18-661 Introduction to Machine Learning

Dimensionality Reduction

Fall 2020

ECE – Carnegie Mellon University

Announcements

- HW 6 is due on 11/19.

Announcements

- HW 6 is due on 11/19.
- HW 7 (last homework) will be released tomorrow.

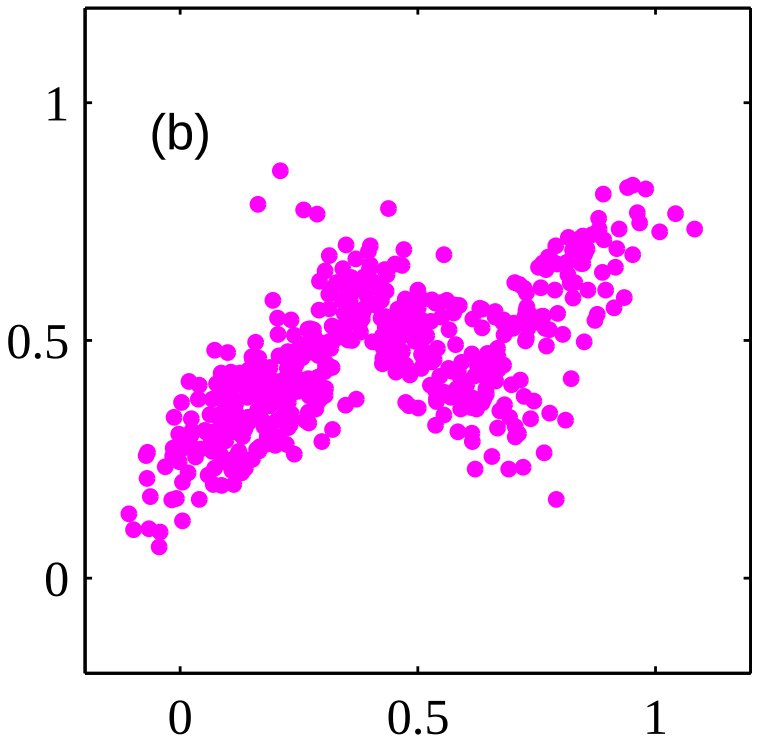
Outline

1. Recap: Gaussian Mixture Models
2. Principal Component Analysis (PCA)
3. Independent Component Analysis (ICA)

Recap: Gaussian Mixture Models

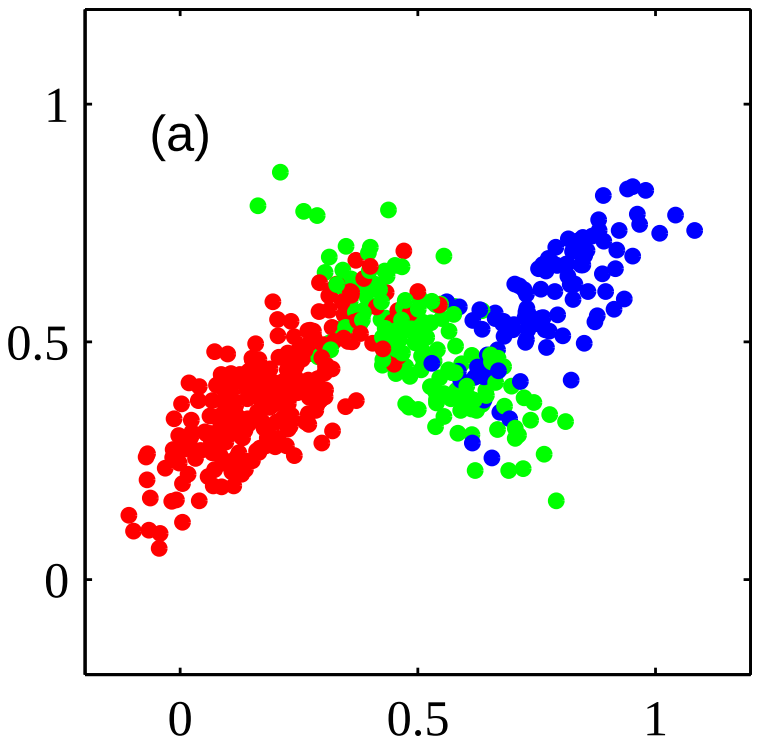
Probabilistic interpretation of clustering?

How can we model $p(\boldsymbol{x})$ to reflect our intuition that points stay close to their cluster centers?



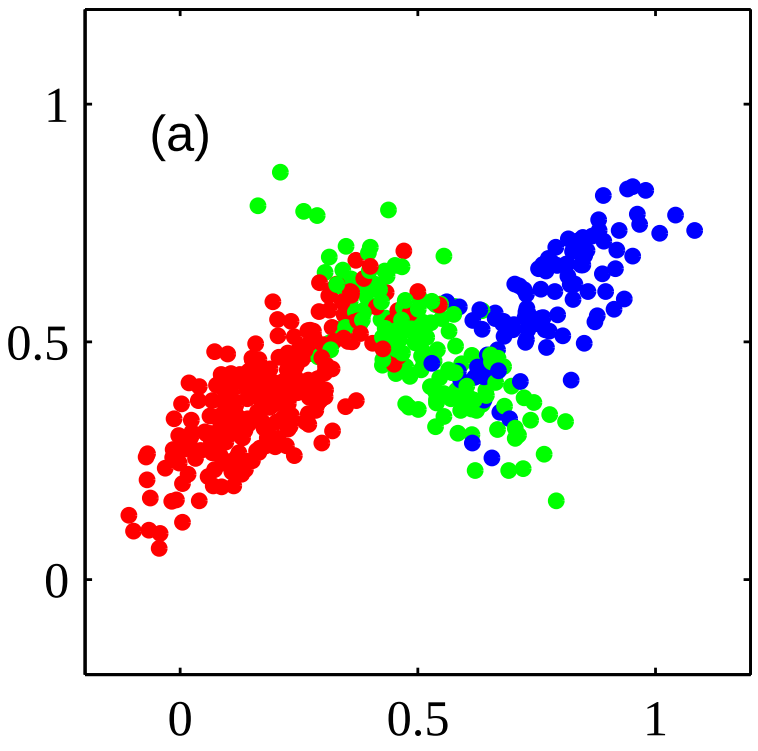
- Points seem to form 3 clusters
- We cannot model $p(\boldsymbol{x})$ with simple and known distributions
- E.g., the data is not a Gaussian b/c we have 3 distinct concentrated regions...

Gaussian mixture models: Intuition



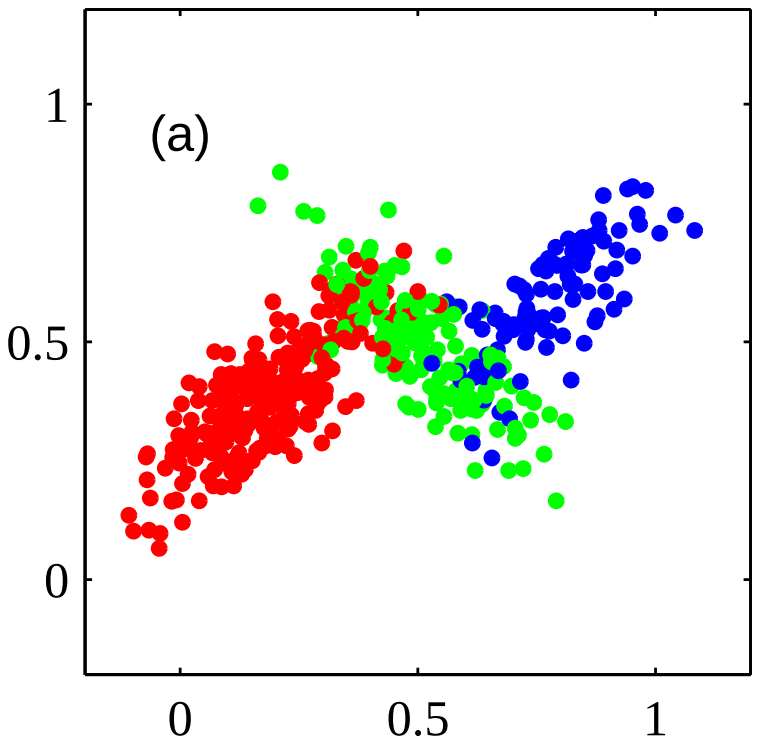
- Key idea: Model *each* region with a distinct distribution

Gaussian mixture models: Intuition



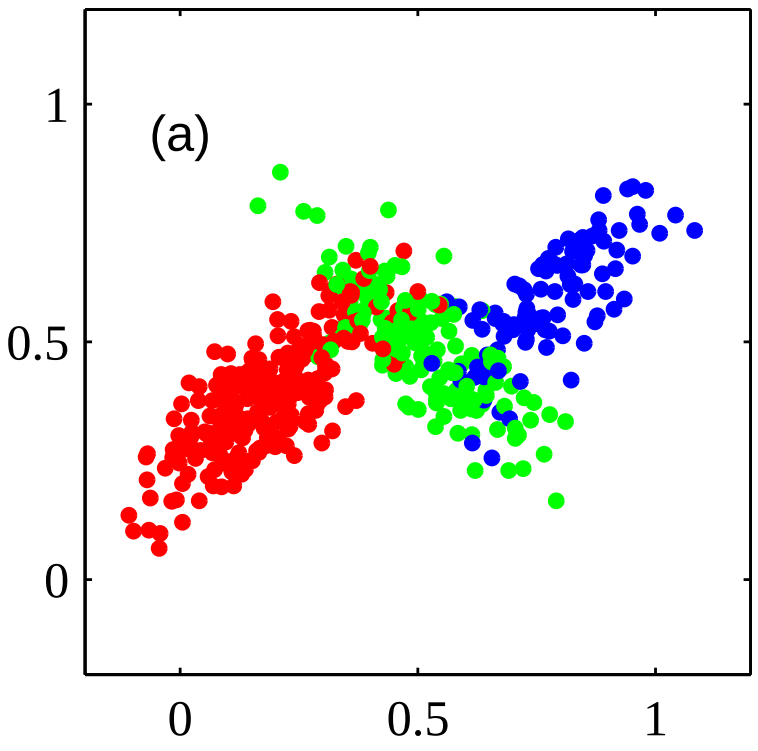
- Key idea: Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)

Gaussian mixture models: Intuition



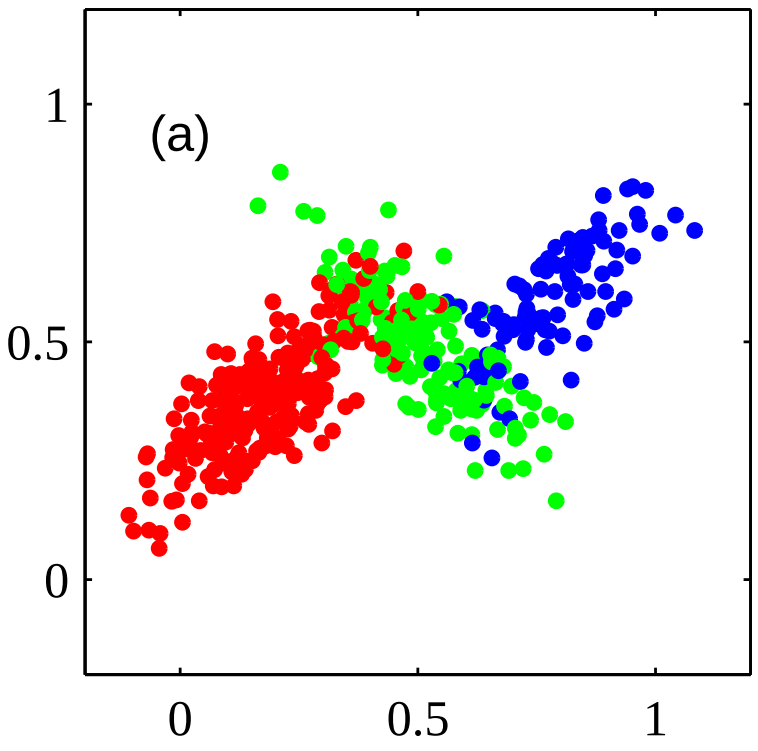
- Key idea: Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)

Gaussian mixture models: Intuition



- Key idea: Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
- *However*, we don't know *cluster assignments* (label), *parameters* of Gaussians, or *mixture components*!

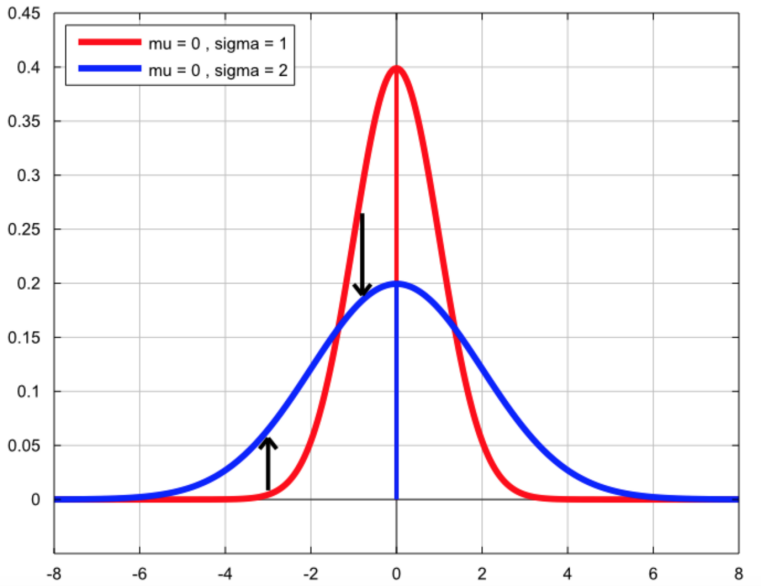
Gaussian mixture models: Intuition



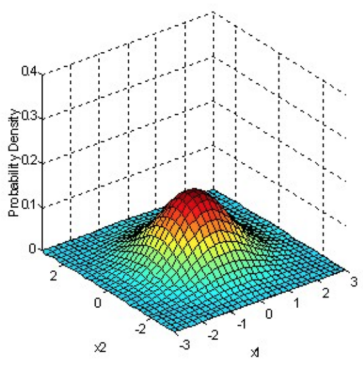
- Key idea: Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
- *However*, we don't know *cluster assignments* (label), *parameters* of Gaussians, or *mixture components*!
- Must learn these values from *unlabeled* data $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$

Recall: Gaussian (normal) distributions

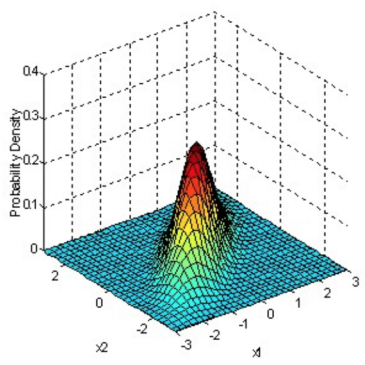
$$\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad f(\boldsymbol{x}) = \frac{\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}}$$



$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$



$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$



Gaussian mixture models: Formal definition

GMM has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K: number of Gaussians — they are called mixture components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of k-th component
- ω_k: mixture weights (or priors) represent how much each component contributes to final distribution. They satisfy 2 properties:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

These properties ensure $p(\boldsymbol{x})$ is in fact a probability density function.

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K.

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K.

Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

$$p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K.

Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

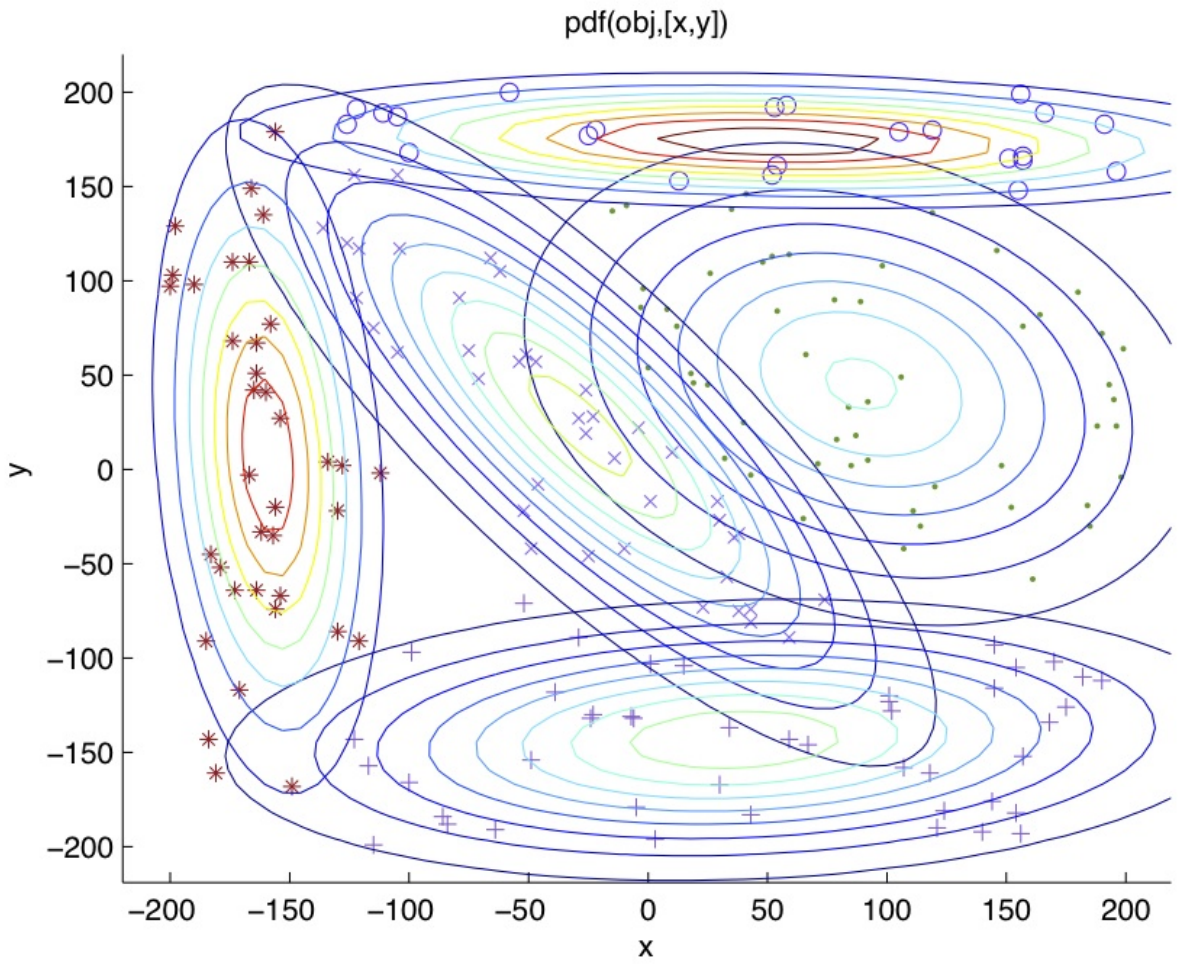
$$p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Then, the marginal distribution of $\boldsymbol{x}$ is

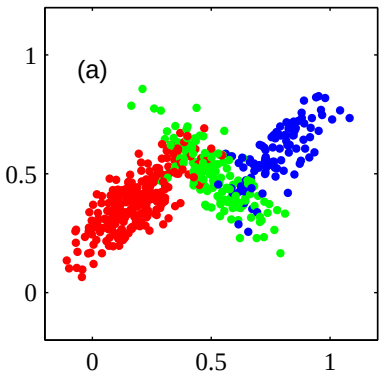
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Namely, the Gaussian mixture model

Gaussian mixture model for clustering



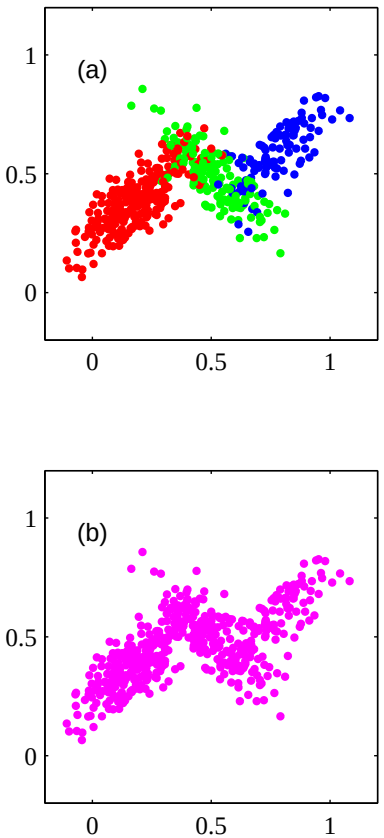
GMMs: Example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

$$p(\boldsymbol{x}|z = red) = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

GMMs: Example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

$$p(\boldsymbol{x}|z = red) = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

The marginal distribution is thus

$$\begin{aligned} p(\boldsymbol{x}) = & \, p(z = red)N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ & + p(z = blue)N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ & + p(z = green)N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3) \end{aligned}$$

Parameter estimation for Gaussian mixture models

The parameters in GMMs are:

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Let's first consider the (unrealistic) case where *we know the labels* z.

Parameter estimation for Gaussian mixture models

The parameters in GMMs are:

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Let's first consider the (unrealistic) case where *we know the labels z*.

Define $\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^{N}$, $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$

- $\mathcal{D}'$ is the complete data
- $\mathcal{D}$ the incomplete data

How can we learn our parameters?

Parameter estimation for Gaussian mixture models

The parameters in GMMs are:

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Let's first consider the (unrealistic) case where *we know the labels z*.

Define $\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^{N}$, $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$

- $\mathcal{D}'$ is the complete data
- $\mathcal{D}$ the incomplete data

How can we learn our parameters?

Given $\mathcal{D}'$, the maximum likelihood estimation of the $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} = \arg\max \sum_n \log p(\boldsymbol{x}_n, z_n)$$

Parameter estimation for GMMs: Complete data

The complete likelihood is decomposable across the labels:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n) p(\boldsymbol{x}_n | z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n) p(\boldsymbol{x}_n | z_n)$$

where we have grouped data by cluster labels z_n.

Parameter estimation for GMMs: Complete data

The complete likelihood is decomposable across the labels:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n) p(\boldsymbol{x}_n | z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n) p(\boldsymbol{x}_n | z_n)$$

where we have grouped data by cluster labels z_n.

Let $r_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

Parameter estimation for GMMs: Complete data

The complete likelihood is decomposable across the labels:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n) p(\boldsymbol{x}_n | z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n) p(\boldsymbol{x}_n | z_n)$$

where we have grouped data by cluster labels z_n.

Let $r_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n r_{nk} \log p(z = k) p(\boldsymbol{x}_n | z = k)$$

Parameter estimation for GMMs: Complete data

The complete likelihood is decomposable across the labels:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n) p(\boldsymbol{x}_n | z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n) p(\boldsymbol{x}_n | z_n)$$

where we have grouped data by cluster labels z_n.

Let $r_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

$$\begin{aligned}\sum_n \log p(\boldsymbol{x}_n, z_n) &= \sum_k \sum_n r_{nk} \log p(z = k) p(\boldsymbol{x}_n | z = k) \\ &= \sum_k \sum_n r_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]\end{aligned}$$

In the complete setting, the r_{nk} just add to the notation, but later we will 'relax' these variables and allow them to take on fractional values.

Parameter estimation for GMMs: Complete data

The MLE is:

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_{k'} \sum_n r_{nk'}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

Parameter estimation for GMMs: Complete data

The MLE is:

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_{k'} \sum_n r_{nk'}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

Since r_{nk} is binary, the previous solution is nothing but:

- ω_k: fraction of total data points whose cluster label z_n is k

Parameter estimation for GMMs: Complete data

The MLE is:

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_{k'} \sum_n r_{nk'}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

Since r_{nk} is binary, the previous solution is nothing but:

- ω_k: fraction of total data points whose cluster label z_n is k
  - note that $\sum_{k'} \sum_n r_{nk'} = N$

Parameter estimation for GMMs: Complete data

The MLE is:

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_{k'} \sum_n r_{nk'}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

Since r_{nk} is binary, the previous solution is nothing but:

- ω_k: fraction of total data points whose cluster label z_n is k
  - note that $\sum_{k'} \sum_n r_{nk'} = N$
- $\boldsymbol{\mu}_k$: mean of all data points whose label z_n is k

Parameter estimation for GMMs: Complete data

The MLE is:

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_{k'} \sum_n r_{nk'}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

Since r_{nk} is binary, the previous solution is nothing but:

- ω_k: fraction of total data points whose cluster label z_n is k
  - note that $\sum_{k'} \sum_n r_{nk'} = N$
- $\boldsymbol{\mu}_k$: mean of all data points whose label z_n is k
- $\boldsymbol{\Sigma}_k$: covariance of all data points whose label z_n is k

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete:

- Observed: $\mathcal{D} = \{\boldsymbol{x}_n\}$
- Unobserved (hidden): $\{\boldsymbol{z}_n\}$

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete:

- Observed: $\mathcal{D} = \{\boldsymbol{x}_n\}$
- Unobserved (hidden): $\{\boldsymbol{z}_n\}$

Goal: Obtain the maximum likelihood estimate of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = \arg\max \log p(\mathcal{D}) = \arg\max \sum_n \log p(\boldsymbol{x}_n)$$

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$$

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete:

- Observed: $\mathcal{D} = \{\boldsymbol{x}_n\}$
- Unobserved (hidden): $\{\boldsymbol{z}_n\}$

Goal: Obtain the maximum likelihood estimate of $\boldsymbol{\theta}$:

$$\begin{aligned}\boldsymbol{\theta} &= \arg\max \log p(\mathcal{D}) = \arg\max \sum_n \log p(\boldsymbol{x}_n) \\ &= \arg\max \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n)\end{aligned}$$

The objective function is called the *incomplete* log-likelihood.

Parameter estimation for GMMs: Incomplete data

No simple way to optimize the incomplete log-likelihood...

Parameter estimation for GMMs: Incomplete data

No simple way to optimize the incomplete log-likelihood...

EM algorithm provides a strategy for iteratively optimizing this function!

Parameter estimation for GMMs: Incomplete data

No simple way to optimize the incomplete log-likelihood...

EM algorithm provides a strategy for iteratively optimizing this function!

E-Step: 'Guess' values of the z_n using existing values of $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. How do we do this?

Parameter estimation for GMMs: Incomplete data

No simple way to optimize the incomplete log-likelihood...

EM algorithm provides a strategy for iteratively optimizing this function!

E-Step: 'Guess' values of the z_n using existing values of $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. How do we do this?

When z_n is not given, we can guess it via the posterior probability (recall: Bayes' rule!)

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)}$$

Parameter estimation for GMMs: Incomplete data

No simple way to optimize the incomplete log-likelihood...

EM algorithm provides a strategy for iteratively optimizing this function!

E-Step: 'Guess' values of the z_n using existing values of $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. How do we do this?

When z_n is not given, we can guess it via the posterior probability (recall: Bayes' rule!)

$$
\begin{aligned}
p(z_n = k|\boldsymbol{x}_n) &= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} \\
&= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^K p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}
\end{aligned}
$$

Parameter estimation for GMMs: Incomplete data

No simple way to optimize the incomplete log-likelihood...

EM algorithm provides a strategy for iteratively optimizing this function!

E-Step: 'Guess' values of the z_n using existing values of $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. How do we do this?

When z_n is not given, we can guess it via the posterior probability (recall: Bayes' rule!)

$$
\begin{aligned}
p(z_n = k|\boldsymbol{x}_n) &= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} \\
&= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^K p(\boldsymbol{x}_n|z_n = k')p(z_n = k')} \\
&= \frac{N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \times \omega_k}{\sum_{k'=1}^K N(\boldsymbol{x}_n|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'}) \times \omega_{k'}}
\end{aligned}
$$

Estimation with soft r_{nk}

We define $r_{nk} = p(z_n = k|\boldsymbol{x}_n)$

Estimation with soft r_{nk}

We define $r_{nk} = p(z_n = k|\boldsymbol{x}_n)$

- Recall that r_{nk} was previously binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component
- Each $\boldsymbol{x}_n$ is assigned to a component fractionally according to $p(z_n = k|\boldsymbol{x}_n)$

Estimation with soft r_{nk}

We define $r_{nk} = p(z_n = k|\boldsymbol{x}_n)$

- Recall that r_{nk} was previously binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component
- Each $\boldsymbol{x}_n$ is assigned to a component fractionally according to $p(z_n = k|\boldsymbol{x}_n)$

M-Step: If we solve for the MLE of $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ given soft r_{nk}s, we get the same expressions as before!

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_k \sum_n r_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

But remember, we're 'cheating' by using $\boldsymbol{\theta}$ to compute r_{nk}!

EM procedure for GMM

Alternate between estimating r_{nk} and estimating parameters θ

- Step 0: Initialize θ with some values (random or otherwise)
- Step 1: E-Step: Set $r_{nk} = p(z_n = k|\boldsymbol{x}_n)$ with the current values of θ using Bayes Rule
- Step 2: M-Step: Update θ using the r_{nk}s from Step 2 using MLE
- Step 3: Go back to Step 1.

At the end convert r_{nk} back to binary by setting the largest r_{nk} for point x_n to 1 and others to 0.

GMMs and K-means

GMMs provide probabilistic interpretation for K-means

GMMs and K-means

GMMs provide probabilistic interpretation for K-means

GMMs reduce to K-means under the following assumptions (in which case EM for GMM parameter estimation simplifies to K-means):

- Assume all Gaussians have $\sigma^2 \boldsymbol{I}$ covariance matrices
- Further assume $\sigma \to 0$, so we only need to estimate $\boldsymbol{\mu}_k$, i.e., means

K-means is often called "hard" GMM or GMMs is called "soft" K-means

The posterior r_{nk} provides a probabilistic assignment for $\boldsymbol{x}_n$ to cluster k

GMMs vs. k-means

Pros/Cons

- k-means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering

GMMs vs. k-means

Pros/Cons

- k-means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering
- GMMs can be more accurate, as they model more information (soft clustering, variance), but can be more expensive to compute

GMMs vs. k-means

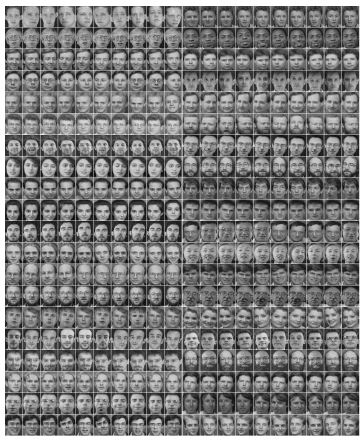
Pros/Cons

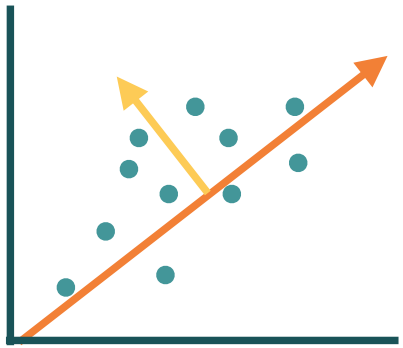
- k-means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering
- GMMs can be more accurate, as they model more information (soft clustering, variance), but can be more expensive to compute
- Both methods have a similar set of practical issues (having to select k, the distance, and the initialization)

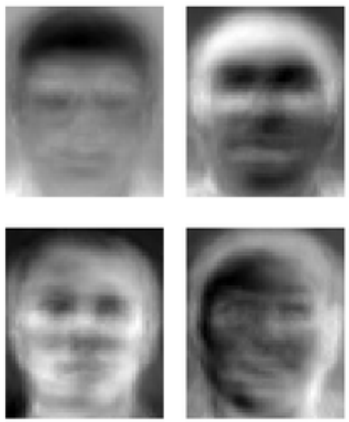
Principal Component Analysis (PCA)

Task 4: Embedding

How to reduce size of dataset?







input: large dataset $\{x\}$ **find**: sources of variation **return**: representation $\{z\}$

Topics: Dimensionality Reduction, PCA

High-dimensional Data

Raw data can be complex and high-dimensional

- To understand a phenomenon we measure various related quantities

High-dimensional Data

Raw data can be complex and high-dimensional

- To understand a phenomenon we measure various related quantities
- If we knew what to measure or how to represent our measurements, we might find simple relationships

High-dimensional Data

Raw data can be complex and high-dimensional

- To understand a phenomenon we measure various related quantities
- If we knew what to measure or how to represent our measurements, we might find simple relationships
- But in practice we often measure redundant signals, e.g., US and European shoe sizes

High-dimensional Data

Raw data can be complex and high-dimensional

- To understand a phenomenon we measure various related quantities
- If we knew what to measure or how to represent our measurements, we might find simple relationships
- But in practice we often measure redundant signals, e.g., US and European shoe sizes
- We also represent data via the method by which it was gathered, e.g., pixel representation of brain imaging data

Dimensionality Reduction

Issues

1. Measure redundant signals
2. Represent data via the method by which it was gathered

Dimensionality Reduction

Issues

1. Measure redundant signals
2. Represent data via the method by which it was gathered

Goal:

Find a "better" representation for data

1. To visualize and discover hidden patterns
2. Preprocessing for supervised task

Dimensionality Reduction

Issues

1. Measure redundant signals
2. Represent data via the method by which it was gathered

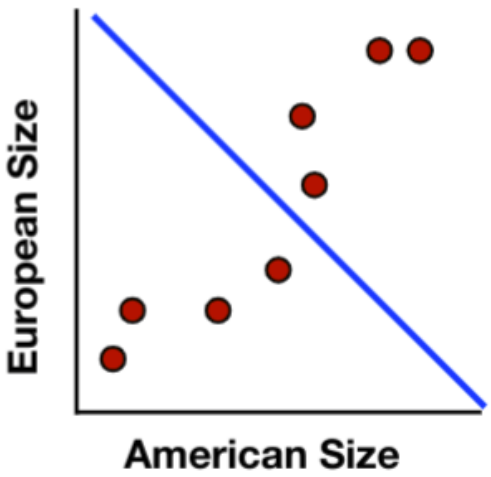
Goal:

Find a "better" representation for data

1. To visualize and discover hidden patterns
2. Preprocessing for supervised task

How do we define "better"?

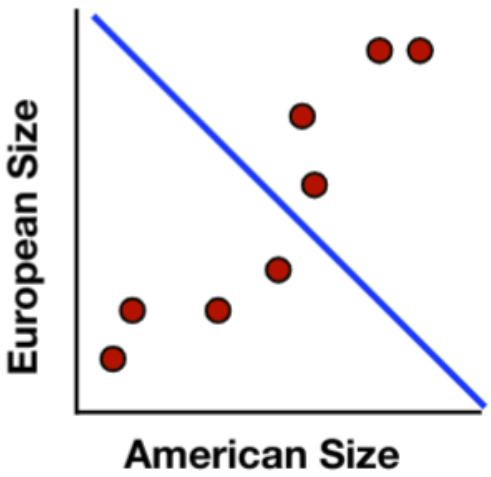
Dimensionality Reduction (Eg. Shoe Size)



We take noisy measurements on the European and American scales

- Modulo noise, we expect perfect correlation

Dimensionality Reduction (Eg. Shoe Size)

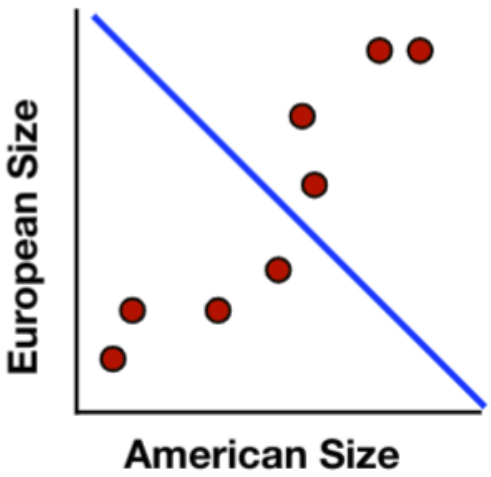


We take noisy measurements on the European and American scales

- Modulo noise, we expect perfect correlation

How can we do better, i.e., find a simpler, compact representation?

Dimensionality Reduction (Eg. Shoe Size)



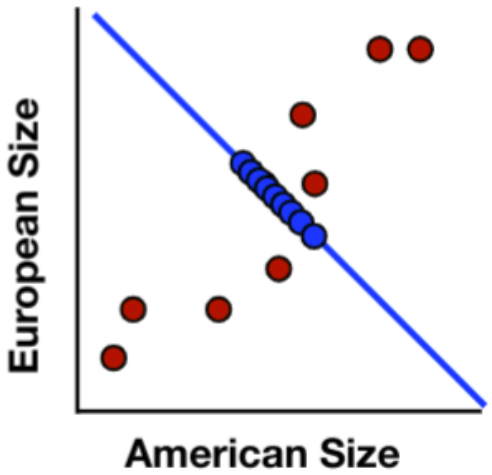
We take noisy measurements on the European and American scales

- Modulo noise, we expect perfect correlation

How can we do better, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction

Dimensionality Reduction (Eg. Shoe Size)



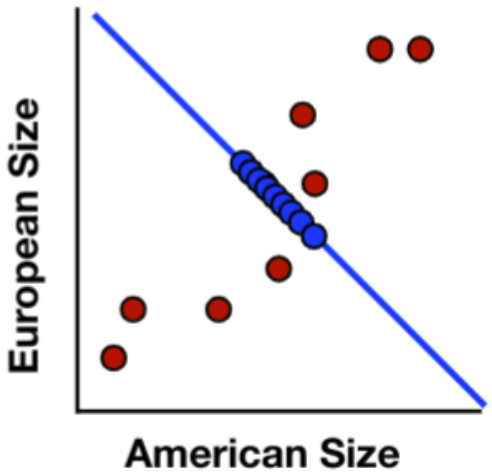
We take noisy measurements on the European and American scales

- Modulo noise, we expect perfect correlation

How can we do better, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction

Dimensionality Reduction (Eg. Shoe Size)



We take noisy measurements on the European and American scales

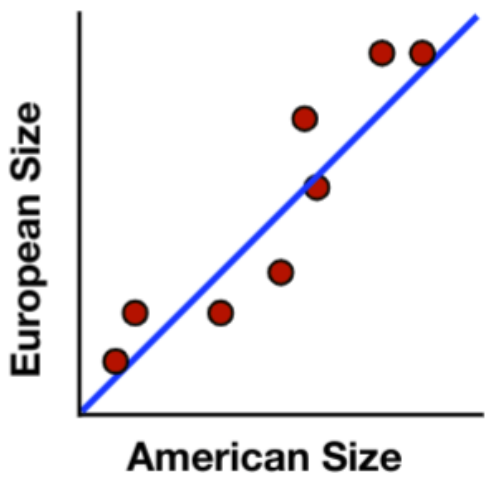
- Modulo noise, we expect perfect correlation

How can we do better, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction

Can we pick a better direction?

Dimensionality Reduction (Eg. Shoe Size)



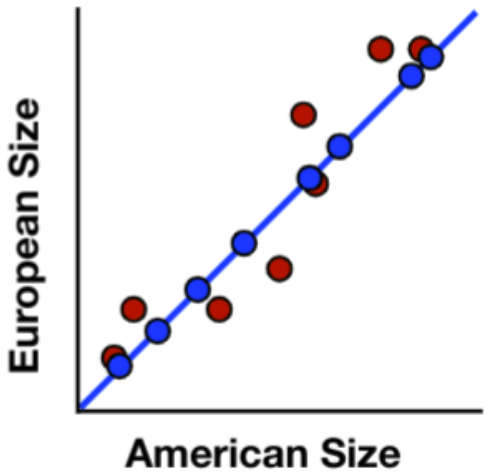
We take noisy measurements on the European and American scales

- Modulo noise, we expect perfect correlation

How can we do better, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction

Dimensionality Reduction (Eg. Shoe Size)



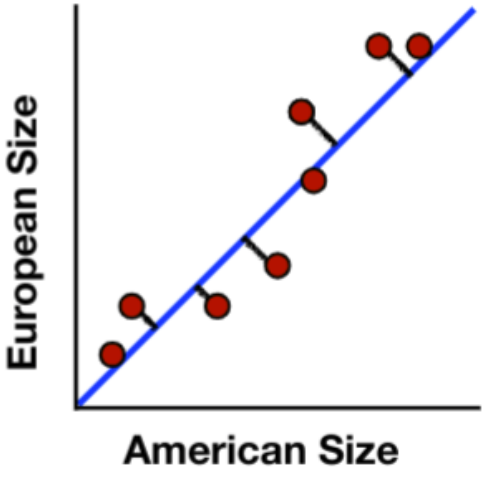
We take noisy measurements on the European and American scales

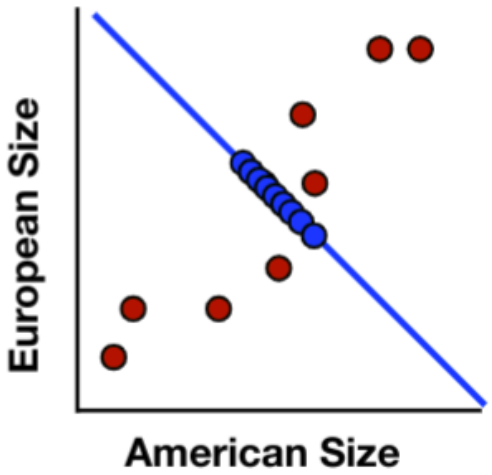
- Modulo noise, we expect perfect correlation

How can we do better, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction

Goal: Minimize Reconstruction Error





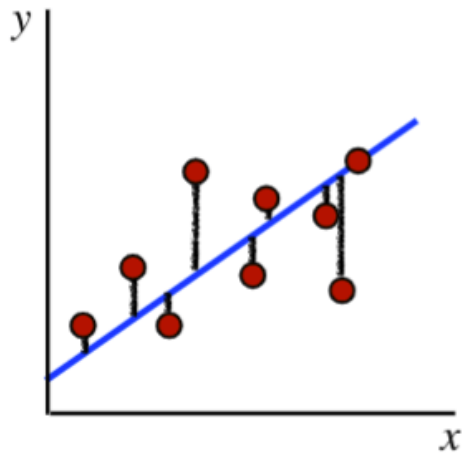
Minimize Euclidean distances between original points and their projections

PCA solution solves this problem!

Goal: Minimize Reconstruction Error

Linear Regression

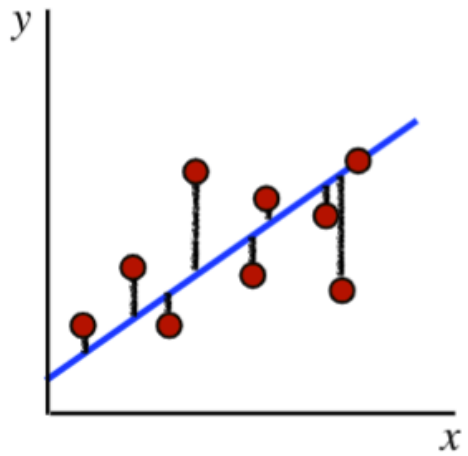
Predict y from x. Evaluate predictions (represented by blue line) by vertical distances between points and the line.



Goal: Minimize Reconstruction Error

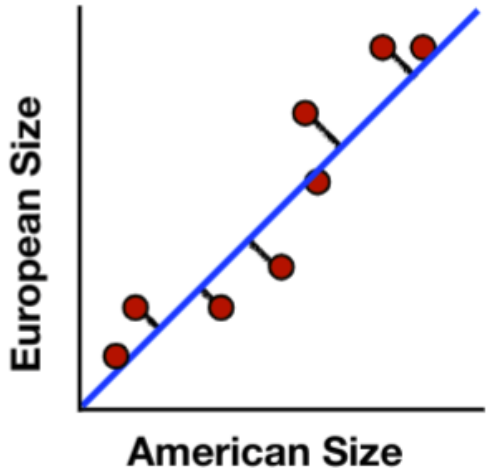
Linear Regression

Predict y from x. Evaluate predictions (represented by blue line) by vertical distances between points and the line.



PCA

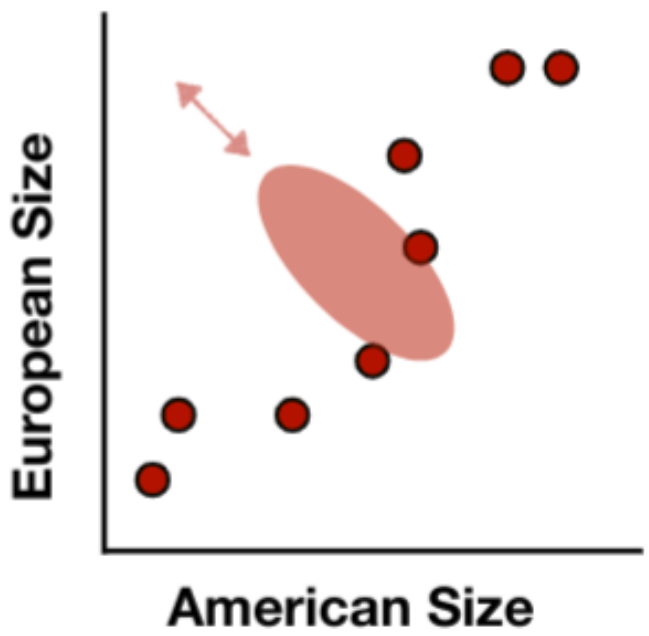
Reconstruct 2D data via 2D data with single degree of freedom. Evaluate reconstructions (represented by blue line) by Euclidean distances.



The distinction is even more apparent with high-dimensional data.

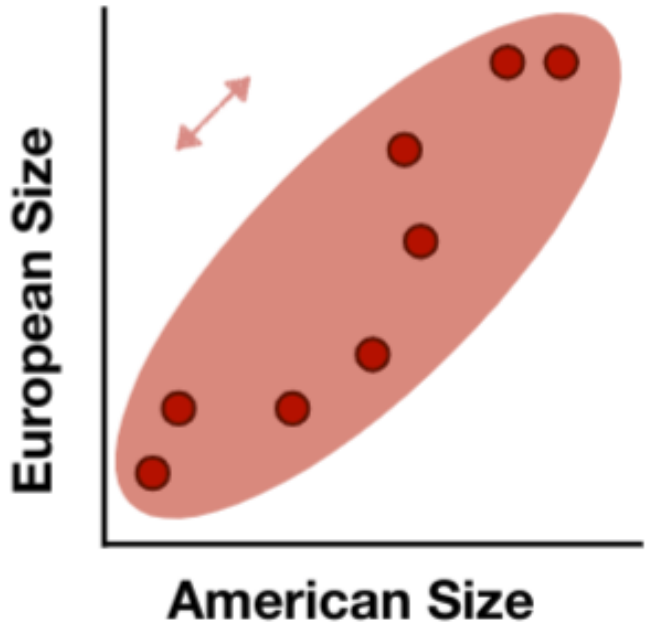
Goal: Maximize Variance and Minimize Feature-Correlation

- To identify patterns we want to study variation across observations
- Can we do "better", i.e., find a compact representation that captures variation?



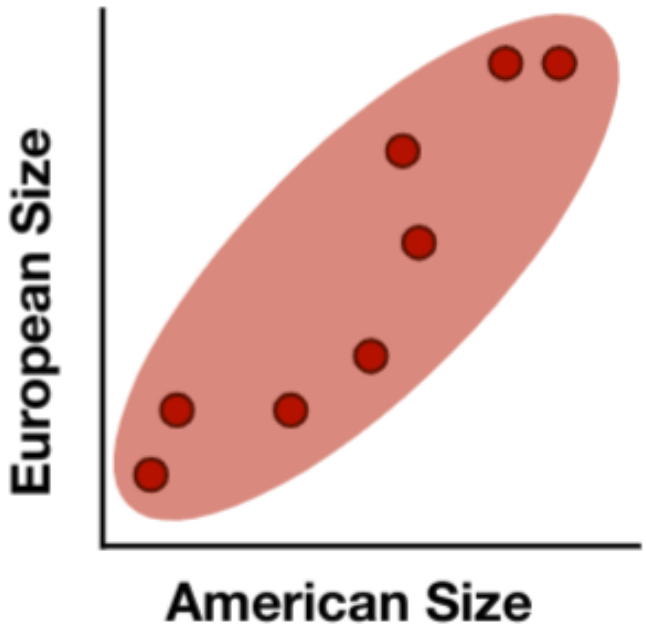
Goal: Maximize Variance and Minimize Feature-Correlation

- To identify patterns we want to study variation across observations
- Can we do "better", i.e., find a compact representation that captures variation?



Goal: Maximize Variance and Minimize Feature-Correlation

- To identify patterns we want to study variation across observations
- Can we do "better", i.e., find a compact representation that captures variation?
- PCA solution finds directions of maximal variance that are orthogonal to each other



PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data, each row is a data point)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")
- Linearity assumption ($\mathbf{Z} = \mathbf{XP}$) simplifies problem

$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{X} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{P} \\ \\ \end{bmatrix}$$

PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data, each row is a data point)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")
- Linearity assumption ($\mathbf{Z} = \mathbf{XP}$) simplifies problem

$$\begin{bmatrix} & \mathbf{Z} & \end{bmatrix} = \begin{bmatrix} & \mathbf{X} & \end{bmatrix} \begin{bmatrix} & \mathbf{P} & \end{bmatrix}$$

- Projecting each row of X along the column vectors of $\mathbf{P}$
- How do we design the matrix $\mathbf{P}$?

Step 1: Zero-Center All the Features

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $\mathbf{x}_j^{(i)}$: j^{th} feature value for i^{th} point
- μ_j: mean of j^{th} feature
- Deduct μ_j from all features

$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \underbrace{\begin{bmatrix} | & | & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_d \\ | & | & | \end{bmatrix}}_{\mathbf{X}} \begin{bmatrix} \\ \mathbf{P} \\ \\ \end{bmatrix}$$

Step 1: Zero-Center All the Features

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $\mathbf{x}_j^{(i)}$: j^{th} feature value for i^{th} point
- μ_j: mean of j^{th} feature
- Deduct μ_j from all features

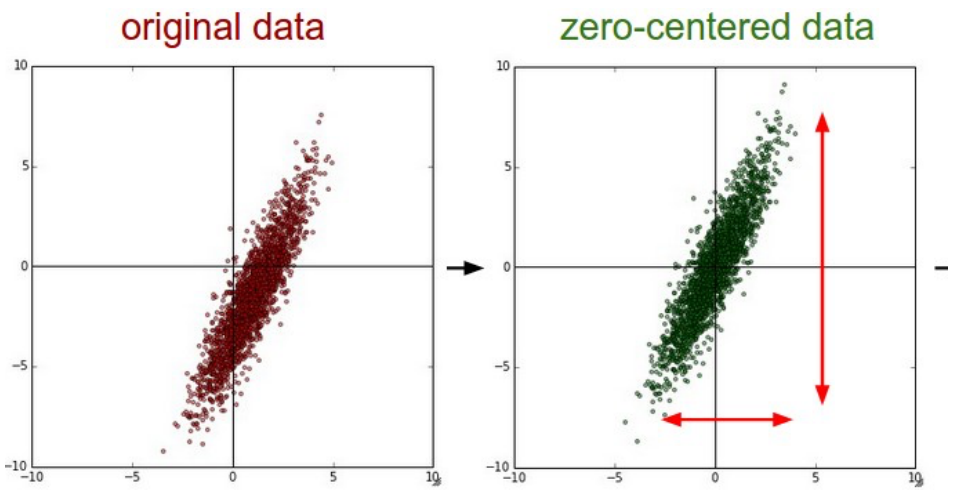
$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{x}_1 - \mu_1 & \dots & \mathbf{x}_d - \mu_d \\ | & | & | \end{bmatrix} \begin{bmatrix} \\ \mathbf{P} \\ \\ \end{bmatrix}$$

From now on, assume that all columns of X are zero-centered

Step 1: Zero-Center All the Features

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $\mathbf{x}_j^{(i)}$: j^{th} feature vector for i^{th} point
- μ_j: mean of j^{th} feature
- Deduct μ_j from all features



Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Variance and Co-variance of the Features

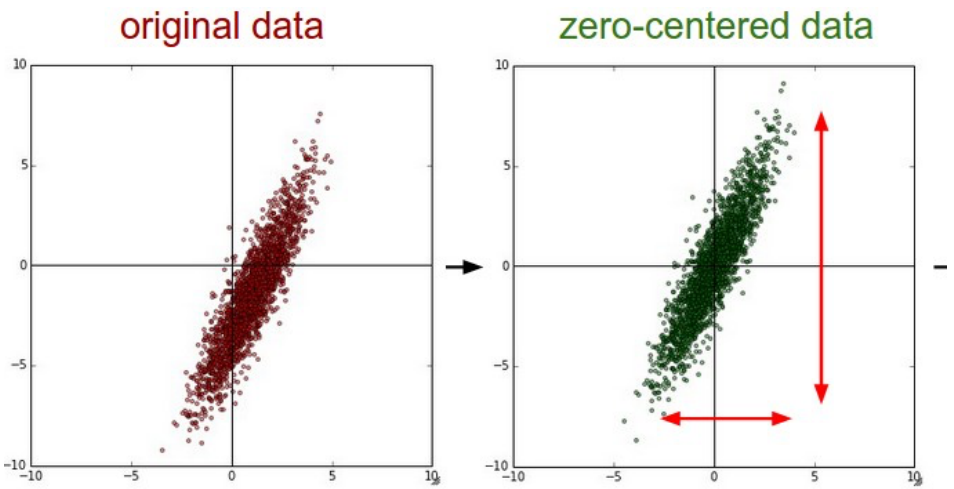
Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$



Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero $\rightarrow$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero $\rightarrow$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero $\rightarrow$ uncorrelated features
- Large magnitude $\rightarrow$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero $\rightarrow$ uncorrelated features
- Large magnitude $\rightarrow$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero $\rightarrow$ uncorrelated features
- Large magnitude $\rightarrow$ (anti) correlated/ redundant
- $\sigma_{12} = \sigma_1^2 = \sigma_2^2 \rightarrow$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero $\rightarrow$ uncorrelated features
- Large magnitude $\rightarrow$ (anti) correlated/ redundant
- $\sigma_{12} = \sigma_1^2 = \sigma_2^2 \rightarrow$

Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- Covariance of 1st and 2nd feature $\sigma_{12} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_1^{(i)}\mathbf{x}_2^{(i)}$

Properties of Co-variance σ_{12}

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero $\rightarrow$ uncorrelated features
- Large magnitude $\rightarrow$ (anti) correlated/ redundant
- $\sigma_{12} = \sigma_1^2 = \sigma_2^2 \rightarrow$ features are the same

Covariance Matrix

- Covariance matrix generalizes this idea for many features

Covariance Matrix

- Covariance matrix generalizes this idea for many features
- For zero-centered features this $d \times d$ matrix is

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

$$= \frac{1}{n}\begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ & \vdots & \\ -- & \mathbf{x}_d^T & -- \end{bmatrix}\begin{bmatrix} | & | & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_d \\ | & | & | \end{bmatrix}$$

Covariance Matrix

- Covariance matrix generalizes this idea for many features
- For zero-centered features this $d \times d$ matrix is

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

$$= \frac{1}{n}\begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ & \vdots & \\ -- & \mathbf{x}_d^T & -- \end{bmatrix}\begin{bmatrix} | & | & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_d \\ | & | & | \end{bmatrix}$$

- i^{th} diagonal entry equals variance of i^{th} feature

Covariance Matrix

- Covariance matrix generalizes this idea for many features
- For zero-centered features this $d \times d$ matrix is

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

$$= \frac{1}{n}\begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ & \vdots & \\ -- & \mathbf{x}_d^T & -- \end{bmatrix}\begin{bmatrix} | & | & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_d \\ | & | & | \end{bmatrix}$$

- i^{th} diagonal entry equals variance of i^{th} feature
- $(i, j)^{th}$ diagonal entry equals covariance of i^{th} and j^{th} features

Covariance Matrix

- Covariance matrix generalizes this idea for many features
- For zero-centered features this $d \times d$ matrix is

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

$$= \frac{1}{n}\begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ & \vdots & \\ -- & \mathbf{x}_d^T & -- \end{bmatrix}\begin{bmatrix} | & | & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_d \\ | & | & | \end{bmatrix}$$

- i^{th} diagonal entry equals variance of i^{th} feature
- $(i,j)^{th}$ diagonal entry equals covariance of i^{th} and j^{th} features
- Symmetric (makes sense given definition of covariance)

Covariance Matrix

- Covariance matrix generalizes this idea for many features
- For zero-centered features this $d \times d$ matrix is

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

$$= \frac{1}{n}\begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ & \vdots & \\ -- & \mathbf{x}_d^T & -- \end{bmatrix}\begin{bmatrix} | & | & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_d \\ | & | & | \end{bmatrix}$$

- i^{th} diagonal entry equals variance of i^{th} feature
- $(i,j)^{th}$ diagonal entry equals covariance of i^{th} and j^{th} features
- Symmetric (makes sense given definition of covariance)
- What is the covariance matrix in terms of mean-removed samples?

Recall: PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")

Recall: PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")

$$\begin{aligned}\mathbf{C_Z} &= \frac{1}{n}\mathbf{Z}^T\mathbf{Z} \\ &= \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{XP} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}\end{aligned}$$

Recall: PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")

$$\begin{aligned}\mathbf{C_Z} &= \frac{1}{n}\mathbf{Z}^T\mathbf{Z} \\ &= \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{XP} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}\end{aligned}$$

We want the covariance matrix $\mathbf{C_Z}$ to have the following properties:

- No feature correlation, i.e., all off-diagonals in $\mathbf{C_Z}$ are zero

Recall: PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")

$$\begin{aligned}\mathbf{C_Z} &= \frac{1}{n}\mathbf{Z}^T\mathbf{Z} \\ &= \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{XP} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}\end{aligned}$$

We want the covariance matrix $\mathbf{C_Z}$ to have the following properties:

- No feature correlation, i.e., all off-diagonals in $\mathbf{C_Z}$ are zero
- Rank-ordered features by variance, i.e., sorted diagonals of $\mathbf{C_Z}$

Recall: PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")

$$\begin{aligned}\mathbf{C_Z} &= \frac{1}{n}\mathbf{Z}^T\mathbf{Z} \\ &= \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{XP} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}\end{aligned}$$

We want the covariance matrix $\mathbf{C_Z}$ to have the following properties:

- No feature correlation, i.e., all off-diagonals in $\mathbf{C_Z}$ are zero
- Rank-ordered features by variance, i.e., sorted diagonals of $\mathbf{C_Z}$

Recall: PCA Formulation

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores")

$$\begin{aligned}\mathbf{C_Z} &= \frac{1}{n}\mathbf{Z}^T\mathbf{Z} \\ &= \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{XP} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}\end{aligned}$$

We want the covariance matrix $\mathbf{C_Z}$ to have the following properties:

- No feature correlation, i.e., all off-diagonals in $\mathbf{C_Z}$ are zero
- Rank-ordered features by variance, i.e., sorted diagonals of $\mathbf{C_Z}$

Idea: Eigen-value decomposition of $\mathbf{C_X}$

Eigen-value Decomposition

The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and therefore has a singular value decomposition:

$$\mathbf{C_X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

$$= \begin{bmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_d \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} \begin{bmatrix} -- & \mathbf{v}_1^T & -- \\ -- & \mathbf{v}_2^T & -- \\ & \vdots & \\ -- & \mathbf{v}_d^T & -- \end{bmatrix}$$

- Eigen-values of $\mathbf{C_X}$ are ordered $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$

Eigen-value Decomposition

The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and therefore has a singular value decomposition:

$$\mathbf{C_X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

$$= \begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \dots & \mathbf{v}_d \\ | & | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} \begin{bmatrix} -- & \mathbf{v}_1^T & -- \\ -- & \mathbf{v}_2^T & -- \\ & \vdots & \\ -- & \mathbf{v}_d^T & -- \end{bmatrix}$$

- Eigen-values of $\mathbf{C_X}$ are ordered $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$
- $\mathbf{Q}$ is an matrix of d orthonormal eigen-vectors $\mathbf{v}_i$ of $\mathbf{C_X}$

Recall: Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

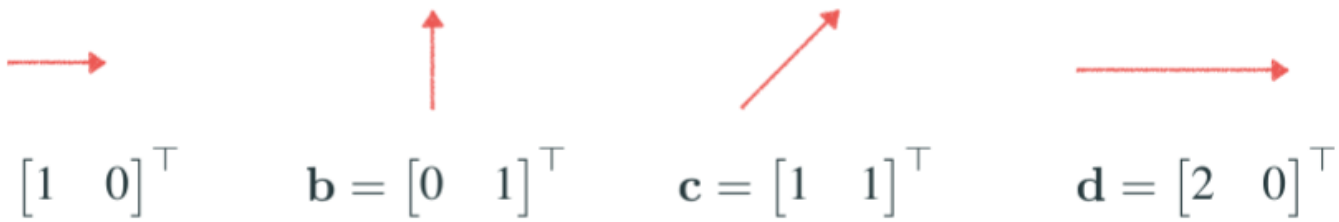
- Equivalently, their dot product equals zero
- $\mathbf{a}^T\mathbf{b} = 0$ and $\mathbf{d}^T\mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

Recall: Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^T\mathbf{b} = 0$ and $\mathbf{d}^T\mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

Orthonormal vectors are orthogonal and have unit norm



$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

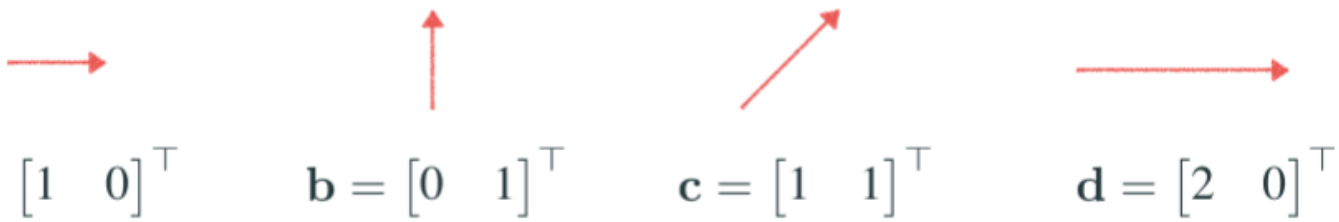
- Are $\mathbf{a}$ and $\mathbf{b}$ orthonormal?

Recall: Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^T\mathbf{b} = 0$ and $\mathbf{d}^T\mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

Orthonormal vectors are orthogonal and have unit norm



$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

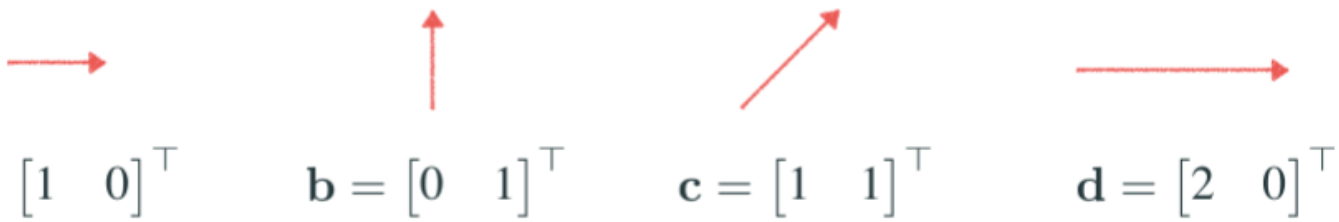
- Are $\mathbf{a}$ and $\mathbf{b}$ orthonormal?

Recall: Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^T\mathbf{b} = 0$ and $\mathbf{d}^T\mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

Orthonormal vectors are orthogonal and have unit norm



$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

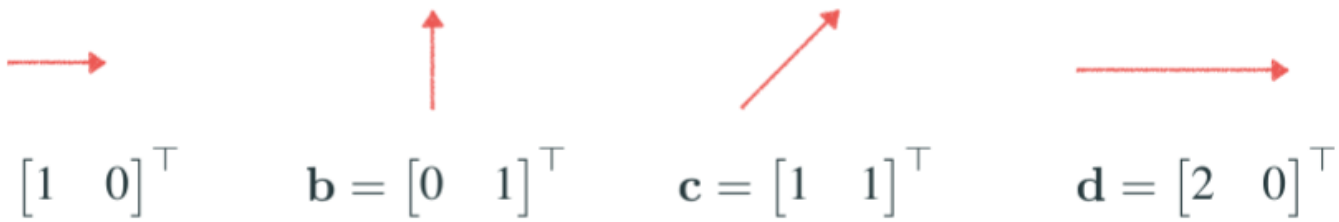
- Are $\mathbf{a}$ and $\mathbf{b}$ orthonormal? **YES**
- Are $\mathbf{b}$ and $\mathbf{d}$ orthonormal?

Recall: Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^T\mathbf{b} = 0$ and $\mathbf{d}^T\mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

Orthonormal vectors are orthogonal and have unit norm



$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

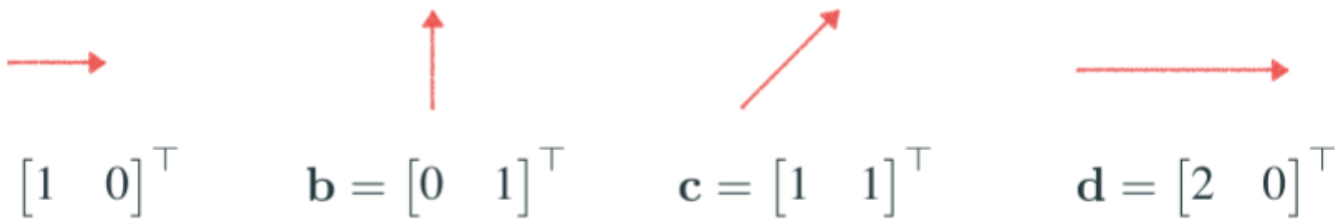
- Are $\mathbf{a}$ and $\mathbf{b}$ orthonormal? **YES**
- Are $\mathbf{b}$ and $\mathbf{d}$ orthonormal?

Recall: Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^T\mathbf{b} = 0$ and $\mathbf{d}^T\mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

Orthonormal vectors are orthogonal and have unit norm



$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

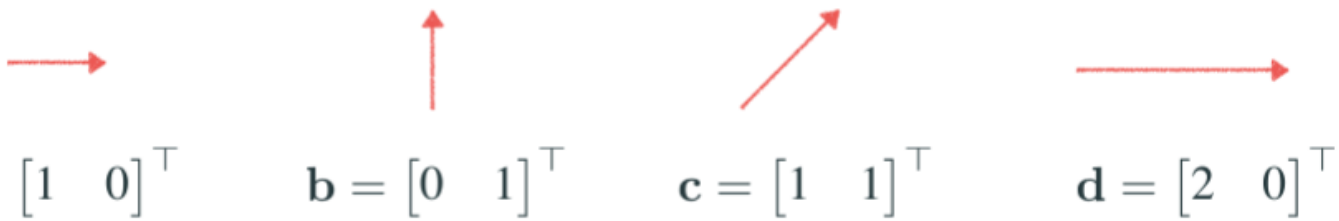
- Are $\mathbf{a}$ and $\mathbf{b}$ orthonormal? **YES**
- Are $\mathbf{b}$ and $\mathbf{d}$ orthonormal? **NO**

Recall: Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^T\mathbf{b} = 0$ and $\mathbf{d}^T\mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

Orthonormal vectors are orthogonal and have unit norm



$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

- Are $\mathbf{a}$ and $\mathbf{b}$ orthonormal? **YES**
- Are $\mathbf{b}$ and $\mathbf{d}$ orthonormal? **NO**

If $\mathbf{Q}$ is an orthonormal matrix, then $\mathbf{Q}^T = \mathbf{Q}^{-1}$

PCA Solution

- The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and it can be decomposed as follows

$$\mathbf{C_X} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$
$$\mathbf{Q}^T\mathbf{C_X}\mathbf{Q} = \boldsymbol{\Lambda}$$

PCA Solution

- The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and it can be decomposed as follows

$$\mathbf{C_X} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$
$$\mathbf{Q}^T\mathbf{C_X}\mathbf{Q} = \boldsymbol{\Lambda}$$

- Recall our desired PCA solution

PCA Solution

- The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and it can be decomposed as follows

$$\mathbf{C_X} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$
$$\mathbf{Q}^T\mathbf{C_X}\mathbf{Q} = \boldsymbol{\Lambda}$$

- Recall our desired PCA solution

PCA Solution

- The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and it can be decomposed as follows

$$\mathbf{C_X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$
$$\mathbf{Q}^T\mathbf{C_X}\mathbf{Q} = \mathbf{\Lambda}$$

- Recall our desired PCA solution

$$\begin{aligned}\mathbf{C_Z} &= \frac{1}{n}\mathbf{Z}^T\mathbf{Z} \\ &= \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{X}\mathbf{P} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}\end{aligned}$$

- $\mathbf{Z} = \mathbf{X}\mathbf{P}$ is $n \times k$ (reduced representation, PCA "scores")

PCA Solution

- The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and it can be decomposed as follows

$$\mathbf{C_X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$
$$\mathbf{Q}^T\mathbf{C_X}\mathbf{Q} = \mathbf{\Lambda}$$

- Recall our desired PCA solution

$$\begin{aligned}\mathbf{C_Z} &= \frac{1}{n}\mathbf{Z}^T\mathbf{Z} \\ &= \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{X}\mathbf{P} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}\end{aligned}$$

- $\mathbf{Z} = \mathbf{X}\mathbf{P}$ is $n \times k$ (reduced representation, PCA "scores")
- $\mathbf{C_Z}$ should have zero off-diagonal entries

PCA Solution

$$\mathbf{\Lambda} = \mathbf{Q}^T \mathbf{C_X} \mathbf{Q}$$

$$\mathbf{C_Z} = \mathbf{P}^T \mathbf{C_X} \mathbf{P}$$

- Can we just take $\mathbf{Q} = \mathbf{P}$ and $\mathbf{C_Z} = \mathbf{\Lambda}$?

PCA Solution

$$\mathbf{\Lambda} = \mathbf{Q}^T \mathbf{C_X} \mathbf{Q}$$

$$\mathbf{C_Z} = \mathbf{P}^T \mathbf{C_X} \mathbf{P}$$

- Can we just take $\mathbf{Q} = \mathbf{P}$ and $\mathbf{C_Z} = \mathbf{\Lambda}$?

PCA Solution

$$\mathbf{\Lambda} = \mathbf{Q}^T \mathbf{C_X} \mathbf{Q}$$

$$\mathbf{C_Z} = \mathbf{P}^T \mathbf{C_X} \mathbf{P}$$

- Can we just take $\mathbf{Q} = \mathbf{P}$ and $\mathbf{C_Z} = \mathbf{\Lambda}$? No, because $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores").
- Choose $\mathbf{P}$ as the first k columns of $\mathbf{Q}$

PCA Solution

$$\mathbf{\Lambda} = \mathbf{Q}^T \mathbf{C_X} \mathbf{Q}$$

$$\mathbf{C_Z} = \mathbf{P}^T \mathbf{C_X} \mathbf{P}$$

- Can we just take $\mathbf{Q} = \mathbf{P}$ and $\mathbf{C_Z} = \mathbf{\Lambda}$? No, because $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA "scores").
- Choose $\mathbf{P}$ as the first k columns of $\mathbf{Q}$
- Capture the k directions of maximum variance.

PCA: Example

Suppose the covariance of X is

$$\mathbf{C_X} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Question: Find $\mathbf{P}$ for $k = 1$.

PCA: Example

Suppose the covariance of X is

$$\mathbf{C_X} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Question: Find $\mathbf{P}$ for $k = 1$.

Eigen-value decomposition of $\mathbf{C_X}$

$$\mathbf{C_X} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

PCA: Example

Suppose the covariance of X is

$$\mathbf{C_X} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Question: Find $\mathbf{P}$ for $k = 1$.

Eigen-value decomposition of $\mathbf{C_X}$

$$\mathbf{C_X} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Thus $\mathbf{P} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$

Choosing k, the number of components

How should we pick the dimension of the new representation?

Choosing k, the number of components

How should we pick the dimension of the new representation?

Visualization:

Pick top 2 or 3 dimensions for plotting purposes

Choosing k, the number of components

How should we pick the dimension of the new representation?

Visualization:

Pick top 2 or 3 dimensions for plotting purposes

Other analyses:

Capture "most" of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted

Choosing k, the number of components

How should we pick the dimension of the new representation?

Visualization:

Pick top 2 or 3 dimensions for plotting purposes

Other analyses:

Capture "most" of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
- Fraction of retained variance: $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$

Choosing k, the number of components

How should we pick the dimension of the new representation?

Visualization:

Pick top 2 or 3 dimensions for plotting purposes

Other analyses:

Capture "most" of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
- Fraction of retained variance: $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$

Choosing k, the number of components

How should we pick the dimension of the new representation?

Visualization:

Pick top 2 or 3 dimensions for plotting purposes

Other analyses:

Capture "most" of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
- Fraction of retained variance: $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$

Can choose k such that we retain some fraction of the variance, eg. 95%

Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Various extensions to PCA with different underlying assumptions, e.g., manifold learning, Kernel PCA, ICA

Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Various extensions to PCA with different underlying assumptions, e.g., manifold learning, Kernel PCA, ICA
- Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA

Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Various extensions to PCA with different underlying assumptions, e.g., manifold learning, Kernel PCA, ICA
- Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA
- PCA results dependent on scaling of data

Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Various extensions to PCA with different underlying assumptions, e.g., manifold learning, Kernel PCA, ICA
- Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA
- PCA results dependent on scaling of data
- Data is sometimes rescaled in practice before applying PCA

Iterative PCA Algorithm

Problem

- For high-dimensional original features (large d) computing the eigenvalue decomposition of $\mathbf{C}_X$, and sorting the eigenvalues can be intractable.

Iterative PCA Algorithm

Problem

- For high-dimensional original features (large d) computing the eigenvalue decomposition of $\mathbf{C}_X$, and sorting the eigenvalues can be intractable.

Iterative Algorithm

1. Find the top principal component $\mathbf{v}_1$ (need an efficient method for this)

Iterative PCA Algorithm

Problem

- For high-dimensional original features (large d) computing the eigenvalue decomposition of $\mathbf{C}_X$, and sorting the eigenvalues can be intractable.

Iterative Algorithm

1. Find the top principal component $\mathbf{v}_1$ (need an efficient method for this)
2. Replace each datapoint $\mathbf{x}$ by $\mathbf{x} - \mathbf{v}_1\mathbf{v}_1^T\mathbf{x}$, that is, remove the projection on $\mathbf{v}_1$

Iterative PCA Algorithm

Problem

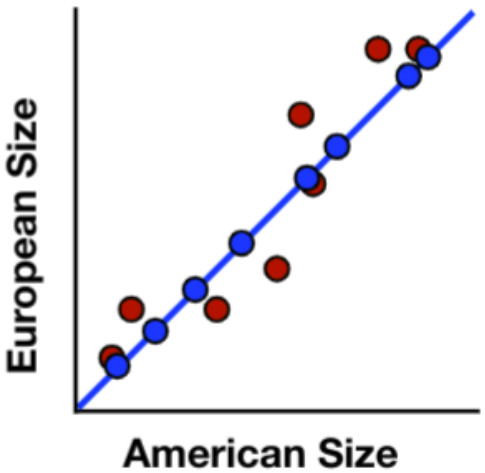
- For high-dimensional original features (large d) computing the eigenvalue decomposition of $\mathbf{C}_X$, and sorting the eigenvalues can be intractable.

Iterative Algorithm

1. Find the top principal component $\mathbf{v}_1$ (need an efficient method for this)
2. Replace each datapoint $\mathbf{x}$ by $\mathbf{x} - \mathbf{v}_1\mathbf{v}_1^T\mathbf{x}$, that is, remove the projection on $\mathbf{v}_1$
3. Recurse until we find k components

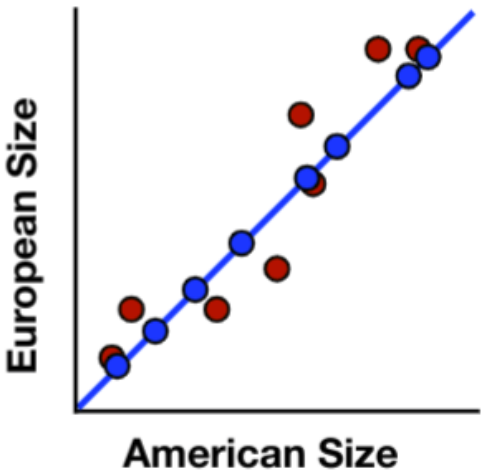
Power Iteration Method to find $\mathbf{v}_1$

1. Find $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$



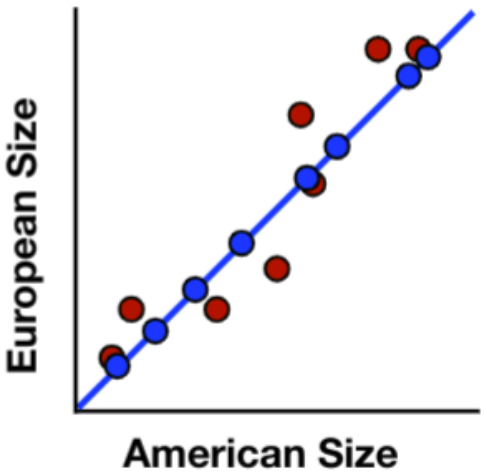
Power Iteration Method to find $\mathbf{v}_1$

1. Find $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$
2. Initialize $\mathbf{v}_1$ to a random normalized vector



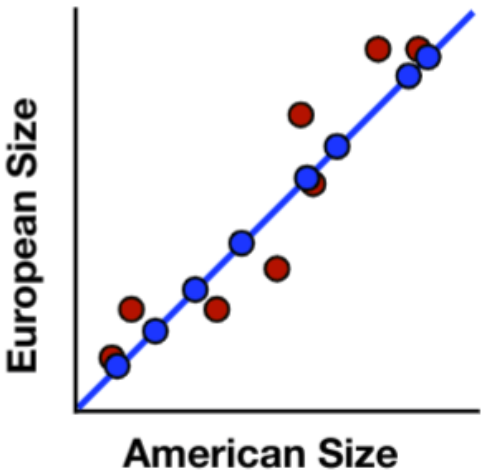
Power Iteration Method to find $\mathbf{v}_1$

1. Find $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$
2. Initialize $\mathbf{v}_1$ to a random normalized vector
3. For many iterations (or until $\mathbf{v}_1$ becomes stable):



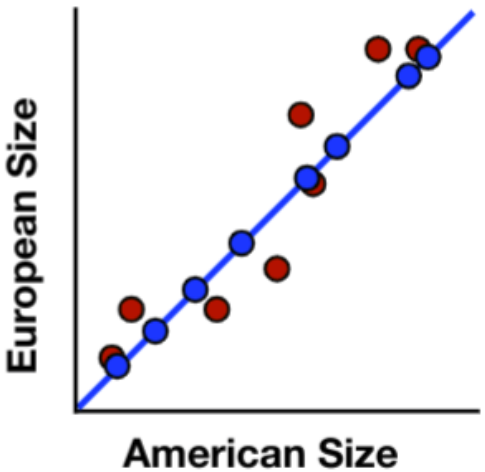
Power Iteration Method to find $\mathbf{v}_1$

1. Find $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$
2. Initialize $\mathbf{v}_1$ to a random normalized vector
3. For many iterations (or until $\mathbf{v}_1$ becomes stable):
   - $\mathbf{v}_1 \leftarrow \mathbf{C_X}\mathbf{v}_1$



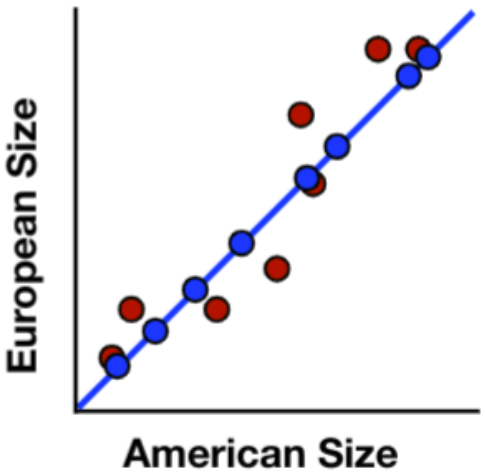
Power Iteration Method to find $\mathbf{v}_1$

1. Find $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$
2. Initialize $\mathbf{v}_1$ to a random normalized vector
3. For many iterations (or until $\mathbf{v}_1$ becomes stable):
   - $\mathbf{v}_1 \leftarrow \mathbf{C_X}\mathbf{v}_1$
   - Normalize $\mathbf{v}_1$ so that $||\mathbf{v}_1||_2^2 = 1$



Power Iteration Method to find $\mathbf{v}_1$

1. Find $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$
2. Initialize $\mathbf{v}_1$ to a random normalized vector
3. For many iterations (or until $\mathbf{v}_1$ becomes stable):
   - $\mathbf{v}_1 \leftarrow \mathbf{C_X}\mathbf{v}_1$
   - Normalize $\mathbf{v}_1$ so that $||\mathbf{v}_1||_2^2 = 1$
4. Return $\mathbf{v}_1$



Why does Eigen-value Decomposition Maximize Variance?

Consider PCA with $k = 1$

- $\mathbf{z} = \mathbf{X}\mathbf{p}$ is $n \times 1$
- $\mathbf{p}$ is $d \times 1$ (first principal component)

$$\begin{aligned}\sigma_{\mathbf{z}}^2 &= \frac{1}{n}||\mathbf{z}||_2^2 \\ &= \frac{1}{n}\mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p} \\ &= \frac{1}{n}\mathbf{p}^T\mathbf{C}_{\mathbf{X}}\mathbf{p}\end{aligned}$$

Why does Eigen-value Decomposition Maximize Variance?

Consider PCA with $k = 1$

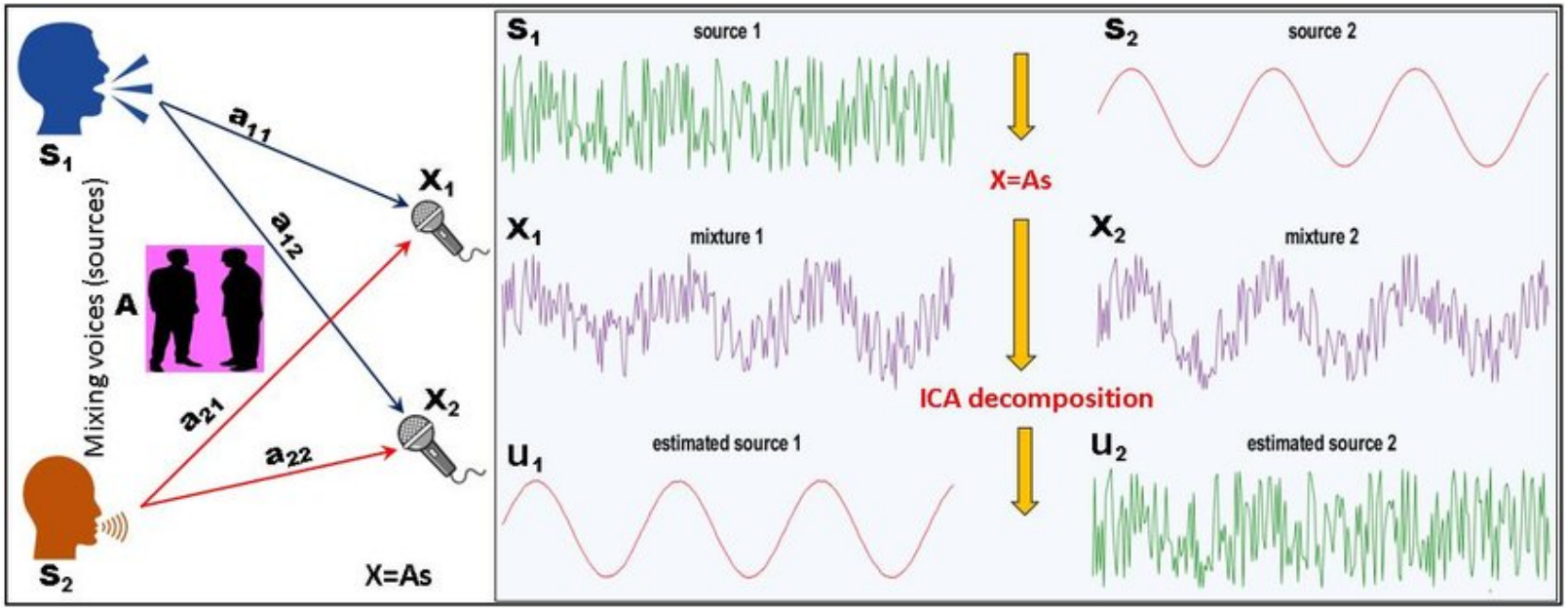
- $\mathbf{z} = \mathbf{X}\mathbf{p}$ is $n \times 1$
- $\mathbf{p}$ is $d \times 1$ (first principal component)

$$\begin{aligned}\sigma_{\mathbf{z}}^2 &= \frac{1}{n}||\mathbf{z}||_2^2 \\ &= \frac{1}{n}\mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p} \\ &= \frac{1}{n}\mathbf{p}^T\mathbf{C}_{\mathbf{X}}\mathbf{p}\end{aligned}$$

- Goal: $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$
- The top eigenvector $\mathbf{v}_1$ is known to achieve the optimum for any symmetric matrix $\mathbf{C}_{\mathbf{X}}$

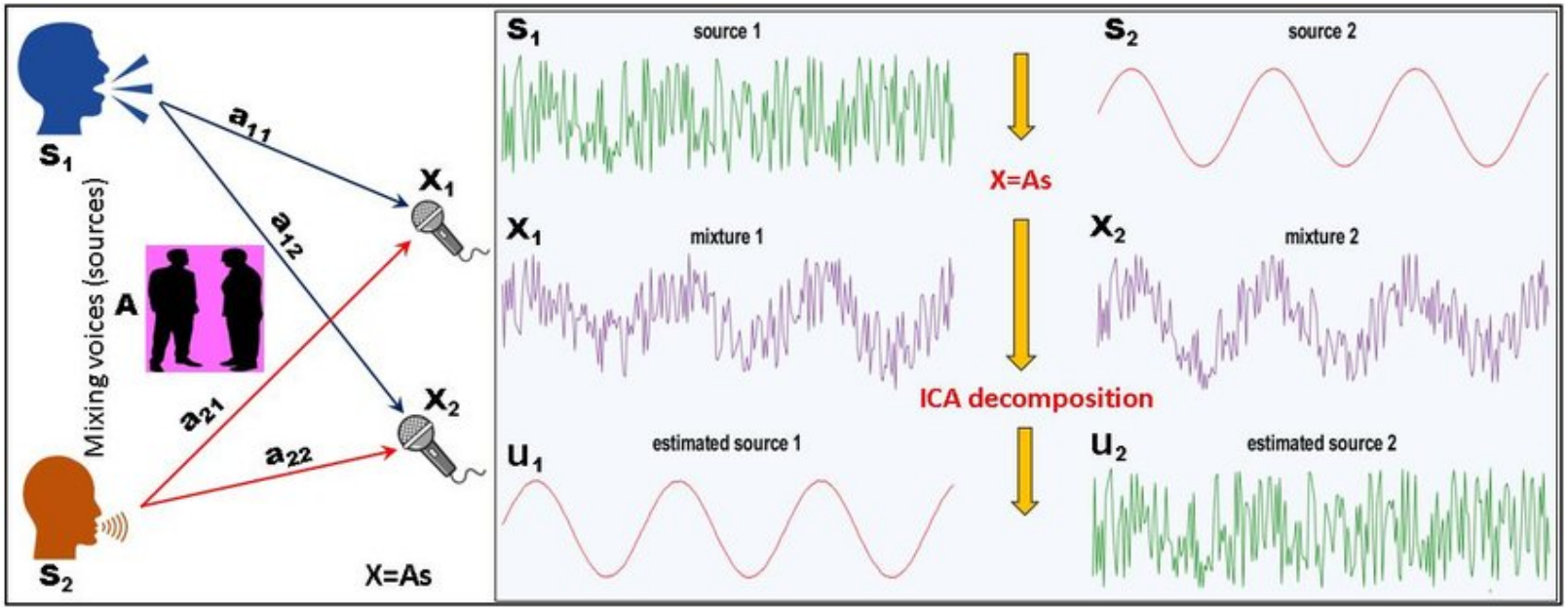
Independent Component Analysis (ICA)

The Cocktail Party Problem



- PCA tries to find an orthogonal representation of the mixed data.

The Cocktail Party Problem

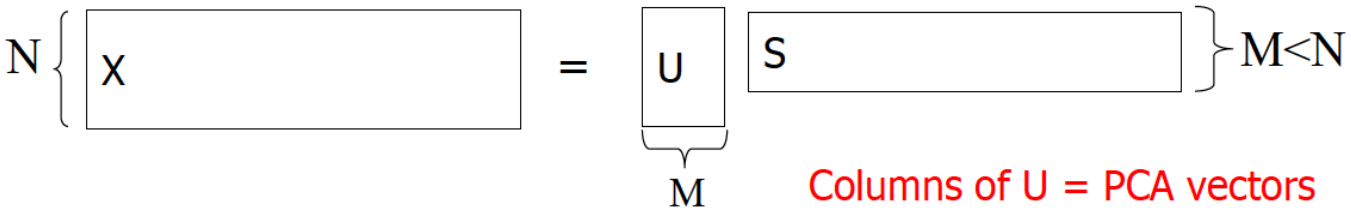


- PCA tries to find an orthogonal representation of the mixed data.
- ICA tries to disentangle the data sources.

How Does ICA Work?

Both ICA and PCA linearly transform the original data with matrix factorization.

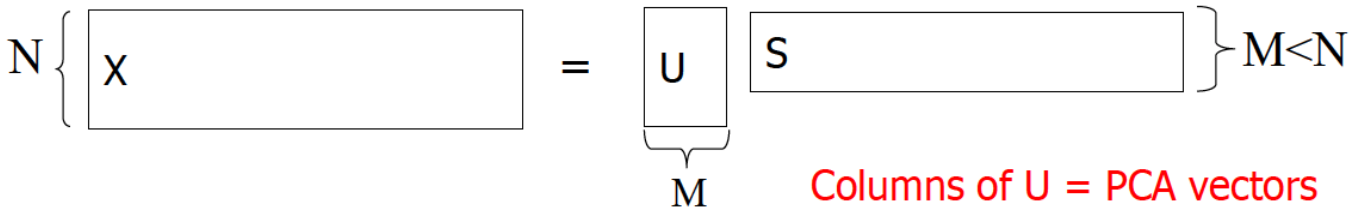
PCA compresses data with low-rank matrix factorization



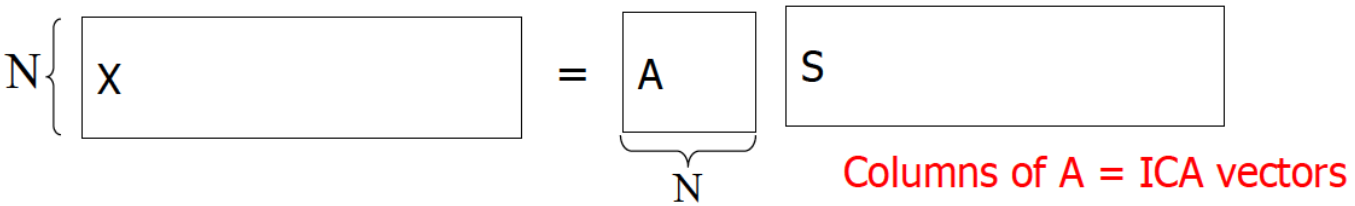
How Does ICA Work?

Both ICA and PCA linearly transform the original data with matrix factorization.

PCA compresses data with low-rank matrix factorization



ICA removes dependencies between data with full-rank matrix factorization



Comparing PCA and ICA

Let $\mathbf{S}$ denote our original data. It can be transformed to a new set of features $\mathbf{X}$:

$$\text{PCA} : \mathbf{X} \approx \mathbf{US},\ \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

$$\text{ICA} : \mathbf{X} \approx \mathbf{AS},\ \mathbf{A} \text{ invertible}$$

- PCA reduces the number of features (compression). ICA does not.

Comparing PCA and ICA

Let $\mathbf{S}$ denote our original data. It can be transformed to a new set of features $\mathbf{X}$:

$$\text{PCA} : \mathbf{X} \approx \mathbf{US},\ \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

$$\text{ICA} : \mathbf{X} \approx \mathbf{AS},\ \mathbf{A} \text{ invertible}$$

- PCA reduces the number of features (compression). ICA does not.
- PCA removes correlations but not higher-order dependencies. ICA removes these dependencies.

Comparing PCA and ICA

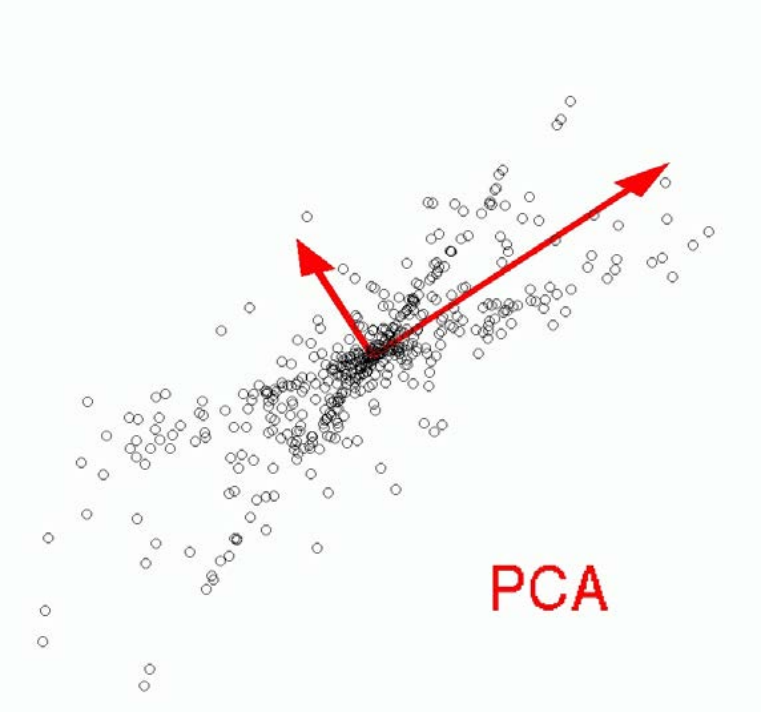
Let $\mathbf{S}$ denote our original data. It can be transformed to a new set of features $\mathbf{X}$:

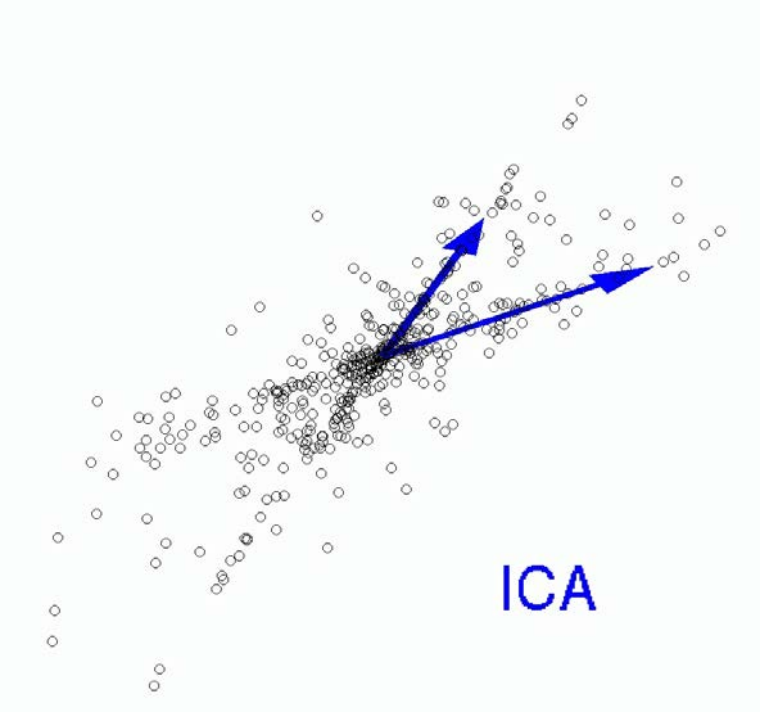
$$\text{PCA} : \mathbf{X} \approx \mathbf{US},\ \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

$$\text{ICA} : \mathbf{X} \approx \mathbf{AS},\ \mathbf{A} \text{ invertible}$$

- PCA reduces the number of features (compression). ICA does not.
- PCA removes correlations but not higher-order dependencies. ICA removes these dependencies.
- PCA allows you to rank the importance of the new features. ICA does not.

ICA vs. PCA





Applications of ICA

- Image denoising: find new representations of a set of images
- Face recognition, face expression recognition
- Feature extraction
- Clustering, classification, deep neural networks
- Timeseries applications: recall the cocktail party example

Applications of ICA

- Image denoising: find new representations of a set of images
- Face recognition, face expression recognition
- Feature extraction
- Clustering, classification, deep neural networks
- Timeseries applications: recall the cocktail party example
  - Medical signal processing: fMRI, ECG, EEG
  - Modeling of the visual cortex, hippocampus
  - Time series analysis
  - Financial applications

Summary

You should know:

- Why we use PCA

Summary

You should know:

- Why we use PCA
- How to execute the PCA algorithm and why it works

Summary

You should know:

- Why we use PCA
- How to execute the PCA algorithm and why it works
- How PCA differs from ICA