18-661 Introduction to Machine Learning

Dimensionality Reduction

Spring 2025

ECE – Carnegie Mellon University

Announcements

- HW 4 is due April 16.
- Mini-exam 3 will be held on April 16.
  - You are allowed one, single-sided, handwritten cheat sheet. No calculators.
  - Mini-exam will cover transformers, distributed learning, k-means, GMMs, and dimensionality reduction.
  - Expect a mix of true/false, multiple choice, and descriptive questions. Practice questions will be released tomorrow.
- Recitation on Friday will go over practice problems for the mini-exam and HW 4's coding question.
- Let us know by Tuesday, April 15 if you cannot take the final exam at the scheduled time (Friday, May 2 at 10am PT/1pm ET/7pm CAT).

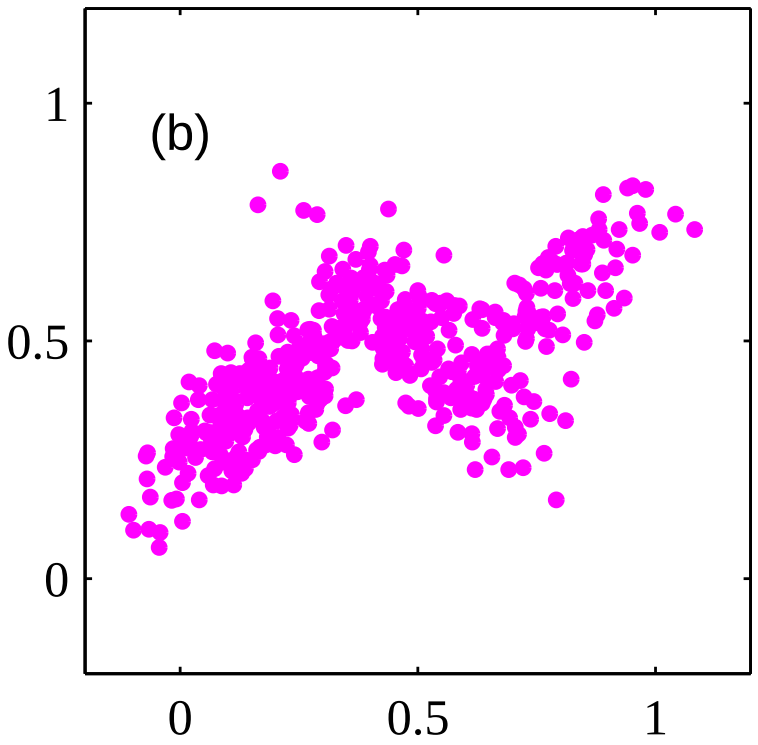
Outline

1. Recap: Gaussian Mixture Models and EM Algorithm
2. Principal Component Analysis (PCA)
3. A Glimpse at Independent Component Analysis (ICA)

Recap: Gaussian Mixture Models and EM Algorithm

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: Formal Definition

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

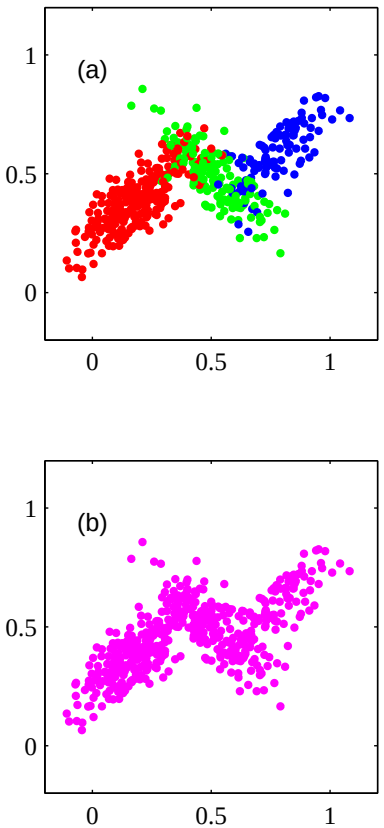
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- 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 that $p(\boldsymbol{x})$ is a probability density function

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

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

Let's first consider the unrealistic case where *we have labels z* ($\mathcal{D}'$)

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

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

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

where $r_{nk} \in \{0, 1\}$ is a binary variable that indicates whether $z_n = k$.

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): $\{\boldsymbol{z}_n\}$

How to estimate $\boldsymbol{\theta}$?

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

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

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

Parameter Estimation for GMMs: Incomplete Data

EM algorithm provides a strategy for iteratively solving this problem!

E(xpectation)-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 $r_{nk} = p(z_n = k|\boldsymbol{x}_n)$

- 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(aximization)-Step: Use the same expression as before to estimate $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ given soft r_{nk}:

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

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

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.

EM Algorithm: Setup

- Suppose the model is given by a joint distribution

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

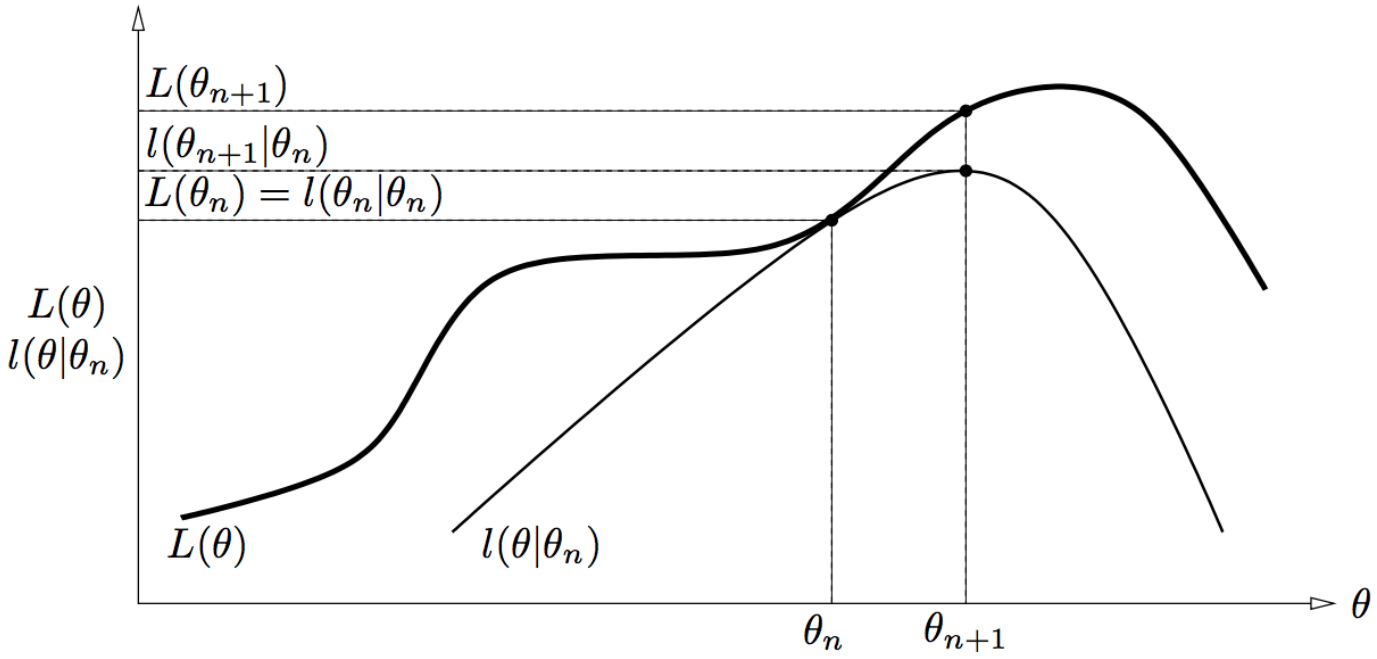
- Given incomplete data $\mathcal{D} = \{\boldsymbol{x}_n\}$ our goal is to compute MLE of $\boldsymbol{\theta}$:

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

- The objective function $\ell(\boldsymbol{\theta})$ is called *incomplete* log-likelihood
- log-sum form of incomplete log-likelihood is difficult to work with

EM: Principle

- EM: construct lower bound on $\ell(\boldsymbol{\theta})$ (E-step) and optimize it (M-step)
- "Majorization-minimization (MM)"
- Optimizing the lower bound will hopefully optimize $\ell(\boldsymbol{\theta})$ too.



(Figure from tutorial by Sean Borman)

E- and M-Steps

Our lower bound for the log-likelihood:

$$\ell(\boldsymbol{\theta}) \geq \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^t) \log \frac{p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})}{p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^t)}$$

$$\geq \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^t) \log p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta}) - \underbrace{\sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^t) \log p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^t)}_{\text{does not depend on } \boldsymbol{\theta}}$$

Therefore, it suffices to maximize the first term.

Why is this called the E-Step? Because we can view it as computing the *expected (complete) log-likelihood*:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^t) = \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^t) \underbrace{\log p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})}_{\text{complete log-likelihood}} = \mathbb{E}_{q_t} \sum_n \log p(\boldsymbol{x}_n,\boldsymbol{z}_n|\boldsymbol{\theta})$$

M-Step: Maximize $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^t)$, i.e., $\boldsymbol{\theta}^{t+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^t)$

Another Example: Multinomial Distributions

Suppose we are trying to model the number of people who will vote for one of four candidates. We know that the probability of a single person voting for each candidate is:

$$(p_1, p_2, p_3, p_4) = \left(\frac{1}{2} + \frac{\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4}\right).$$

Let $Y = (y_1, y_2, y_3, y_4)$ denote our observation of the number of votes for each candidate. How do we estimate θ?

- Option 1: MLE. But it is "hard" to optimize the log-likelihood...

$$\log p(Y|\theta) = y_1 \log\left(\frac{1}{2} + \frac{\theta}{4}\right) + (y_2 + y_3)\log(1-\theta) + y_4 \log\theta$$

- Option 2: EM. We will introduce two *unobserved/latent variables* x_0 and x_1, where $x_0 + x_1 = y_1$. In other words, y_1 combines the counts of two categories, whose individual counts are given by x_0 and x_1. We have a probability $\frac{1}{2}$ of picking the x_0 category, and a probability $\frac{\theta}{4}$ of picking the x_1 category.

Applying EM to Multinomial Distributions

Complete likelihood function:

$$\ell(\theta) = \log p(X, Y|\theta) = (x_1 + y_4)\log\theta + (y_2 + y_3)\log(1-\theta)$$

E-Step: Find $Q(\theta|\theta^t) = \mathbb{E}_{q_t}[\ell(\theta)]$, where q_t is the posterior distribution of x_1 given $y_1, y_2, y_3, y_4, \theta$. To do this, we need to find $\mathbb{E}_{q_t}[x_1]$:

$$x_1^{t+1} = \mathbb{E}_{q_t}[x_1] = y_1 \frac{\frac{\theta^t}{4}}{\frac{1}{2} + \frac{\theta^t}{4}}.$$

M-Step: Find θ that maximizes

$$Q(\theta|\theta^t) = \left(y_1 \frac{\frac{\theta^t}{4}}{\frac{1}{2}+\frac{\theta^t}{4}} + y_4\right)\log\theta + (y_2 + y_3)\log(1-\theta).$$

$$\theta^{t+1} = \frac{x_1^{t+1} + y_4}{x_1^{t+1} + y_4 + y_2 + y_3}$$

What Does This Look Like in Practice?

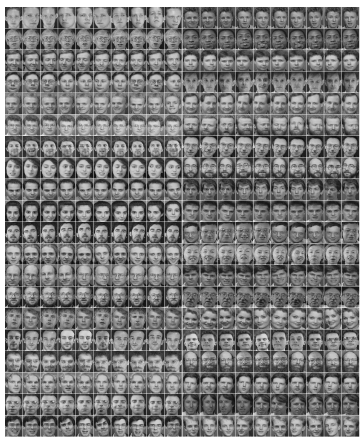
Observe $Y = (125, 18, 20, 34)$ and initialize $\theta^0 = 0.5$.

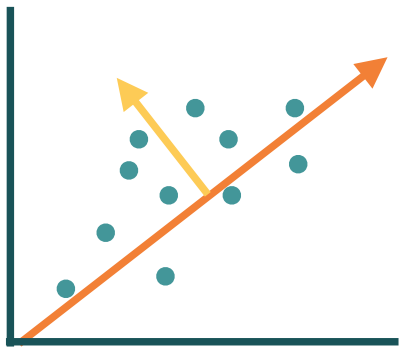
k	Parameter update $\theta^{(k)}$	Convergence to $\hat{\theta}$ $\theta^{(k)} - \hat{\theta}$	Convergence rate $(\theta^{(k)} - \hat{\theta})/(\theta^{(k-1)} - \hat{\theta})$
0	.500000000	.126821498	
1	.608247423	.018574075	.1465
2	.624321051	.002500447	.1346
3	.626488879	.000332619	.1330
4	.626777323	.000044176	.1328
5	.626815632	.000005866	.1328
6	.626820719	.000000779	.1328
7	.626821395	.000000104	
8	.626821484	.000000014	
$\hat{\theta}$	.626821498	Stop	

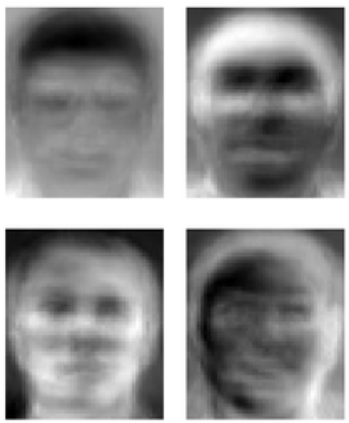
Principal Component Analysis (PCA)

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
- If we knew what to measure or how to represent our measurements, we might end up with a small dimensional dataset.
- 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, which might not be the best way to represent it.

Dimensionality Reduction

Issues

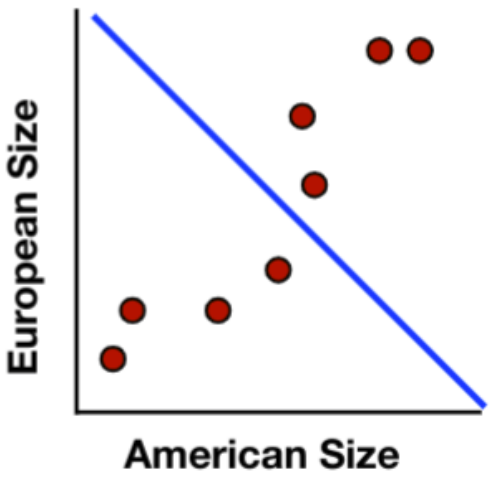
1. Measure redundant signals.
2. Represent data via the method by which it was gathered, which might not be a good representation.

Goal:
Find a "better" representation for data

1. To discover hidden patterns and to reduce dimension
2. This serves as a preprocessing step for the later supervised task

How do we define "better"?

Dimensionality Reduction (E.g., Shoe Size)



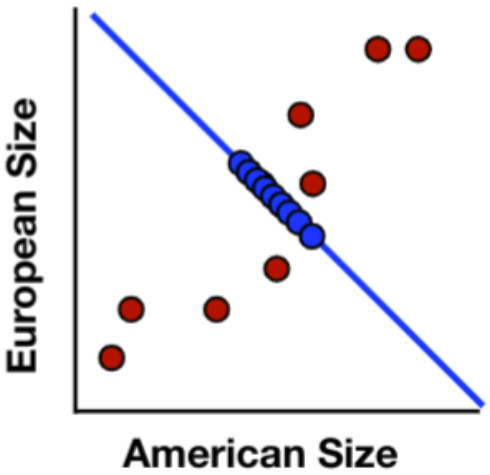
We take noisy measurements on the European and American scales

- 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 (E.g., Shoe Size)



We take noisy measurements on the European and American scales

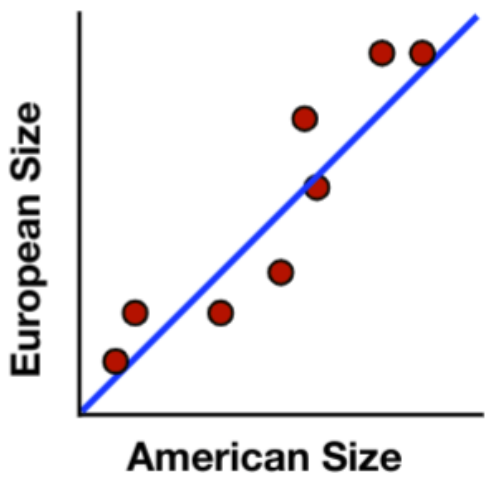
- Modulo noise, we expect perfect correlation

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 (E.g., Shoe Size)



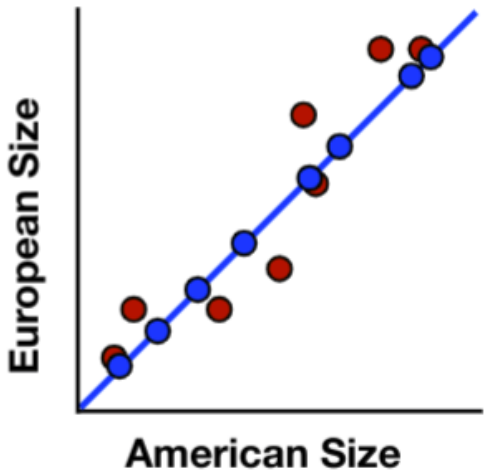
We take noisy measurements on the European and American scales

- 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 (E.g., Shoe Size)



We take noisy measurements on the European and American scales

- Modulo noise, we expect perfect correlation

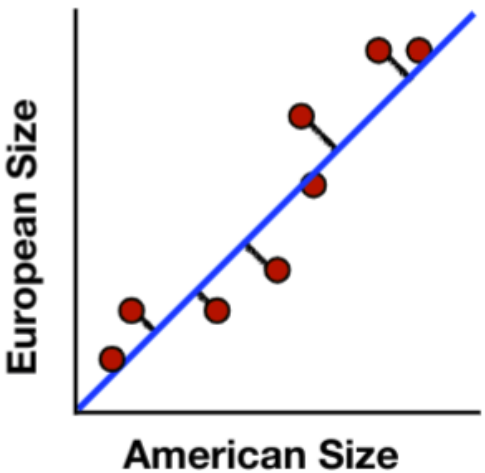
How can we do better, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction

Goal: Minimize Reconstruction Error

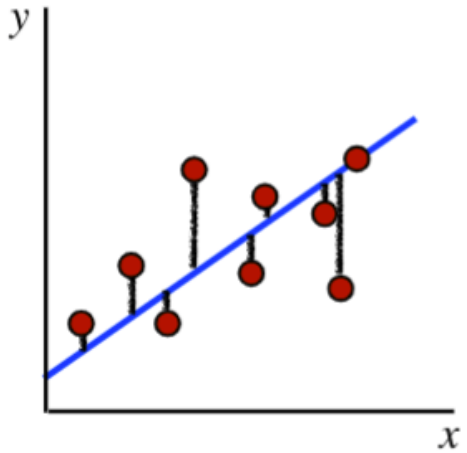
PCA

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



Linear Regression

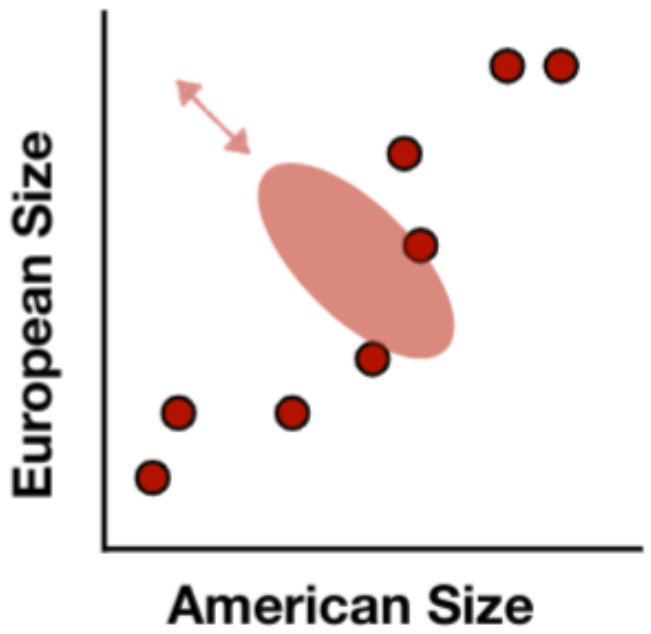
Predict y from x. Evaluate predictions (represented by blue line) by vertical distances between points and the line.



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

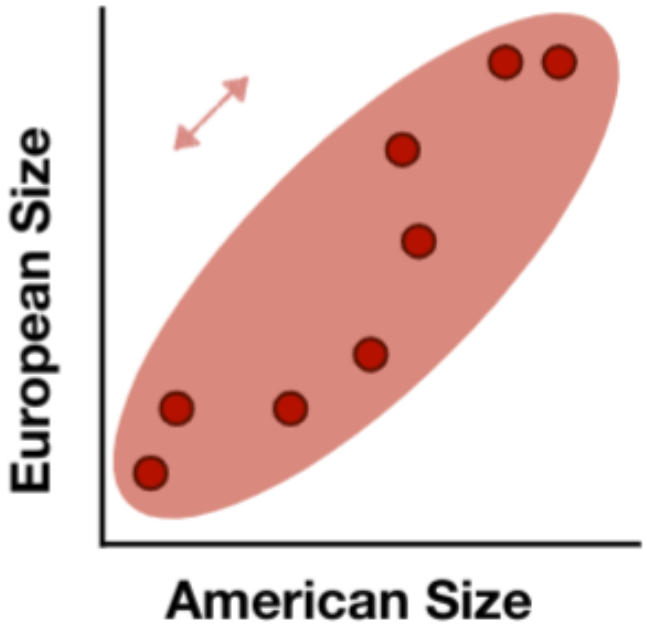
Goal: Maximize Variance

- For PCA, to identify the right direction to project to, we study the variation of the data in different directions.
- Is the direction on the right a good direction? Does it capture the variation in the data?



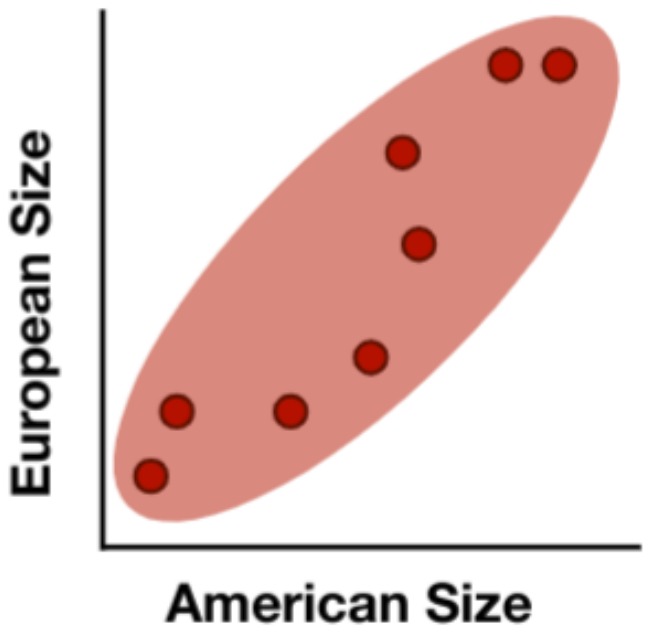
Goal: Maximize Variance

- For PCA, to identify the right direction to project to, we study the variation of the data in different directions.
- Is the direction on the right a good direction? Does it capture the variation in the data?



Goal: Maximize Variance

- For PCA, to identify the right direction to project to, we study the variation of the data in different directions.
- Is the direction on the right a good direction? Does it capture the variation in the data?
- PCA solution finds directions of maximal variance.



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)

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

  - Intuitively, each column of P is a direction we project to.
  - Each row of Z is the coordinates of the reduced representation under the frame defined by the column vectors in P.
- How do we design the matrix $\mathbf{P}$? The k directions should be the "high variance" directions!

Zero-Center All the Features

For the convenience of calculating variance, it is useful to 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

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

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

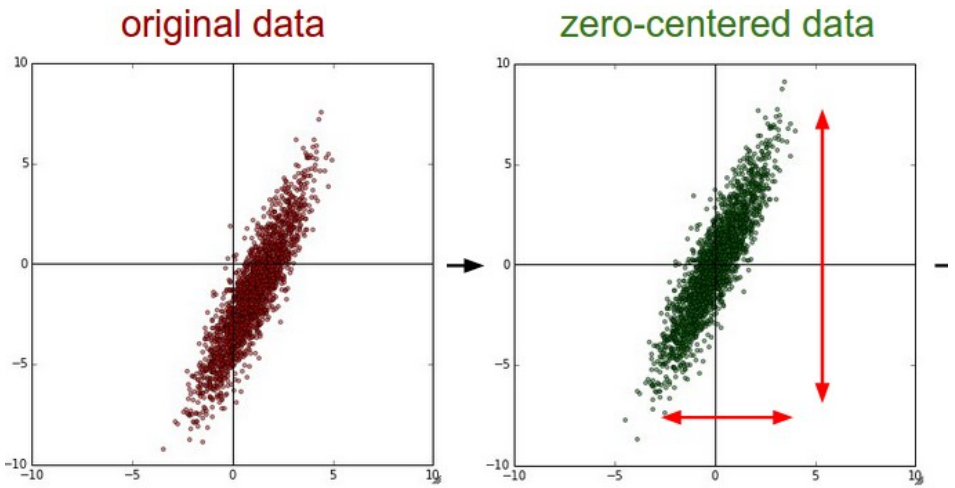
$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{x}_1 - \mu_1 & \dots & \mathbf{x}_d - \mu_d \\ | & | & | \end{bmatrix} \begin{bmatrix} \\ \mathbf{P} \\ \\ \end{bmatrix}$$

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

Zero-Center All the Features

Given n training points with d features:

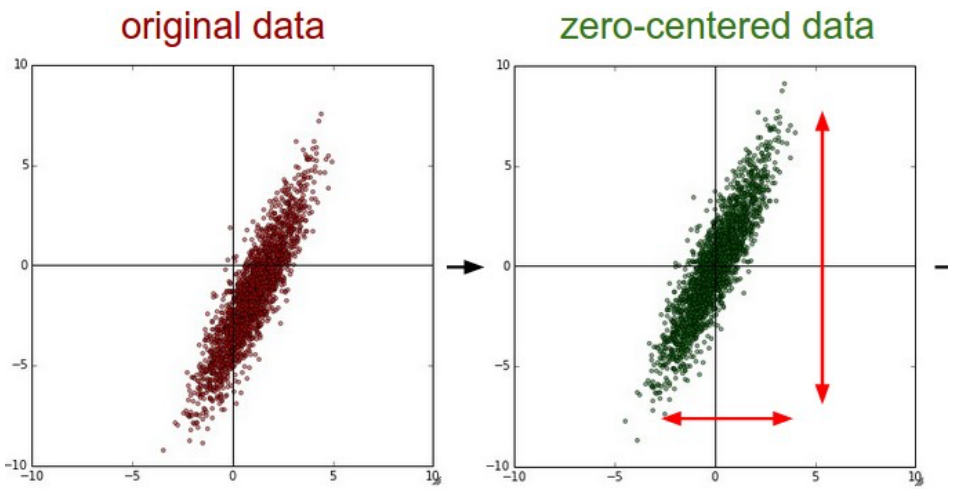
- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $\mathbf{x}_j^{(i)}$: j^{th} feature vector for i^{th} point
- μ_j: mean of j^{th} feature
- Deduct μ_j from all features



Variance and Co-variance of the Features

Given n training points with d features:

- Variance of 1st (zero-centered) feature is $\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_1^{(i)})^2$
- 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: $\sigma_{12} = \sigma_{21}$
- 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
- For zero-centered features $\mathbf{X}$, the covariance matrix is a $d \times d$ matrix

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

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

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)

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

We want P to be the k directions with the largest variance, i.e. the $\mathbf{C_Z}$ should be the "largest".

How to identify those k directions? Eigenvalue decompostion!

Eigen-value Decomposition

The covariance matrix $\mathbf{C_X}$ is symmetric, positive semi-definite and therefore has an eigenvalue 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 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$.
  - λ_1 is the top eivenvalue.
- $\mathbf{Q}$ is an matrix of d orthonormal eigenvectors $\mathbf{v}_i$ of $\mathbf{C_X}$.
  - $\mathbf{v}_1$ is the top eigenvector.
  - $\mathbf{Q}$'s columns are orthonormal, meaning the columns have unit length and are orthogonal to each other. This implies $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$.
  - $\mathbf{Q}$ is norm preserving, i.e. $\|\mathbf{Q}^T\mathbf{p}\|_2 = \|\mathbf{p}\|_2$ for any vector $\mathbf{p}$.

PCA Solution for $k = 1$

Consider PCA with $k = 1$

- P is $d \times 1$, a column vector which we denote as $\mathbf{p}$.
- $\mathbf{Z} = \mathbf{X}\mathbf{p}$ is $n \times 1$ and $\mathbf{C_Z}$ will be a 1×1, i.e. the variance in $\mathbf{Z}$

$$\mathbf{C_Z} = \mathbf{p}^T \mathbf{C_X} \mathbf{p} = \mathbf{p}^T \mathbf{Q} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} \mathbf{Q}^T \mathbf{p}$$

- Goal: $\max_{\mathbf{p}} \mathbf{C_Z}$ s.t. $\|\mathbf{p}\|_2 = 1$ (find direction with max. variance)
- Solution: Substitute $\mathbf{y} = \mathbf{Q}^T \mathbf{p}$, then $\|\mathbf{y}\|_2 = \|\mathbf{p}\|_2 = 1$. Then,

$$\mathbf{C_Z} = \mathbf{y}^T \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} \mathbf{y} = y_1^2 \lambda_1 + y_2^2 \lambda_2 + \ldots + y_d^2 \lambda_d$$

- Note $y_1^2 + \ldots + y_d^2 = 1$, and λ_1 is the largest eigenvalue.
- Max variance achieved with $\mathbf{y} = [1, 0, \ldots, 0]^T$, i.e. $\mathbf{p} = \mathbf{Q}\mathbf{y} = \mathbf{v}_1$, the top eigenvector of $\mathbf{C_X}$.

PCA Solution

- For the general k case, recall the covariance matrix of $\mathbf{Z}$ is

$$\mathbf{C_Z} = \mathbf{P}^T \mathbf{C_X} \mathbf{P} = \mathbf{P}^T \mathbf{Q} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} \mathbf{Q}^T \mathbf{P}$$

- How to find the k directions s.t. the variance in $\mathbf{Z}$ is the largest? Since $\mathbf{C_Z}$ is a matrix, the "largest" is measured using the matrix trace, i.e. sum of the diagonal entries of $\mathbf{C_Z}$.
- Solution: Choose $\mathbf{P}$ as the first k columns of $\mathbf{Q}$,

$$\mathbf{P} = \begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \dots & \mathbf{v}_k \\ | & | & | \end{bmatrix}$$

  i.e. the eigenvectors of the top k eigenvalues.
- This captures the k directions of maximum variance. Also, these k directions are orthogonal to each other.

PCA: Example

Suppose the covariance of $X \in \mathbb{R}^{n \times 2}$ 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 $\mathbf{P} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$.

A data point $x = \begin{bmatrix} 1 & 0 \end{bmatrix}$ is then represented by $x\mathbf{P} = \begin{bmatrix} \frac{1}{\sqrt{2}} \end{bmatrix}$.

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

- Can be shown the 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%

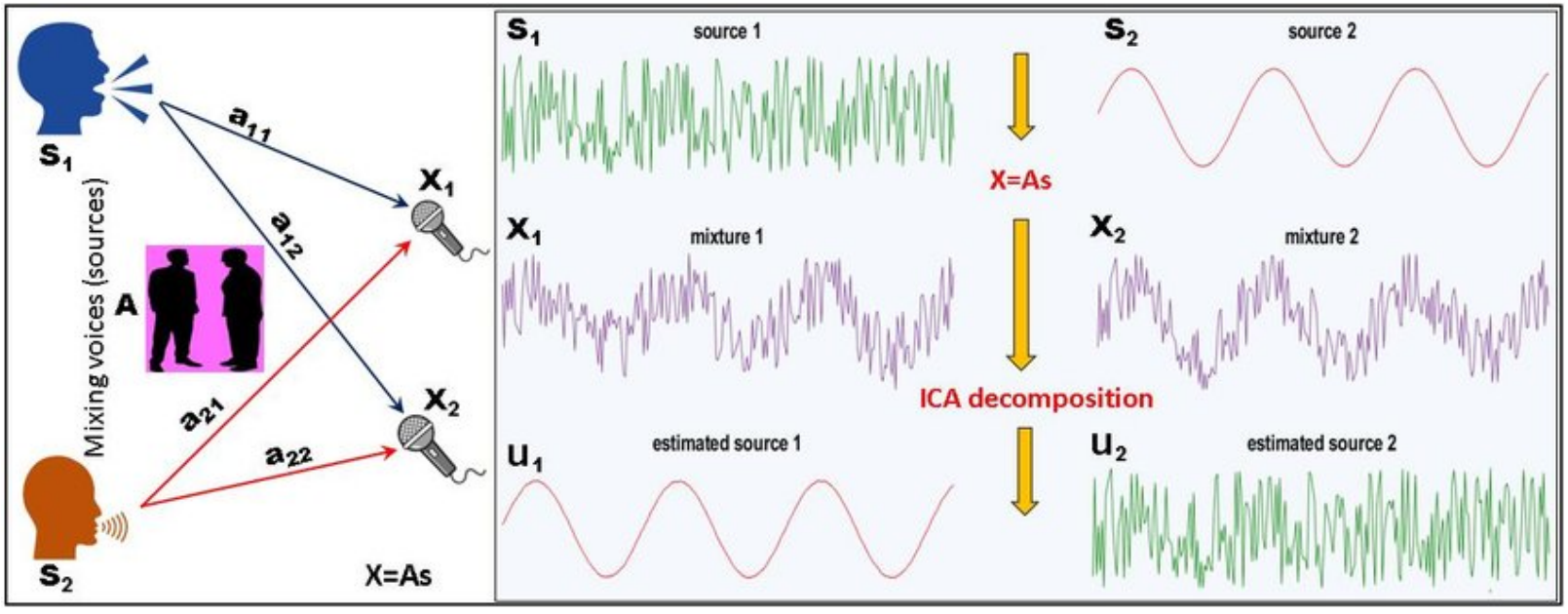
Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

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

A Glimpse at Independent Component Analysis (ICA)

The Cocktail Party Problem

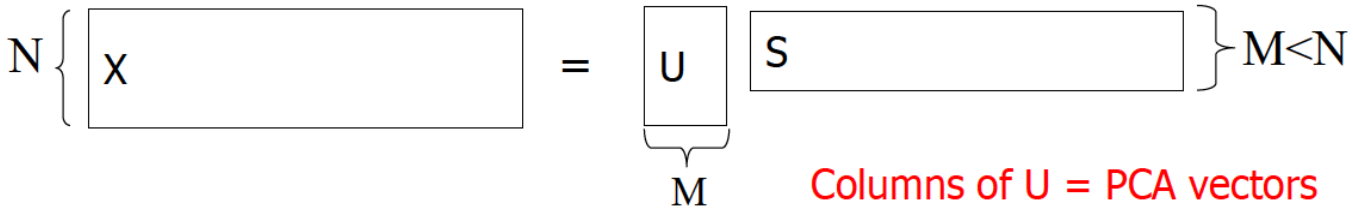


- PCA tries to find an orthogonal representation of the mixed data.
- ICA tries to disentangle the data sources.

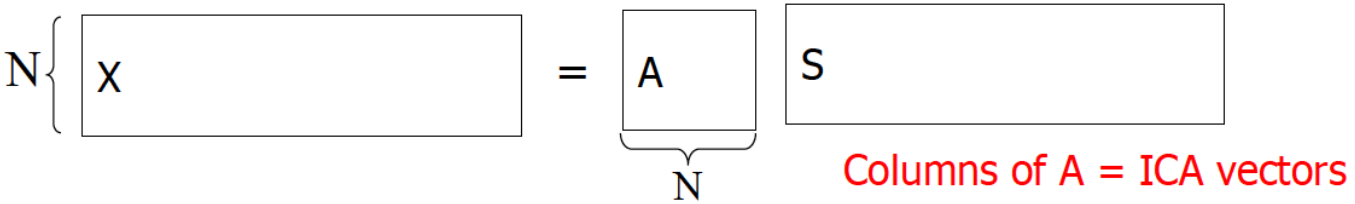
How Does ICA Work?

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

PCA compresses data with low-rank matrix factorization



ICA removes dependencies between data with full-rank matrix factorization



ICA Formulation

- $\mathbf{X}$ is $d \times n$ (raw data, each column is a data point)
- $\mathbf{S}$ is $d \times n$ (hidden sources of the data)
- $\mathbf{A}$ is $d \times d$ (mixing matrix)

$$\mathbf{X} = \mathbf{AS} \Rightarrow \mathbf{S} = \mathbf{A}^{-1}\mathbf{X}$$

- How do we find the hidden sources $\mathbf{S}$ (and mixing matrix $\mathbf{A}$)?
  - Intuitively, each observed feature vector in $\mathbf{X}$ is a mixture of the hidden sources in $\mathbf{S}$
  - Can only find $\mathbf{S}$ and $\mathbf{A}$ up to scaling and permutation
  - Enforce independence among the sources (rows of $\mathbf{S}$)
  - Source data must be non-Gaussian (otherwise the observed data is symmetric for orthogonal $\mathbf{A}$)

ICA Optimization

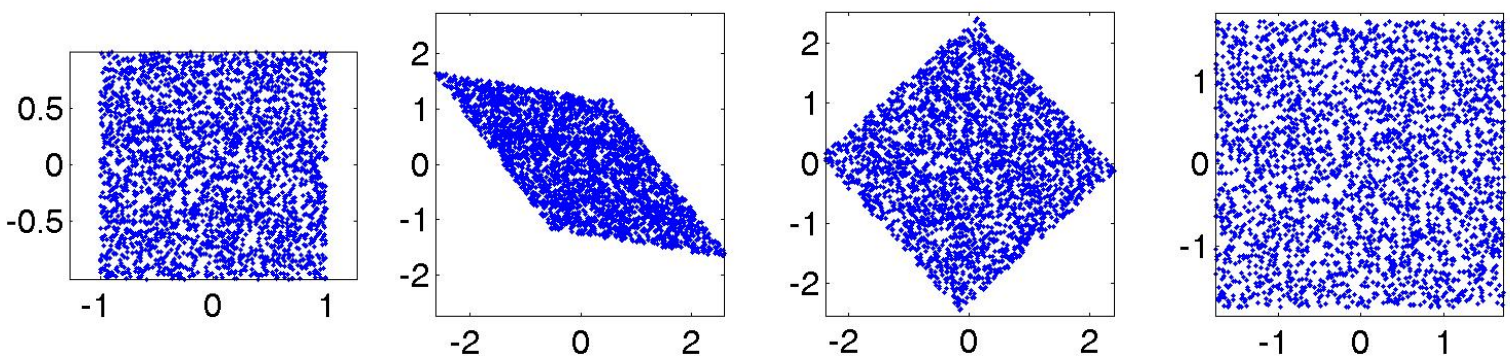
Maximize **non-Gaussianity**, or equivalently, minimize **mutual information**

- Zero-center and whiten data before applying ICA algorithms

$$\mathbb{E}[\mathbf{x}] = 0, \quad \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] = I$$

- Minimize mutual information (recall H denotes the entropy function)

$$\min_{\mathbf{s}=\mathbf{A}^{-1}\mathbf{x}} \quad \sum_{i=1}^{d} H(s_i) - H(\mathbf{x}) - \log\left|\det \mathbf{A}^{-1}\right|$$



Comparing PCA and ICA

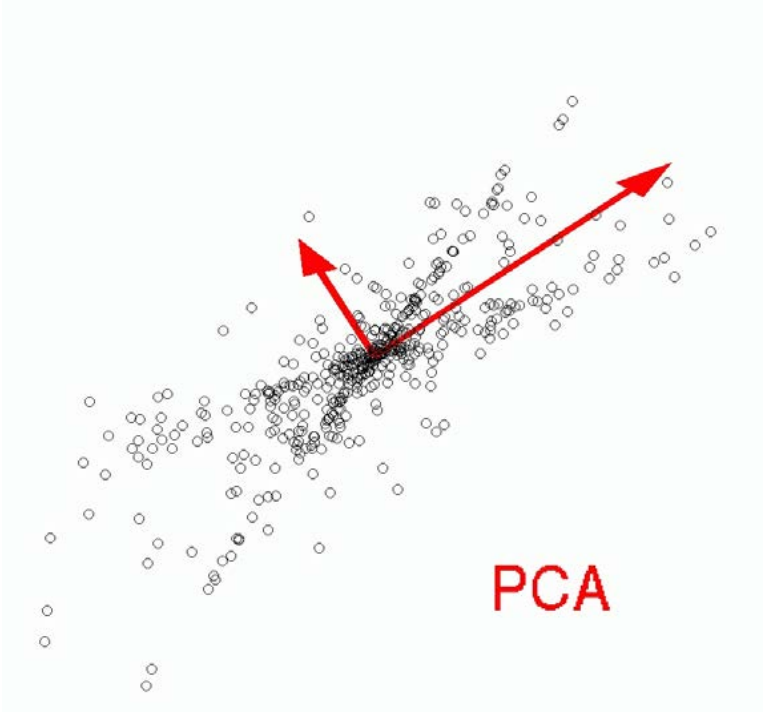
Let $\mathbf{S}$ denote our original data. It can be transformed to a new set of features $\mathbf{X}$:

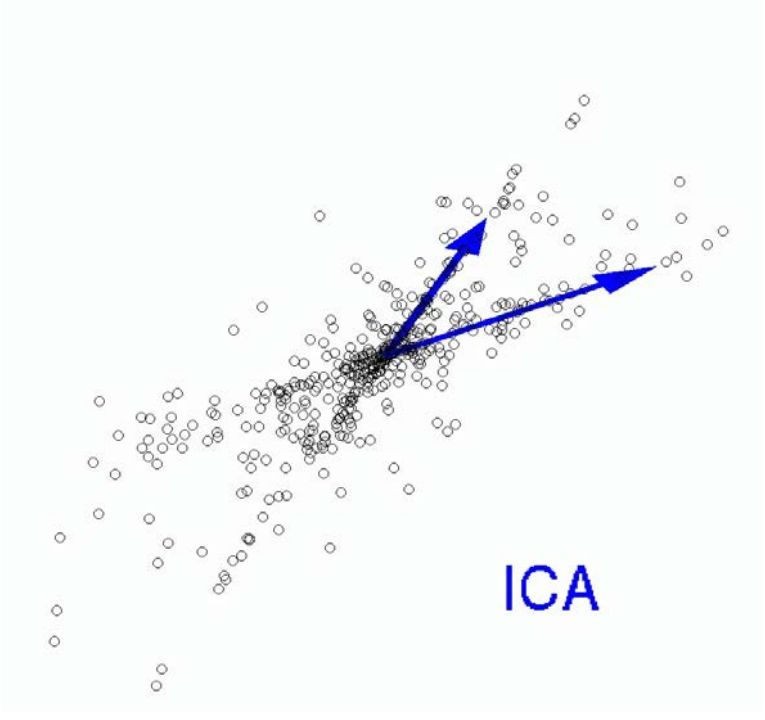
$$\text{PCA} : \mathbf{X} \approx \mathbf{US},\ \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

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

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