can generalize the idea to k dimensional subspaces by solving the following optimization problem to minimize reconstruction error:

$$\arg \min_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \frac{1}{n} \sum_{i=1}^{n} \| \boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i \|_2^2$$

This is equivalent to this optimization problem

$$\arg \max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \operatorname{trace}(\boldsymbol{U}^\top (\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X}) \boldsymbol{U})$$

in fact, this is **maximizing the total variance:**

$$\sum_{i=1}^{k} \boldsymbol{u}_i^\top (\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X}) \boldsymbol{u}_i = \operatorname{trace}(\boldsymbol{U}^\top (\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X}) \boldsymbol{U})$$

PCA solution ($k = 1$)

Objective ($k = 1$): both Variance Maximizer and Reconstruction Error Minimizer both resulted in:

$$\arg \max_{\boldsymbol{u} \in \mathbb{R}^d : \|\boldsymbol{u}\|_2 = 1} \boldsymbol{u}^\top \left(\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} \right) \boldsymbol{u}$$

Solution: The leading eigenvector of $\frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X}$ corresponding to largest eigenvalue!

Variational Characterization

For a real symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, we can characterize the leading eigenvector the solution to the following optimization problem:

$$\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A} \boldsymbol{u}$$

$$\text{such that} \quad \|\boldsymbol{u}\|_2 = 1$$

Maximum value: $\lambda_1 \quad (\text{top eigen-value})$

Maximizer: $\boldsymbol{u}_1 \quad (\text{top eigen-vector})$

A bit more linear algebra: Generalized Rayleigh Quotient

For a real symmetric, positive semidefinite matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ with eigenvalue decomposition

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

Then a solution for

$$\max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \operatorname{trace}(\boldsymbol{U}^\top \boldsymbol{A} \boldsymbol{U})$$

is the top k leading eigenvectors of $\boldsymbol{A}$, and the maximum value is the sum of top k leading eigenvalues:

Maximizer: $\boldsymbol{U} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]$

Maximum value: $\sum_{i=1}^{k} \lambda_i$

Generalized Rayleigh Quotient - Proof

For $\boldsymbol{U} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]$, we have $\operatorname{trace}(\boldsymbol{U}^\top \boldsymbol{A}\boldsymbol{U}) = \operatorname{trace}(\boldsymbol{U}^\top \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top \boldsymbol{U}) = \sum_{i=1}^{k} \lambda_i$

In fact, the value $\sum_{i=1}^{k} \lambda_i$ is the upper bound for $\max_{\boldsymbol{U} \in \mathbb{R}^{d \times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}} \operatorname{trace}(\boldsymbol{U}^\top \boldsymbol{A}\boldsymbol{U})$

Generalized Rayleigh Quotient - Proof

For any $\boldsymbol{U} \in \mathbb{R}^{d\times k} : \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$, denote $\boldsymbol{B} = \boldsymbol{V}^\top \boldsymbol{U}$ and $\beta_i = \sum_{j=1}^{k} (\boldsymbol{B}_{ij})^2$
First,

$$\boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{U}^\top \boldsymbol{V} \boldsymbol{V}^\top \boldsymbol{U} = \boldsymbol{I} \in \mathbb{R}^{k\times k}$$

Therefore,

$$\sum_{i=1}^{d} \beta_i = \sum_{i=1}^{d} \left(\sum_{j=1}^{k} (\boldsymbol{B}_{ij})^2 \right) = \operatorname{trace}(\boldsymbol{B}\boldsymbol{B}^\top) = \operatorname{trace}(\boldsymbol{B}^\top \boldsymbol{B}) = \operatorname{trace}(\boldsymbol{I}) = k$$

Second, we claim that

$$\forall i = 1, \ldots, d \quad 0 \le \beta_i = \sum_{j=1}^{k} (\boldsymbol{B}_{ij})^2 \le 1$$

To prove this, we can construct a matrix $\tilde{\boldsymbol{B}} \in \mathbb{R}^{d\times d}$ by augmenting $\boldsymbol{B} \in \mathbb{R}^{d\times k}$ such that the first k columns of $\tilde{\boldsymbol{B}}$ is $\boldsymbol{B}$, and we can make $\tilde{\boldsymbol{B}}$ an orthogonal matrix, i.e., $\tilde{\boldsymbol{B}}^\top \tilde{\boldsymbol{B}} = \tilde{\boldsymbol{B}} \tilde{\boldsymbol{B}}^\top = \boldsymbol{I}$. Since $\tilde{\boldsymbol{B}} \tilde{\boldsymbol{B}}^\top = \boldsymbol{I}$, $\sum_{j=1}^{d} (\tilde{\boldsymbol{B}}_{ij})^2 = 1$, which implies $\sum_{j=1}^{k} (\boldsymbol{B}_{ij})^2 \le 1$.

For the objective,

$$\begin{aligned} \operatorname{trace}(\boldsymbol{U}^\top \boldsymbol{A} \boldsymbol{U}) &= \operatorname{trace}(\boldsymbol{U}^\top \boldsymbol{V} \boldsymbol{\Sigma} \boldsymbol{V}^\top \boldsymbol{U}) = \operatorname{trace}(\boldsymbol{B}^\top \boldsymbol{\Sigma} \boldsymbol{B}) = \operatorname{trace}(\boldsymbol{\Sigma} \boldsymbol{B} \boldsymbol{B}^\top) \\ &= \sum_{i=1}^{d} \left(\sum_{j=1}^{k} (\boldsymbol{B}_{ij})^2 \right) \lambda_i = \sum_{i=1}^{d} \beta_i \lambda_i \end{aligned}$$

where $0 \le \beta_i \le 1$, $\sum_{i=1}^{d} \beta_i = k$. Therefore, the upper bound is achieved when $\beta_1 = \cdots = \beta_k = 1$ and it is equal to $\sum_{i=1}^{k} \lambda_i$

PCA algorithm with General k

Input: data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and target dimensionality k

Compute the top k eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k$ of the sample covariance matrix

$$\boldsymbol{\mu} = (1/n) \sum_{i=1}^{n} \boldsymbol{x}_i \qquad \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top$$

Map each data point to:

$$\Phi(\boldsymbol{x}_i) = (\langle \boldsymbol{u}_1, \boldsymbol{x}_i \rangle, \langle \boldsymbol{u}_2, \boldsymbol{x}_i \rangle, \ldots, \langle \boldsymbol{u}_k, \boldsymbol{x}_i \rangle) \in \mathbb{R}^k$$

Reconstruction:

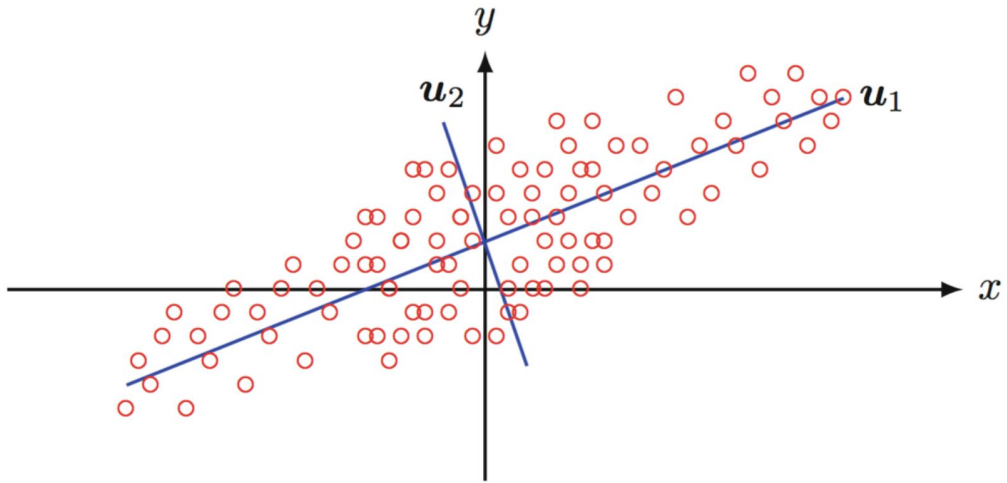
$$\boldsymbol{x}_i \longrightarrow \boldsymbol{\mu} + \boldsymbol{U}\Phi(\boldsymbol{x}_i)$$

Example

$\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^2$ ⇒ centering ⇒ $\frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{2\times 2}$

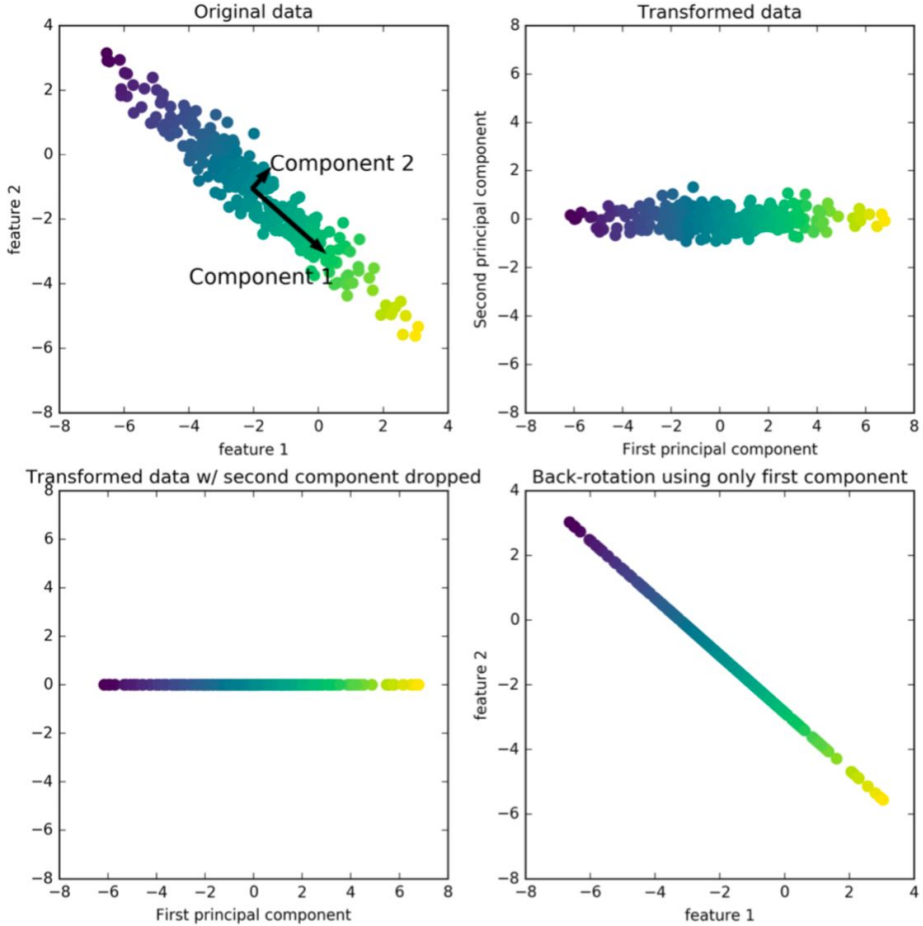
1st Principle Component (eigenvector) of covariance matrix corresponding to largest eigenvalue

2nd Principle Component (eigenvector) of covariance matrix corresponding to 2nd largest eigenvalue



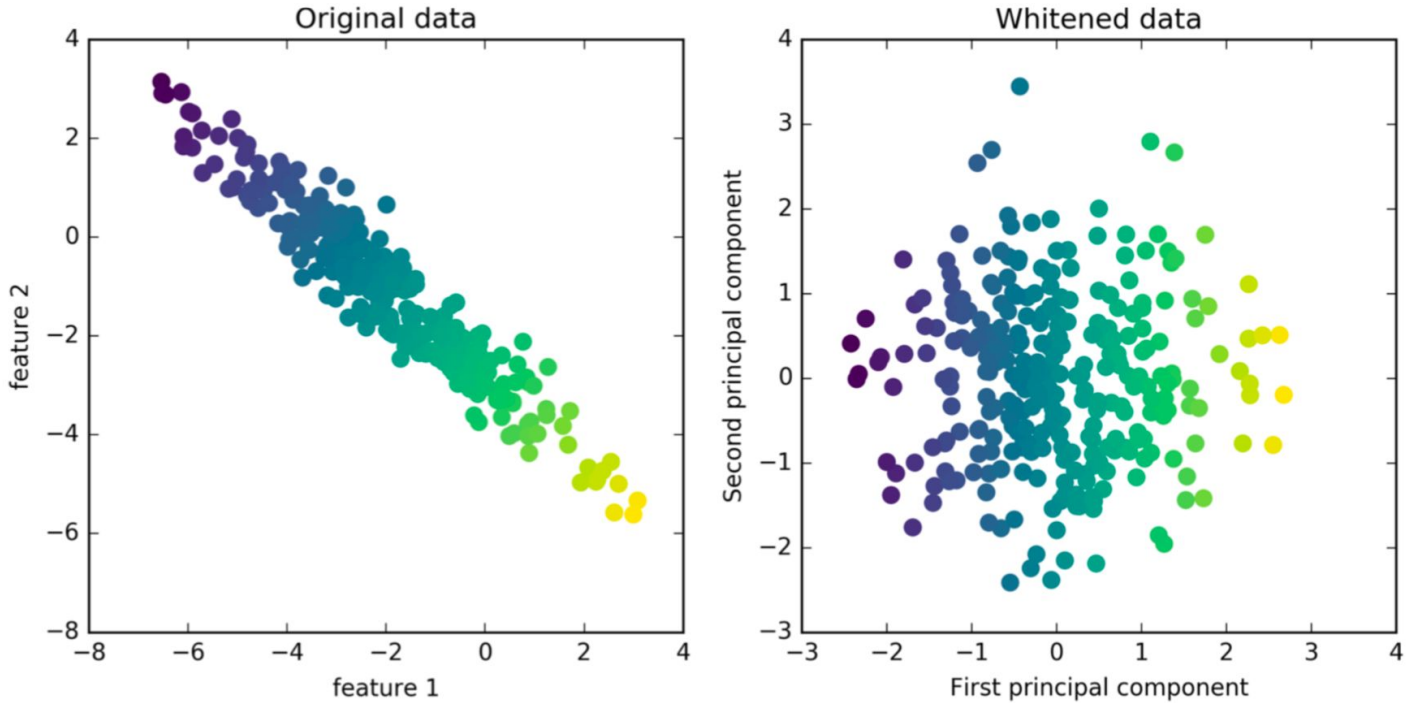
Example showing a two-dimensional data set and its two principal components.

What does it actually do?

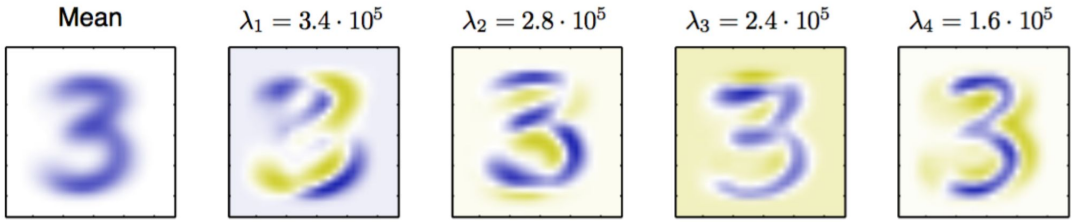


Whitening & PCA

Whitening (also known as sphering) is a linear transformation used for decorrelating signals such that the covariance matrix is identity matrix.

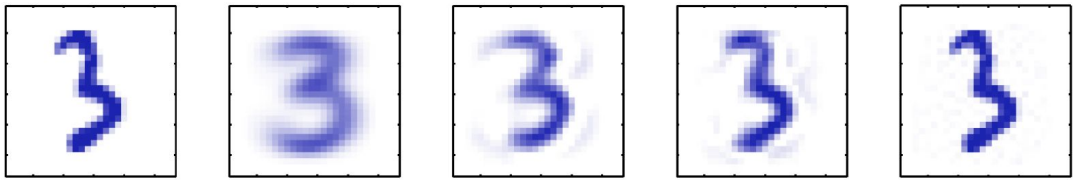


Digits (compression)



16 × 16 pixel images of handwritten 3s (as vectors in R^{256})

Reconstruction:



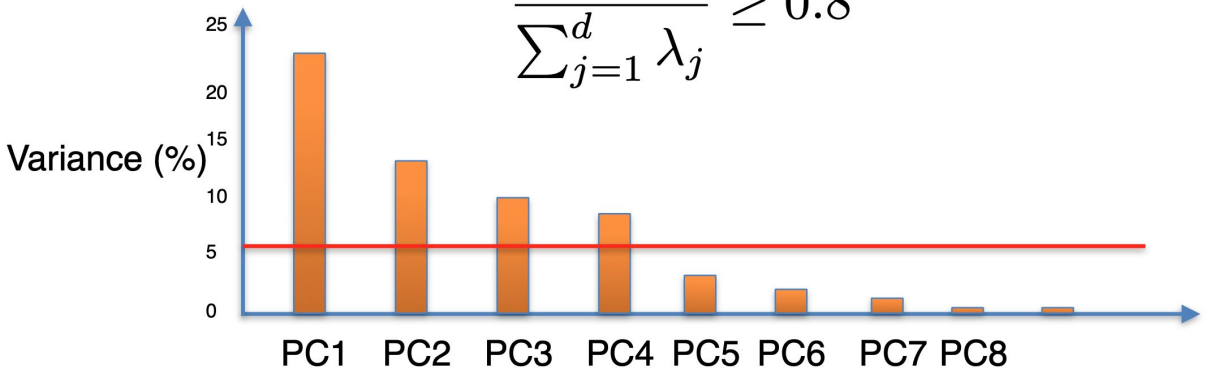
Only have to store k numbers per image, along with the mean and k eigenvectors (256(k + 1) numbers).

How to pick k

- In high-dimensional problems, data sometimes lies near a linear subspace, as noise introduces small variability.
- Only keep data projections onto principal components with large eigenvalues.
- Can ignore the components of smaller significance.

To retain enough components to cover 80% of the total variation, pick k such that:

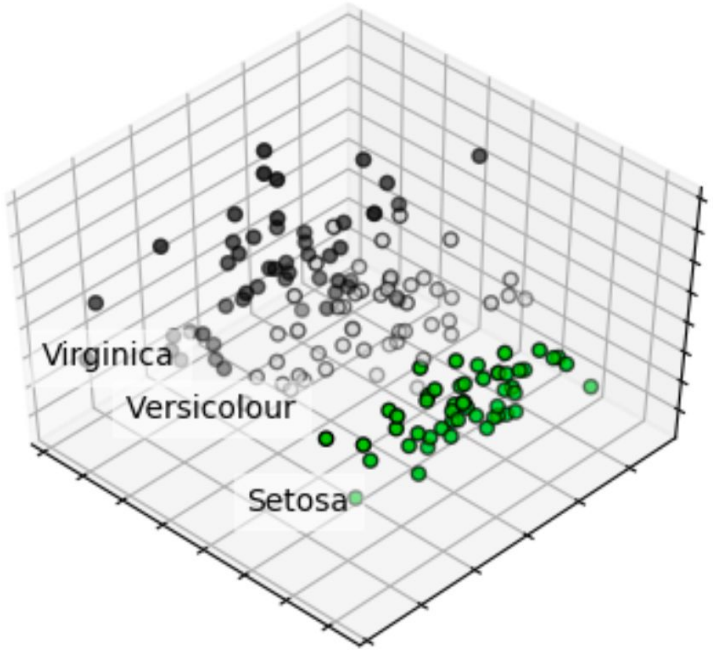
$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j} \geq 0.8$$



Might lose some info, but if eigenvalues are small, do not lose much.

PCA in scikit-learn

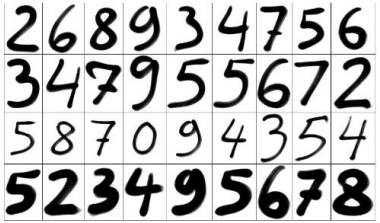
```
from sklearn.decomposition import PCA

pca = PCA(n_components=3)

pca.fit(X)

X = pca.transform(X)
```



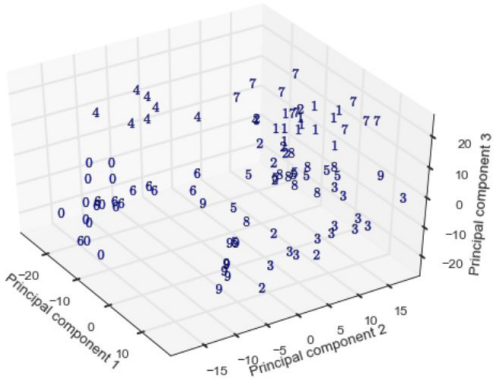
PCA on MNIST digits

```
>>> from sklearn import datasets
>>> from sklearn.decomposition import PCA

# Load the data
>>> digits_data = datasets.load_digits()
>>> n = len(digits_data.images)

# Each image is represented as an 8-by-8 array.
# Flatten this array as input to PCA.
>>> image_data = digits_data.images.reshape((n, -1))
>>> image_data.shape
(1797, 64)

# Groundtruth label of the number appearing in each image
>>> labels = digits_data.target
>>> labels
array([0, 1, 2, ..., 8, 9, 8])

# Fit a PCA transformer to the dataset.
# The number of components is automatically chosen to account for
# at least 80% of the total variance.
>>> pca_transformer = PCA(n_components=0.8)
>>> pca_images = pca_transformer.fit_transform(image_data)
>>> pca_transformer.explained_variance_ratio_
array([ 0.14890594,  0.13618771,  0.11794594,  0.08409979,  0.057
        0.0491691 ,  0.04315987,  0.03661373,  0.03353248,  0.030
        0.02372341,  0.02272697,  0.01821863])
>>> pca_transformer.explained_variance_ratio_[:3].sum()
0.40303958587675121

# Visualize the results
>>> import matplotlib.pyplot as plt
>>> from mpl_toolkits.mplot3d import Axes3D
>>> %matplotlib notebook
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='3d')
>>> for i in range(100):
...     ax.scatter(pca_images[i,0], pca_images[i,1], pca_images[i,
...                marker=r'${}$'.format(labels[i]), s=64)

>>> ax.set_xlabel('Principal component 1')
>>> ax.set_ylabel('Principal component 2')
>>> ax.set_zlabel('Principal component 3')
```

source: https://bit.ly/2G5RoBU

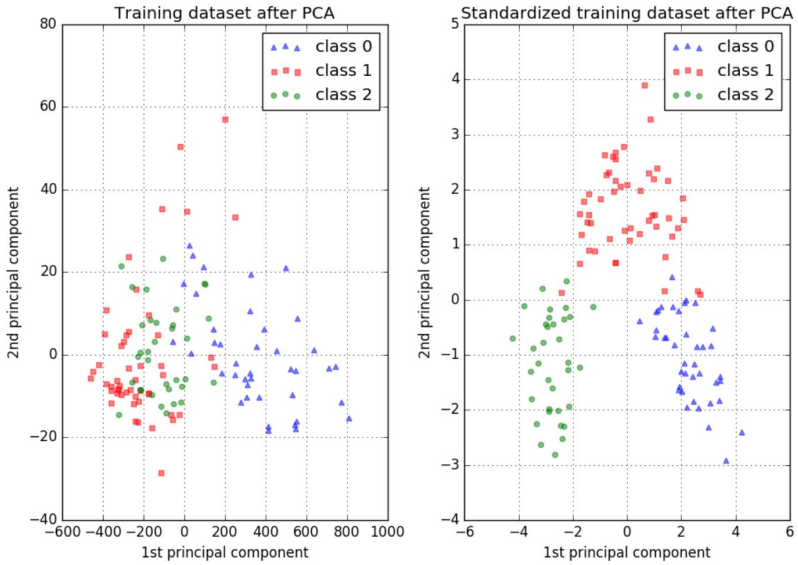


map each image to
top three eigenvectors!



PCA failures

PCA is sensitive to different scaling/normalization of coordinates.



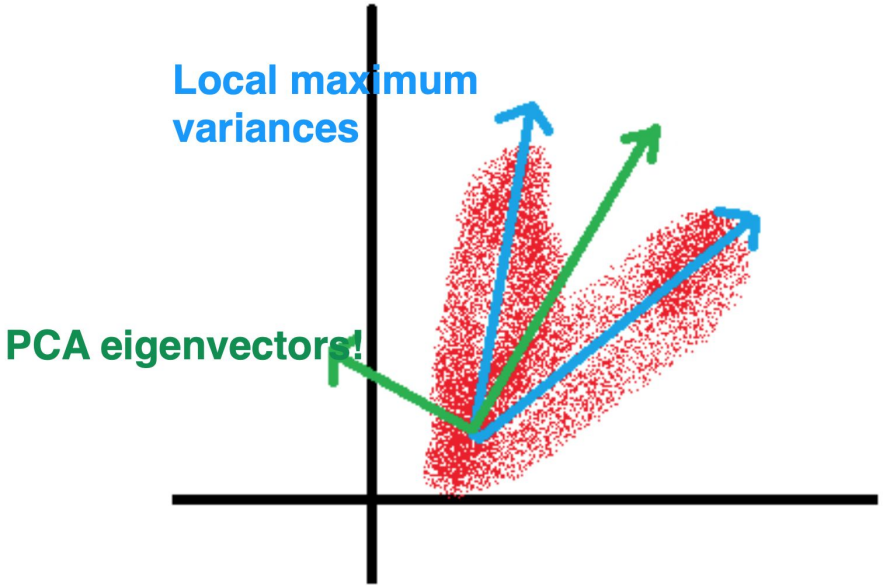
source: https://bit.ly/2GdW88p

An appropriate scaling of features using normalization seems necessary before applying PCA, e.g,

```
from sklearn import preprocessing

std_scale = preprocessing.StandardScaler().fit(X)
X_std = std_scale.transform(X)
```

PCA failures

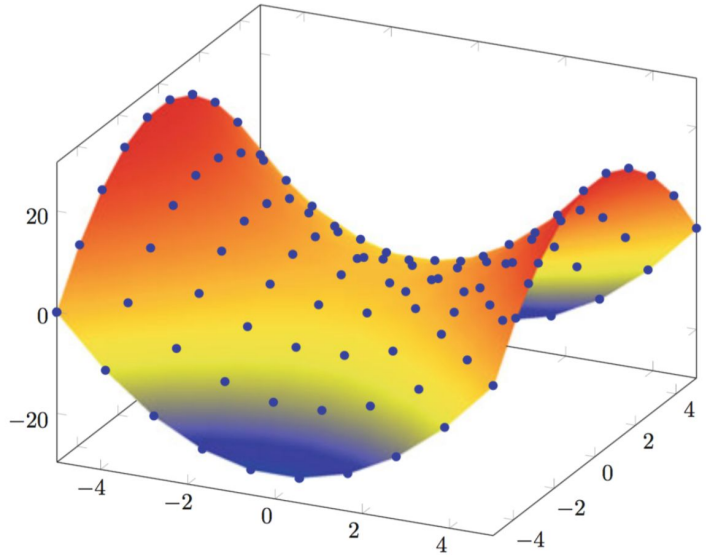
PCA always finds orthogonal principal components. Sometimes, our data demands non-orthogonal principal components to represent the data.



Independent Component Analysis (ICA) might work better!

PCA failures

PCA is all about finding linear structure in data!

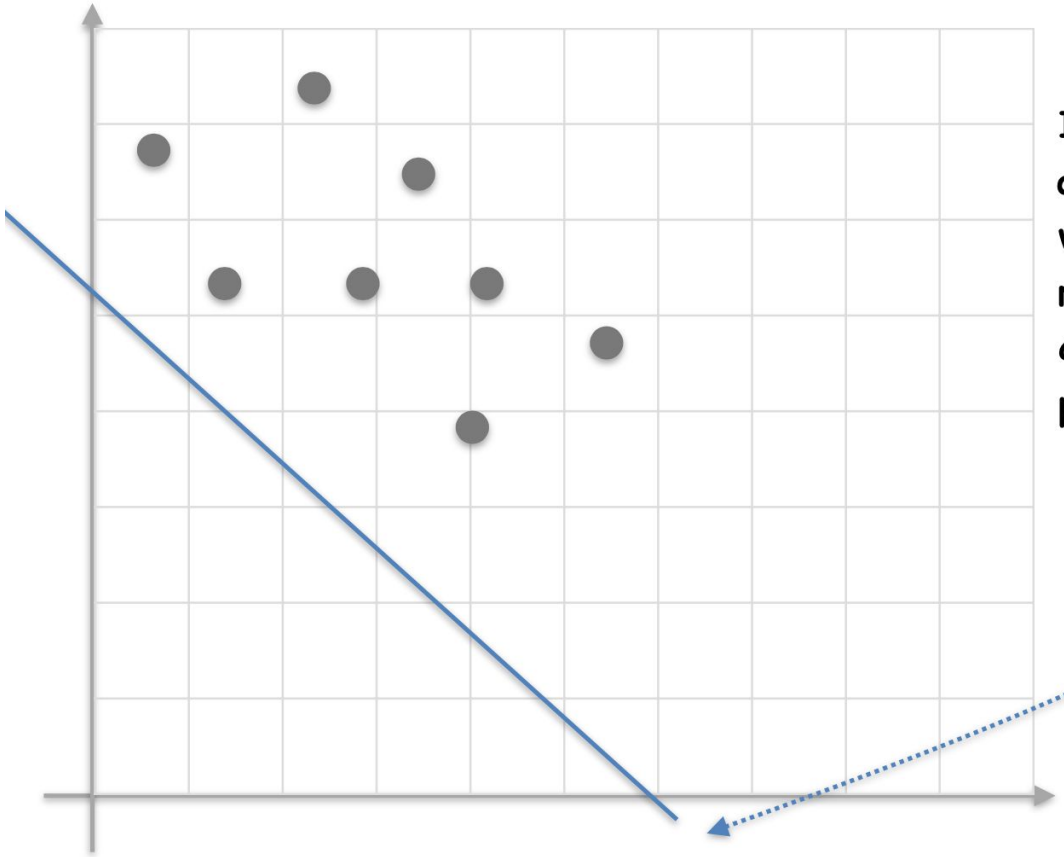


PCA is a linear mapping (each point is a linear combination of original features weighted by the top eigenvectors of covariance matrix). It fails to capture nonlinear structures (manifold)

Sometimes, linear mapping is very limited

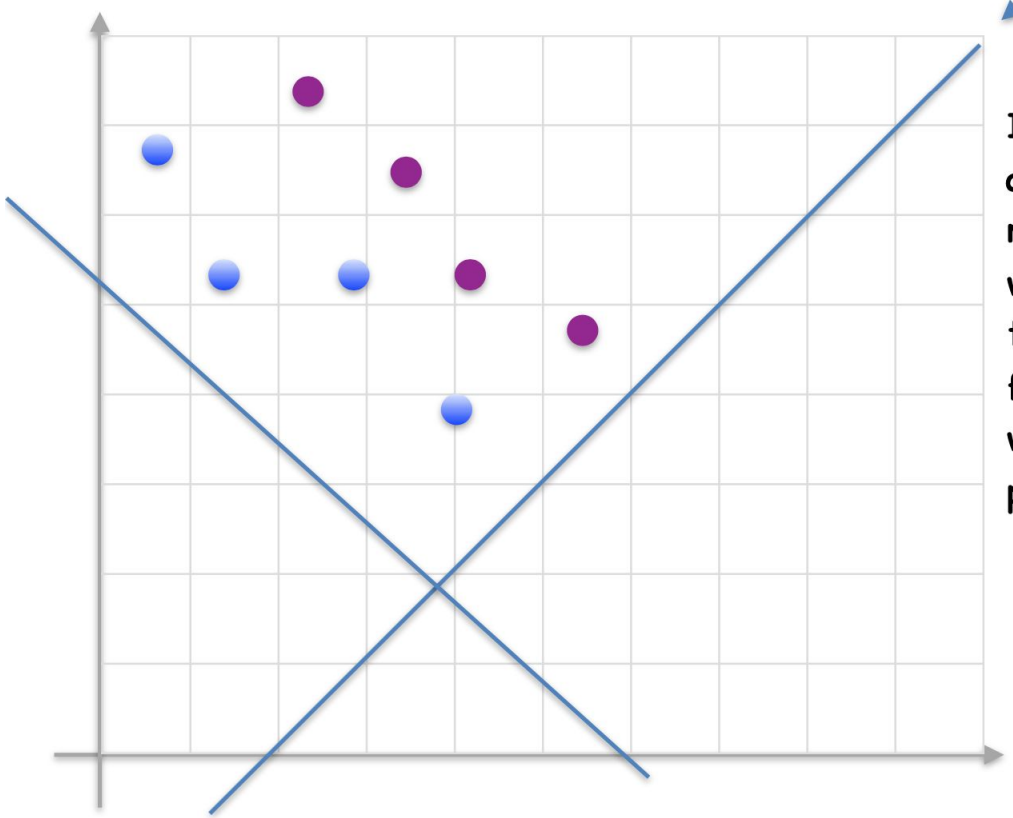
We can use kernelized PCA (manifold learning)

PCA versus supervised dimensionality reduction



In PCA (unsupervised dimensionality reduction), we would like to have maximum spread (variance) among points after projection!

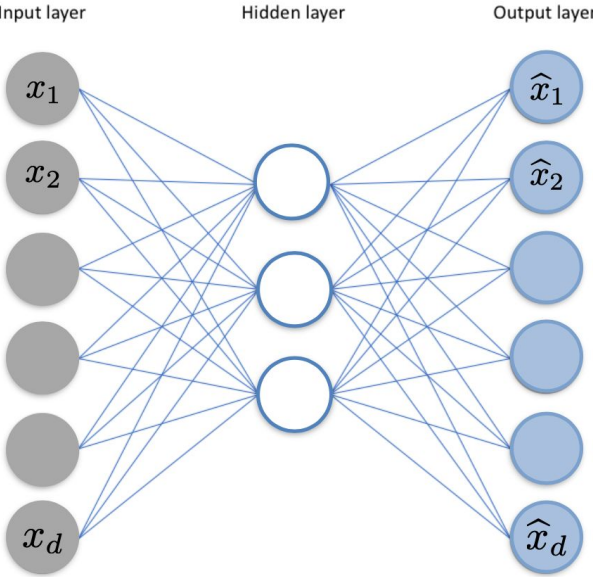
PCA versus supervised dimensionality reduction



In LDA (supervised dimensionality reduction, not covered in this course), we would like to have the samples from different classes are well separated after projection!

Autoencoders

- Representation learning
- An autoencoder is a feed-forward neural net whose job it is to take an input and predict/reconstruct the input
- The hidden layer can be considered as compression of data
- Learn by optimizing reconstruction error: $\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2$



Source: https://www.jeremyjordan.me/autoencoders/

Reducing the Dimensionality of Data with Neural Networks

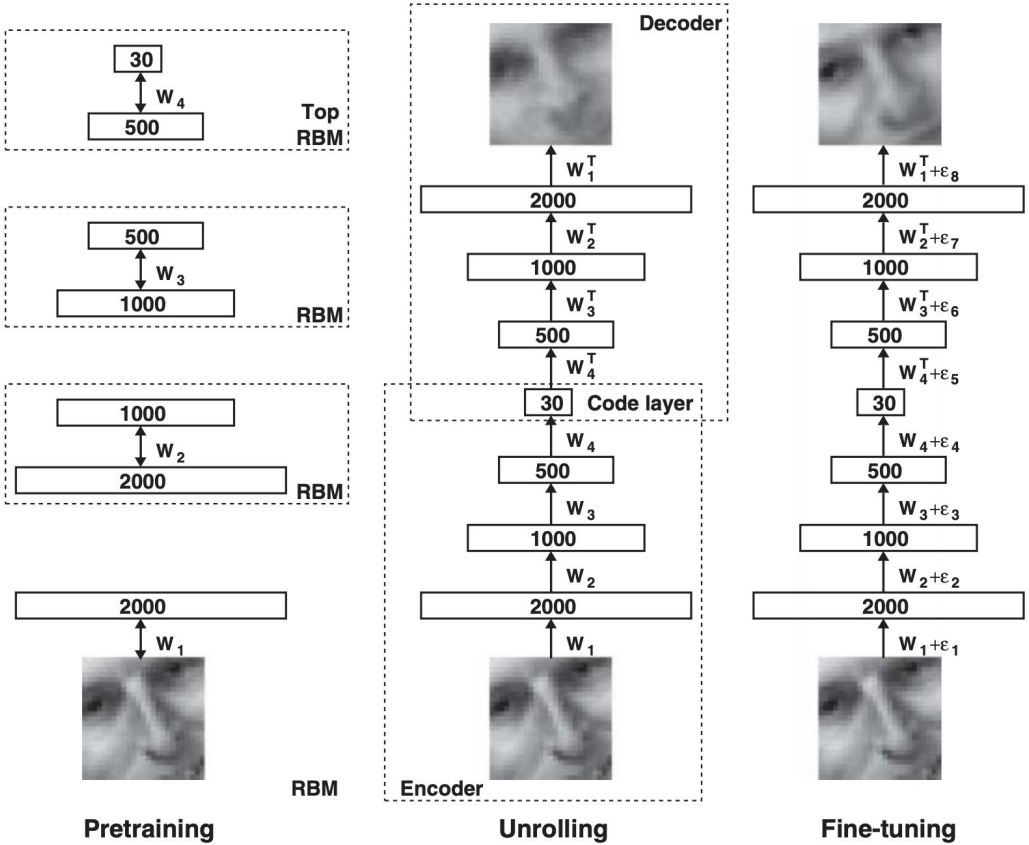
Reducing the Dimensionality of Data with Neural Networks

Science
AAAS

G. E. Hinton* and R. R. Salakhutdinov

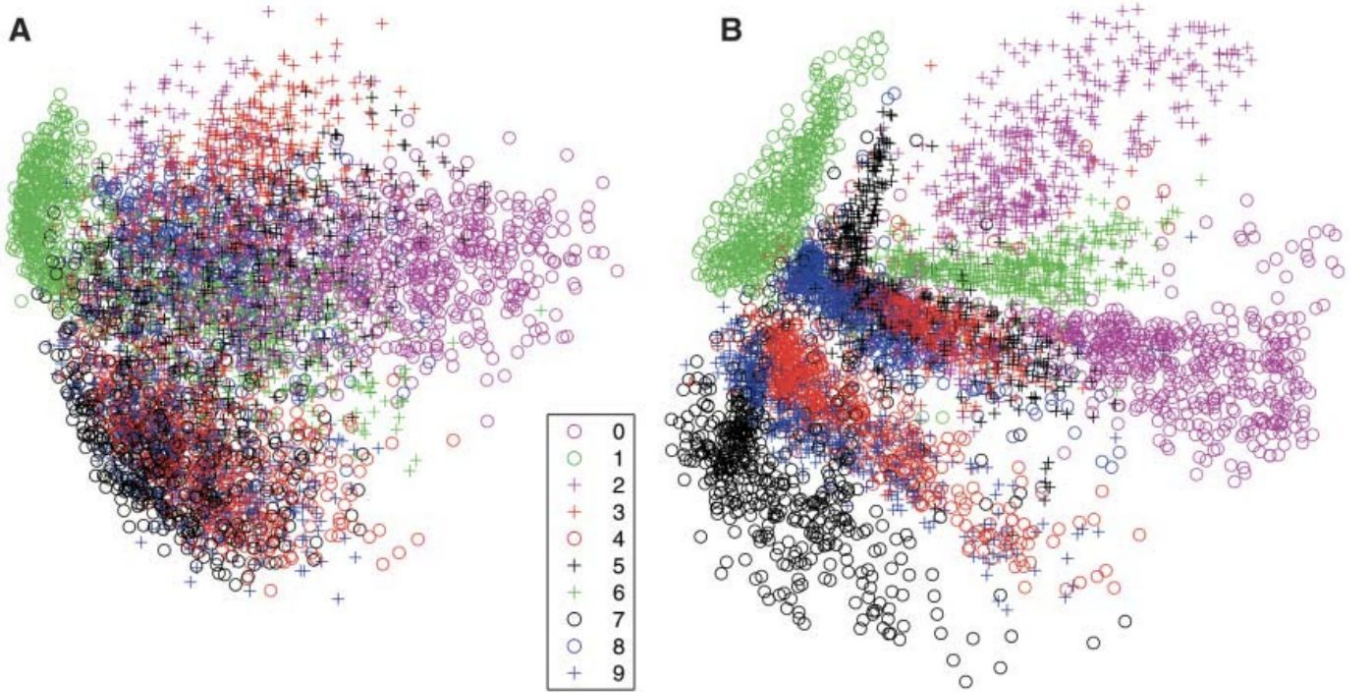
High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Reducing the Dimensionality of Data with Neural Networks



Reducing the Dimensionality of Data with Neural Networks

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



Appendix

Variational Characterization

For a real symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, we can characterize the leading eigenvector the solution to the following optimization problem:

$$\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A} \boldsymbol{u}$$

$$\text{such that} \quad \|\boldsymbol{u}\|_2 = 1$$

Maximum value: $\lambda_1 \quad (\text{top eigen-value})$

Maximizer: $\boldsymbol{u}_1 \quad (\text{top eigen-vector})$

Variational Characterization

For a real symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, we can characterize the leading eigenvector the solution to the following optimization problem:

$$\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A} \boldsymbol{u}$$

$$\text{such that} \quad \|\boldsymbol{u}\|_2 = 1$$

Maximum value: $\lambda_1 \quad (\text{top eigen-value})$

Maximizer: $\boldsymbol{u}_1 \quad (\text{top eigen-vector})$

Solution:

- Use a Lagrange multiplier to enforce $||\boldsymbol{u}||_2^2 = \boldsymbol{u}^\top \boldsymbol{u} = 1$
- Maximizing $\boldsymbol{u}^\top \boldsymbol{A} \boldsymbol{u} + \lambda(1 - \boldsymbol{u}^\top \boldsymbol{u})$
- Derivative is zero when $\boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{u}$, that is $\boldsymbol{u}^\top \boldsymbol{A} \boldsymbol{u} = \lambda$
- So, the maximum value is λ_1, and the maximizer is $\boldsymbol{u}_1$

Power method

How can we decompose a symmetric matrix (i.e., compute its eigenvectors and corresponding eigenvalues)?

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

Power method

How can we decompose a symmetric matrix (i.e., compute its eigenvectors and corresponding eigenvalues)?

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

$$\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u}$$

$$\text{such that} \quad \|\boldsymbol{u}\|_2 = 1$$

This is a non-convex optimization problem (objective is convex, but the constraint set is sphere which is non-convex)

Power method

How can we decompose a symmetric matrix (i.e., compute its eigenvectors and corresponding eigenvalues)?

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

$$\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u}$$
$$\text{such that} \quad \|\boldsymbol{u}\|_2 = 1$$

This is a non-convex optimization problem (objective is convex, but the constraint set is sphere which is non-convex)

Power Method for top eigenvector ($\boldsymbol{A} \in \mathbb{R}^{d \times d}$)

- Randomly initialize $\boldsymbol{u}_0 \in \mathbb{R}^d$
- Repeat t = 1, 2, …, T
  - $\tilde{\boldsymbol{u}}_{t+1} = \boldsymbol{A}\boldsymbol{u}_t$
  - $\boldsymbol{u}_{t+1} = \frac{\tilde{\boldsymbol{u}}_{t+1}}{\|\tilde{\boldsymbol{u}}_{t+1}\|}$
- Return $\boldsymbol{u}_T$

Power method

How can we decompose a symmetric matrix (i.e., compute its eigenvectors and corresponding eigenvalues)?

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

$$\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u}$$
$$\text{such that} \quad \|\boldsymbol{u}\|_2 = 1$$

This is a non-convex optimization problem (objective is convex, but the constraint set is sphere which is non-convex)

Power Method for top eigenvector ($\boldsymbol{A} \in \mathbb{R}^{d \times d}$)

- Randomly initialize $\boldsymbol{u}_0 \in \mathbb{R}^d$
- Repeat t = 1, 2, …, T
  - $\tilde{\boldsymbol{u}}_{t+1} = \boldsymbol{A}\boldsymbol{u}_t$
  - $\boldsymbol{u}_{t+1} = \frac{\tilde{\boldsymbol{u}}_{t+1}}{\|\tilde{\boldsymbol{u}}_{t+1}\|}$
- Return $\boldsymbol{u}_T$

After finding the top eigenvector $\boldsymbol{u}_1$
we can compute the top eigenvalues as

$$\lambda_1 = \boldsymbol{u}_1^\top \boldsymbol{A}\boldsymbol{u}_1$$

Power method

How can we decompose a symmetric matrix (i.e., compute its eigenvectors and corresponding eigenvalues)?

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

$$\max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u}$$
$$\text{such that} \quad \|\boldsymbol{u}\|_2 = 1$$

This is a non-convex optimization problem (objective is convex, but the constraint set is sphere which is non-convex)

Power Method for top eigenvector ($\boldsymbol{A} \in \mathbb{R}^{d \times d}$)

- Randomly initialize $\boldsymbol{u}_0 \in \mathbb{R}^d$
- Repeat t = 1, 2, …, T
  - $\tilde{\boldsymbol{u}}_{t+1} = \boldsymbol{A}\boldsymbol{u}_t$
  - $\boldsymbol{u}_{t+1} = \frac{\tilde{\boldsymbol{u}}_{t+1}}{\|\tilde{\boldsymbol{u}}_{t+1}\|}$
- Return $\boldsymbol{u}_T$

After finding the top eigenvector $\boldsymbol{u}_1$
we can compute the top eigenvalues as

$$\lambda_1 = \boldsymbol{u}_1^\top \boldsymbol{A}\boldsymbol{u}_1$$

Then we can repeatedly find the remaining eigenvectors recursively (how?)
Note that

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$

So apply power method on

$$\boldsymbol{A} - \lambda_1 \boldsymbol{u}_1 \boldsymbol{u}_1^\top$$

Algorithm 1 The Power Method

Choose a random vector $q^{(0)} \in R^n$

for $k = 1, 2, \ldots$ (while $\|q^{(k-1)} - q^{(k-2)}\| > \epsilon$)

$z^{(k)} = Aq^{(k-1)}$

$q^{(k)} = z^{(k)} / \|z^{(k)}\|$

$\lambda^{(k)} = [q^{(k)}]^T A q^{(k)}$

end

Let us examine the convergence properties of the power iteration. If A is diagonalizable (see appendix for a reminder) then there exist n independent eigenvectors of A. Let $x_1, \ldots x_n$ be these eigenvectors, then $x_1, \ldots x_n$ form a basis of R^n. Hence the initial vector $q^{(0)}$ can be written as:

$$q^{(0)} = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \tag{1}$$

where $a_1, \ldots, a_n$ are scalars. multiplying both sides of the equation in A^k yields:

$$A^k q^{(0)} = A^k(a_1 x_1 + a_2 x_2 + \cdots + a_n x_n) = a_1 A^k x_1 + a_2 A^k x_2 + \ldots + a_n A^k x_n \tag{2}$$

$$= a_1 \lambda_1^k x_1 + a_2 \lambda_2^k x_2 + \ldots + a_n \lambda_n^k x_n = a_1 \lambda_1^k \left(x_1 + \sum_{j=2}^{n} \frac{a_j}{a_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k x_j \right)$$

If $|\lambda_1| > |\lambda_2| \geq \ldots |\lambda_n|$ then we say that λ_1 is a dominant eigenvalue. In this case $\left(\frac{\lambda_j}{\lambda_1}\right)^k \to 0$ and therefore if $a_1 \neq 0$, $A^k q^{(0)} \to a_1 \lambda_1^k x_1$. The power method normalizes the products $Aq^{(k-1)}$ to avoid overflow/underflow, therefore it converges to x_1 (assuming it has unit norm).