18-661 Introduction to Machine Learning

Dimensionality Reduction

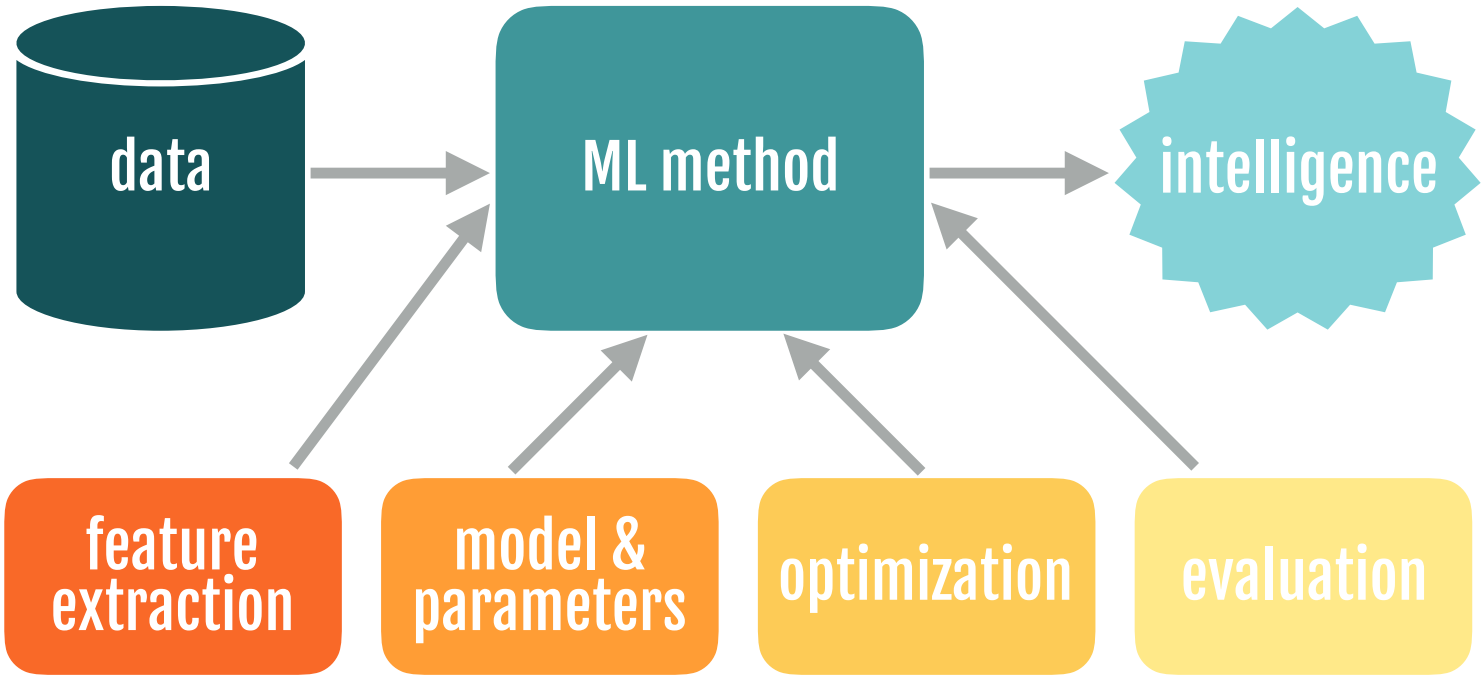
Fall 2020

ECE – Carnegie Mellon University

Announcements

- HW 6 is due on Sunday, April 12.
- HW 7 will be a Python homework, due April 23.
  - There will be one mandatory problem on PyTorch for deep neural networks, and two optional problems on decision trees and clustering.
  - 661 students (most of you) must do one of the two optional problems. If you choose to do both problems, we will take the best of the two in computing your grade.
  - 461 students only need to do the mandatory problem.
  - Attend the April 13 lecture if you are unfamiliar with PyTorch.
- Recitation on Friday will cover ICA (independent component analysis), a variant of some dimensionality reduction techniques that we will introduce today.

Goal: Learn about ML Pipeline



Outline

1. EM Recap
2. Principal Component Analysis (PCA)
3. Sparsity
4. Independent Component Analysis (ICA)

EM Recap

EM algorithm: Motivation

EM is a general procedure to estimate parameters for probabilistic models with hidden/latent variables.

Maximum Likelihood from Incomplete Data via the *EM* Algorithm

By A. P. Dempster, N. M. Laird and D. B. Rubin

Harvard University and Educational Testing Service

[Read before the Royal Statistical Society at a meeting organized by the Research Section on Wednesday, December 8th, 1976, Professor S. D. Silvey in the Chair]

Summary

A broadly applicable algorithm for computing maximum likelihood estimates from incomplete data is presented at various levels of generality. Theory showing the monotone behaviour of the likelihood and convergence of the algorithm is derived. Many examples are sketched, including missing value situations, applications to grouped, censored or truncated data, finite mixture models, variance component estimation, hyperparameter estimation, iteratively reweighted least squares and factor analysis.

Maximum Likelihood from Incomplete Data Via the *EM* Algorithm

AP Dempster, NM Laird… - Journal of the Royal …, 1977 - Wiley Online Library

A broadly applicable algorithm for computing maximum likelihood estimates from incomplete data is presented at various levels of generality. Theory showing the monotone behaviour of the likelihood and convergence of the algorithm is derived. Many examples are sketched, including missing value situations, applications to grouped, censored or truncated data, finite mixture models, variance component estimation, hyperparameter estimation, iteratively reweighted least squares and factor analysis.

☆ Cited by 56239 Related articles All 59 versions Web of Science: 25101

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 $\boldsymbol{\theta}$:

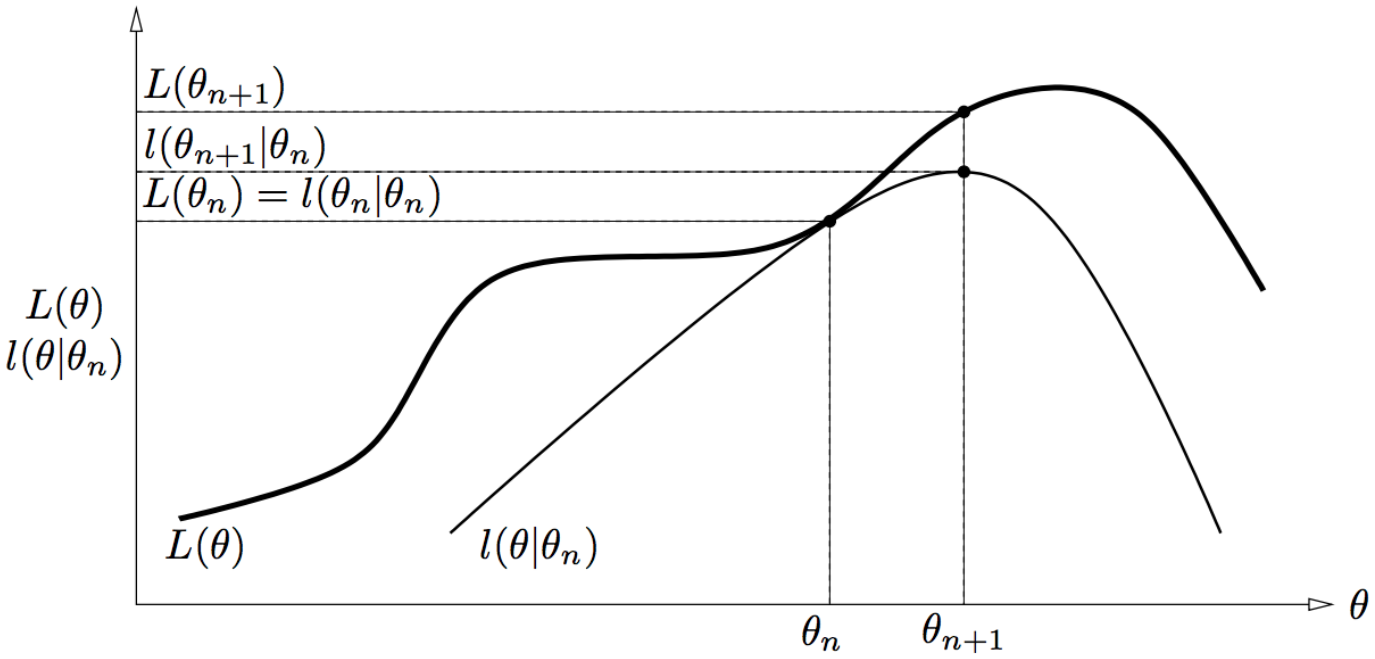
$$\boldsymbol{\theta} = \arg\max \ell(\boldsymbol{\theta}) = \arg\max \log p(\mathcal{D}) = \arg\max \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

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

- 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)

Constructing a lower bound

If we define $q(\boldsymbol{z})$ as a distribution over $\boldsymbol{z}$, then

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}) \\ &= \sum_n \underbrace{\log \sum_{\boldsymbol{z}_n} q(\boldsymbol{z}_n) \frac{p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})}{q(\boldsymbol{z}_n)}}_{f\left(\mathbb{E}_{q(\boldsymbol{z}_n)}\left[\frac{p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})}{q(\boldsymbol{z}_n)}\right]\right)}\end{aligned}$$

Apply Jensen's inequality to each term, i.e., $f(\mathbb{E}X) \geq \mathbb{E}f(X)$, for concave function $f(\cdot) = \log(\cdot)$. We take the expectation of $X = \frac{p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})}{q(\boldsymbol{z}_n)}$, a random variable depending on $\boldsymbol{z}_n$, over the probability distribution $q(\boldsymbol{z}_n)$.

$$\ell(\boldsymbol{\theta}) \geq \sum_n \underbrace{\sum_{\boldsymbol{z}_n} q(\boldsymbol{z}_n) \log \frac{p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})}{q(\boldsymbol{z}_n)}}_{\mathbb{E}_{q(\boldsymbol{z}_n)}\left[f\left(\frac{p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})}{q(\boldsymbol{z}_n)}\right)\right]}$$

Pick $q_t(z_n)$

Pick $q_t(z_n)$ so that

$$\ell(\theta^t) = \sum_n \log p(x_n, z_n|\theta^t) = \sum_n \sum_{z_n} q_t(z_n) \log \frac{p(x_n, z_n|\theta^t)}{q_t(z_n)}$$

- Pick the distribution where the equality in Jensen's inequality holds.
- Choose $q_t(z_n) \propto p(x_n, z_n|\theta^t)$!
- Since $q_t(\cdot)$ is a distribution, we have

$$q_t(z_n) = \frac{p(x_n, z_n|\theta^t)}{\sum_k p(x_n, z_n = k|\theta^t)} = \frac{p(x_n, z_n|\theta^t)}{p(x_n|\theta^t)} = p(z_n|x_n; \theta^t)$$

- This is the posterior distribution of z_n given x_n and θ^t

E- and M-Steps

Our lower bound for the log-likelihood:

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

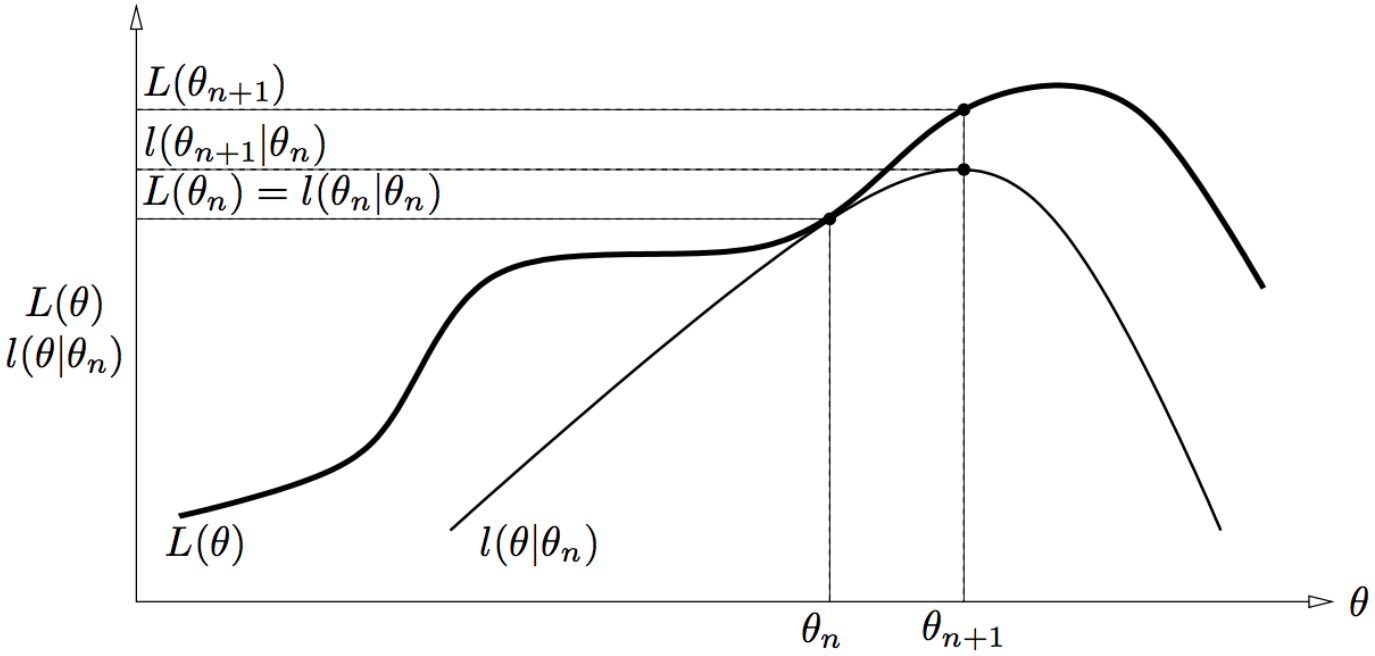
Why is this called the E-Step? Because we can view it as computing the *expected (complete) log-likelihood*:

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

Where did the $p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^t)$ in the denominator go? It is not a function of $\boldsymbol{\theta}$, so can be ignored.

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

EM in Pictures

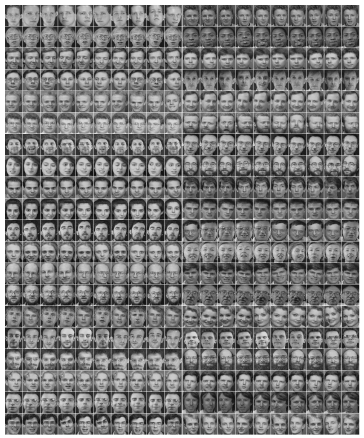


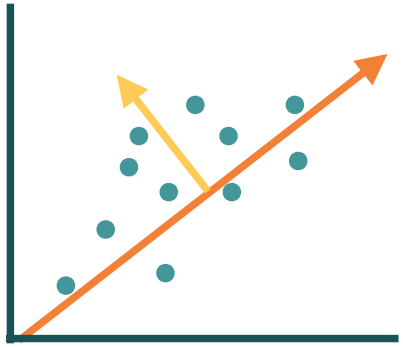
(Figure from tutorial by Sean Borman)

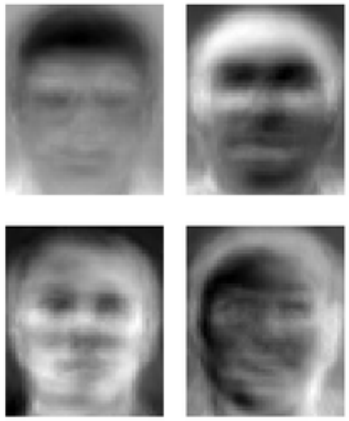
Principal Component Analysis (PCA)

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 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

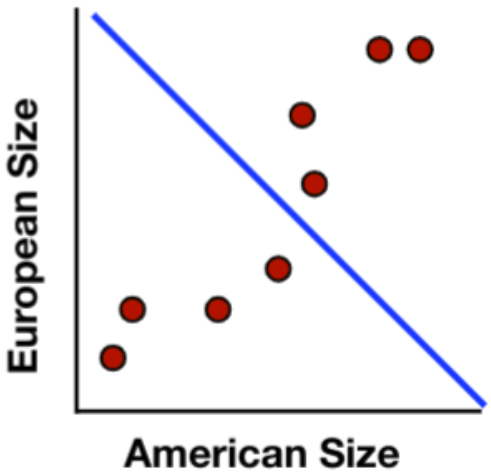
Goal:

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)



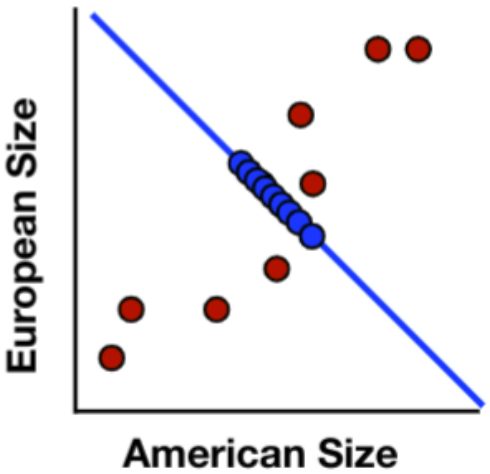
We take noisy measurements on the European and American scales

- Modulo noise, we expect perfect correlation

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

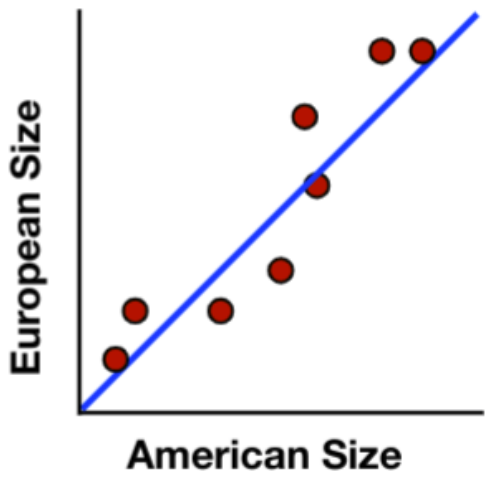
- Modulo noise, we expect perfect correlation

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)



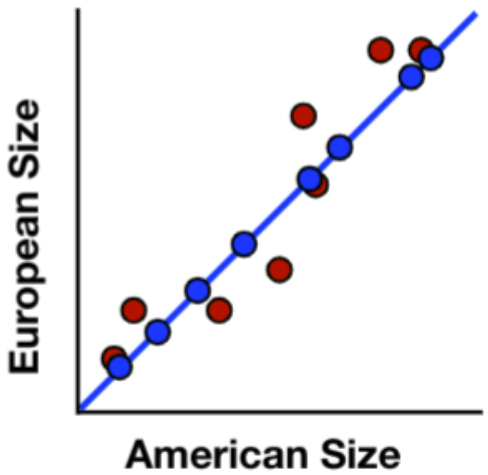
We take noisy measurements on the European and American scales

- 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)



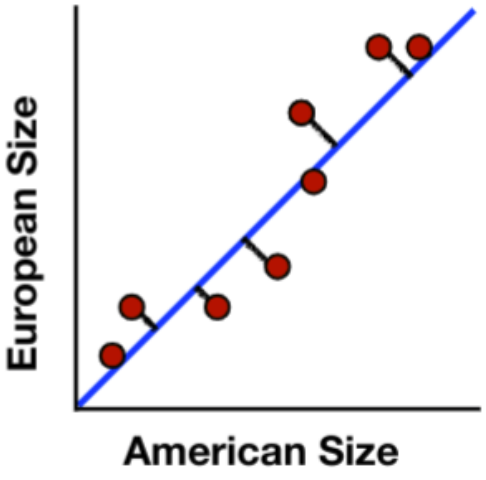
We take noisy measurements on the European and American scales

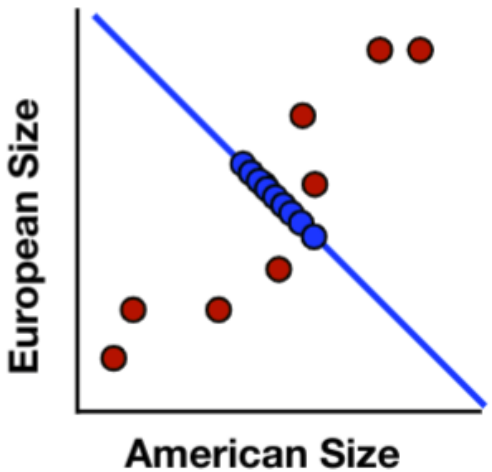
- Modulo noise, we expect perfect correlation

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

- Pick a direction and project onto this direction

Goal: Minimize Reconstruction Error





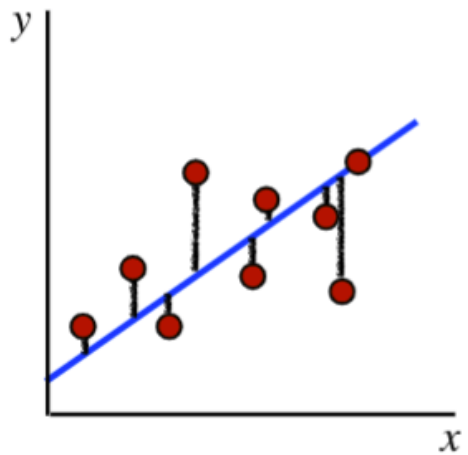
Minimize Euclidean distances between original points and their projections

PCA solution solves this problem!

Goal: Minimize Reconstruction Error

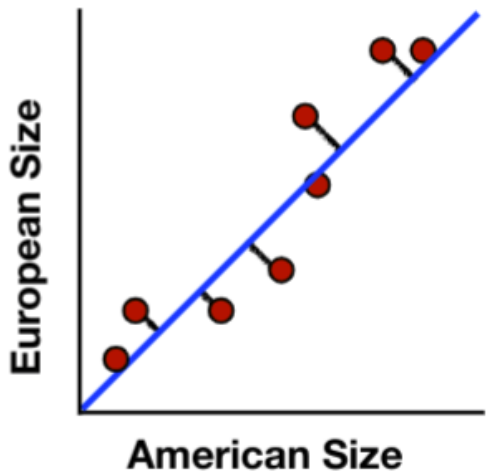
Linear Regression

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



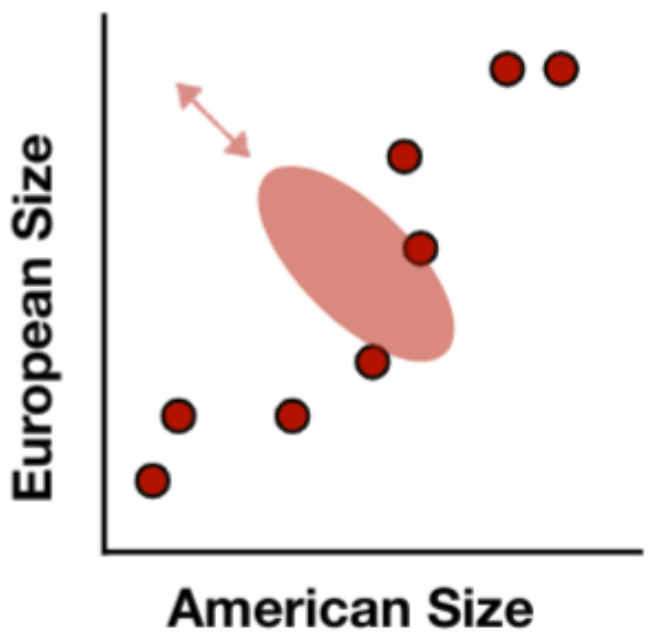
PCA

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



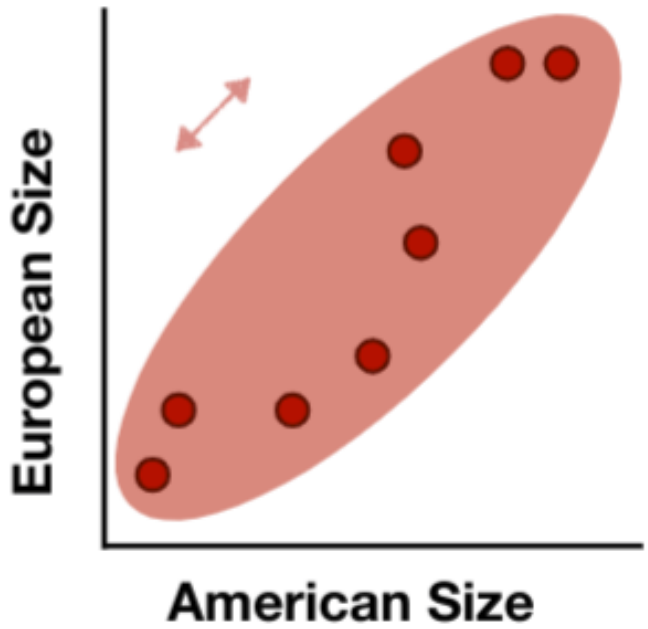
Goal: Maximize Variance and Minimize Feature-Correlation

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



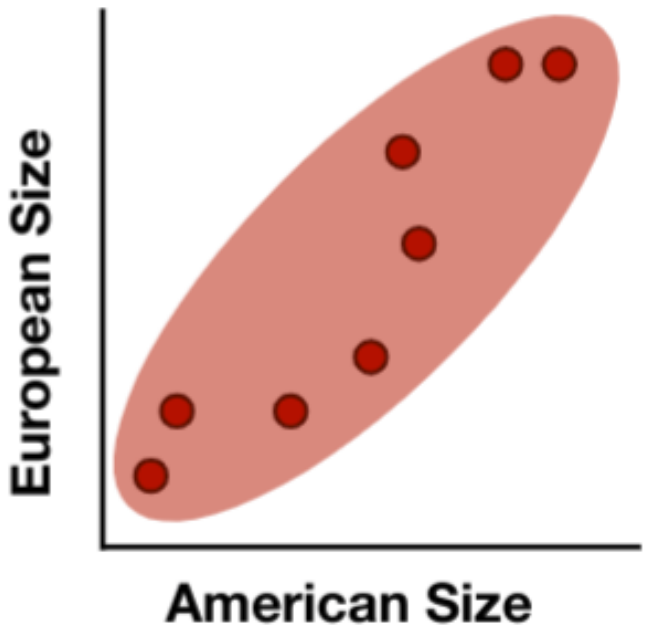
Goal: Maximize Variance and Minimize Feature-Correlation

- 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

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

- 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

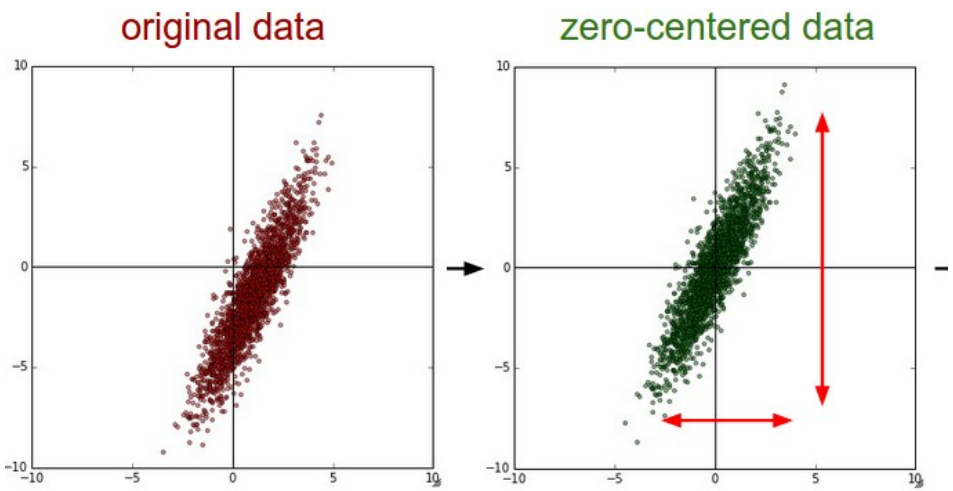
$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{x}_1 - \mu_1 & \dots & \mathbf{x}_d - \mu_d \\ | & | & | \end{bmatrix} \begin{bmatrix} \\ \mathbf{P} \\ \\ \end{bmatrix}$$

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:

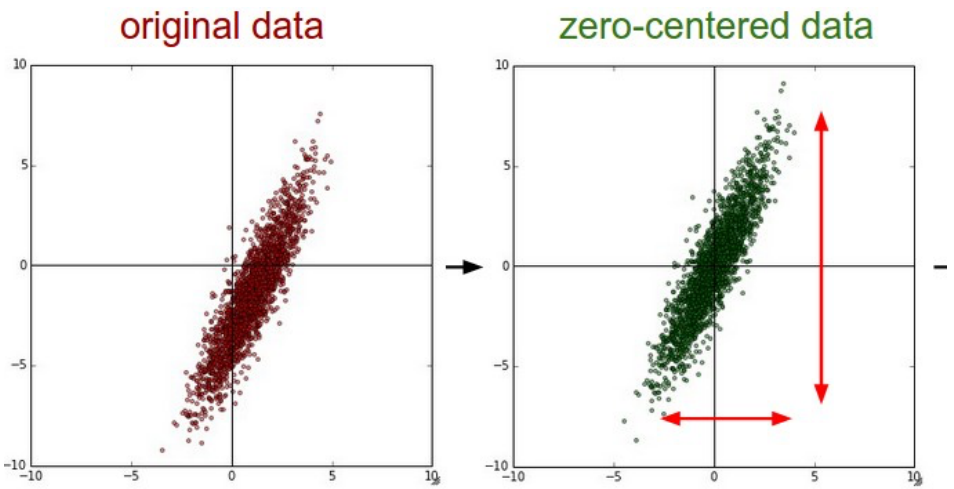
- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $\mathbf{x}_j^{(i)}$: j^{th} feature vector for i^{th} point
- μ_j: mean of j^{th} feature
- Deduct μ_j from all features



Variance and Co-variance of the Features

Given n training points with d features:

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

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 co-variance 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

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 co-variance 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 $\mathbf{C_Z} = \mathbf{\Lambda}$? 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 $k = 1$.

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?

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
- 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

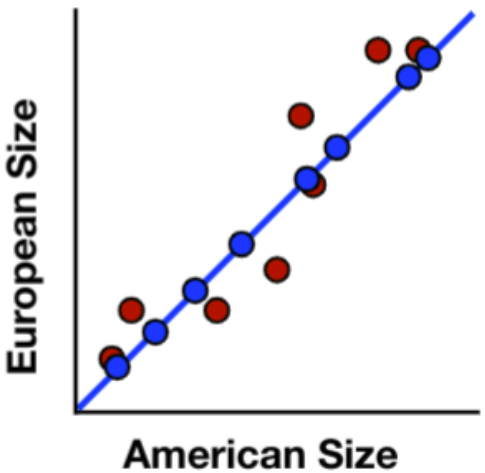
- For high-dimensional original features (large d) computing the eigenvalue decomposition of $\mathbf{C}_X$, and sorting the eigenvalues can be intractable.

Iterative 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 $\mathbf{C_X} = \mathbf{X}^T\mathbf{X}$
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)

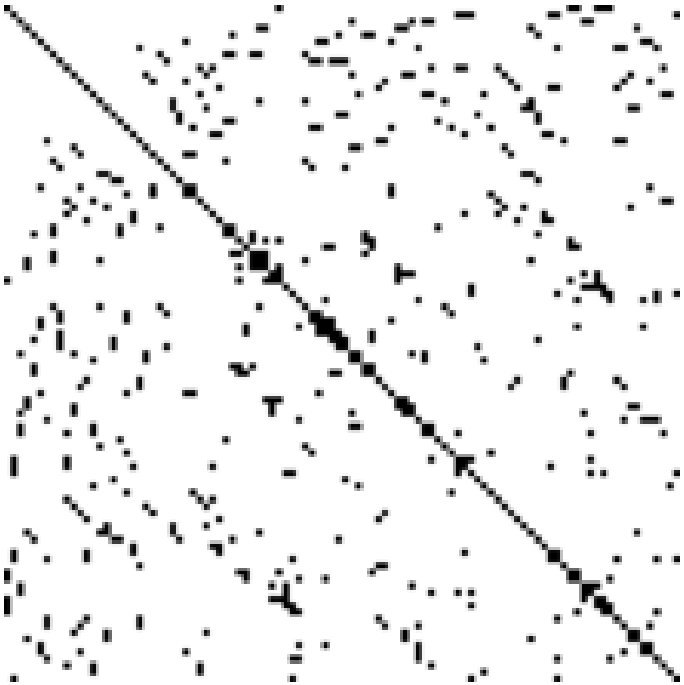
$$\begin{aligned}\sigma_{\mathbf{z}}^2 &= \frac{1}{n}||\mathbf{z}||_2^2 \\ &= \frac{1}{n}\mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p} \\ &= \frac{1}{n}\mathbf{p}^T\mathbf{C}_{\mathbf{X}}\mathbf{p}\end{aligned}$$

- 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 $\mathbf{C}_{\mathbf{X}}$

Sparsity

What Is Sparsity?

A sparse matrix is one in which most of the entries are zero.



$$\text{Sparsity score} = \frac{\#\text{ of zero entries}}{\text{total }\#\text{ of entries}}$$

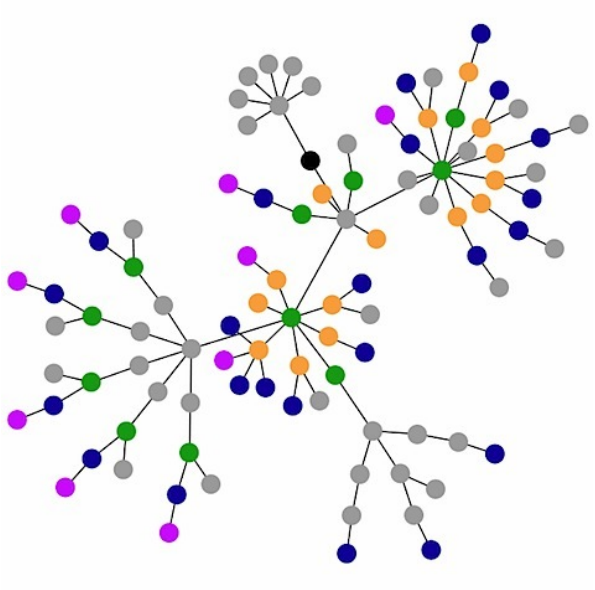
- Set a threshold for "sparse."
- Black elements at left are nonzero.

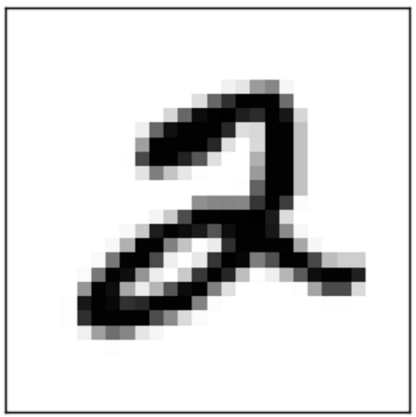
Where Does Sparse Data Occur?

Some typical examples:

- Connections in a social network.
- Sensor data–zero unless an event happens.
- One-hot encoding.
- Images with black backgrounds.

Relatively common in ML applications!





Working with Sparse Data

Traditional algorithms are very inefficient on sparse data–you need to push around a lot of zeros.

More efficient data storage:

- Only store indices and nonzero entries, instead of an entire two-dimensional array.
- Dictionary of keys, list of lists, ...

Easier matrix computations:

- Fast matrix multiplication, since we can ignore zero entries.
- Useful for gradient descent, computing distances, ...

Sparse PCA

PCA's features are often linear combinations of *all* the input features. Sparse PCA aims to find sparse linear combinations that also maximize the variance (just like in normal PCA).

If we take $k = 1$ (find the dominant feature):

$$\max_{\mathbf{p}} \mathbf{p}^T \mathbf{C_X} \mathbf{p} \tag{1}$$

$$\text{s.t. } \|\mathbf{p}\|_2 = 1, \ \|\mathbf{p}\|_0 \leq p. \tag{2}$$

Here p, $1 \leq p \leq d$ is a parameter setting the maximum sparsity and $\|\cdot\|_0$ counts the number of nonzero entries in a vector (this is the 0-norm).

This problem is non-convex and difficult to solve...

Solving for the Sparse Features

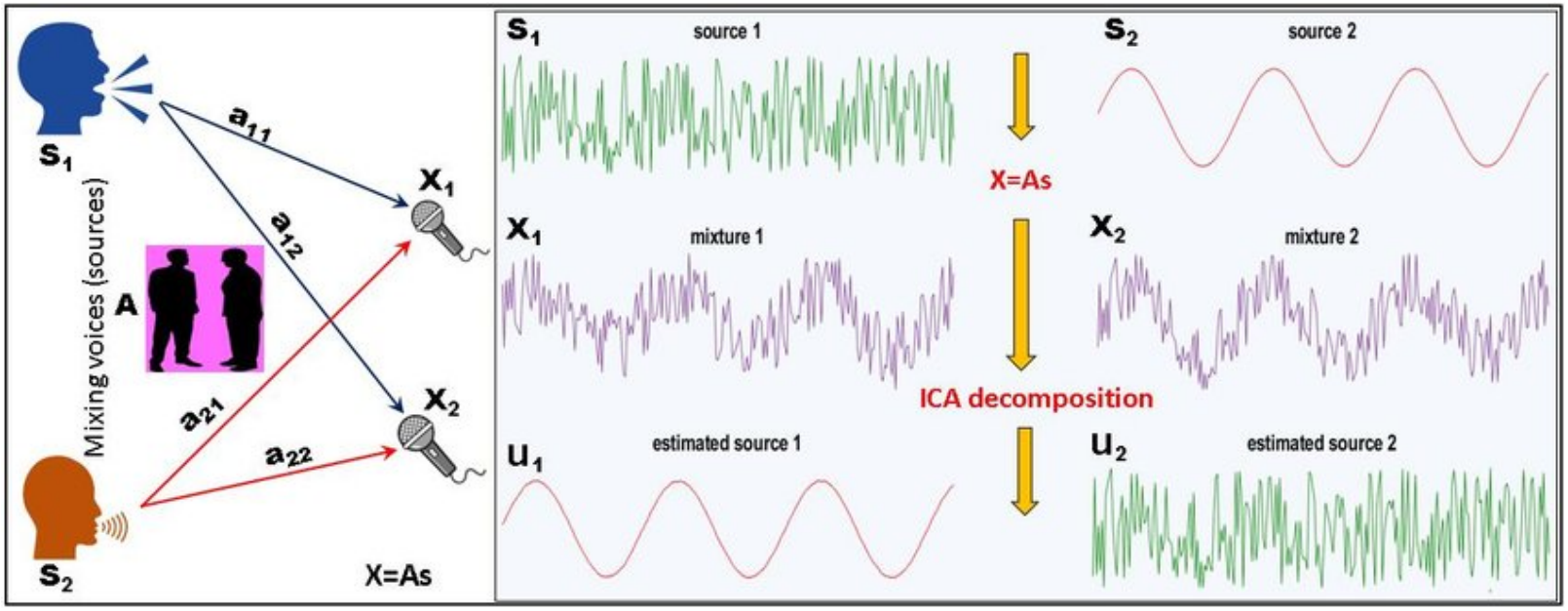
Similar algorithm as for standard PCA.

- Solve for the first principal component:
  $\arg\max_{\mathbf{p}} \ \mathbf{p}^T \mathbf{C_X} \mathbf{p} \text{ s.t. } \|\mathbf{p}\|_2 = 1, \ \|\mathbf{p}\|_0 \leq p.$
- Deflate the covariance matrix: $\mathbf{C_1} \leftarrow \mathbf{C_X} - \left(\mathbf{p}^T \mathbf{C_X} \mathbf{p}\right) \mathbf{p}\mathbf{p}^T.$
- Solve for the next principal component and repeat.

The sparse features are not guaranteed to be orthogonal to each other. When $p = d$, sparse PCA reduces to normal PCA (since up to d features can contribute to the PCA features).

Independent Component Analysis (ICA)

The Cocktail Party Problem

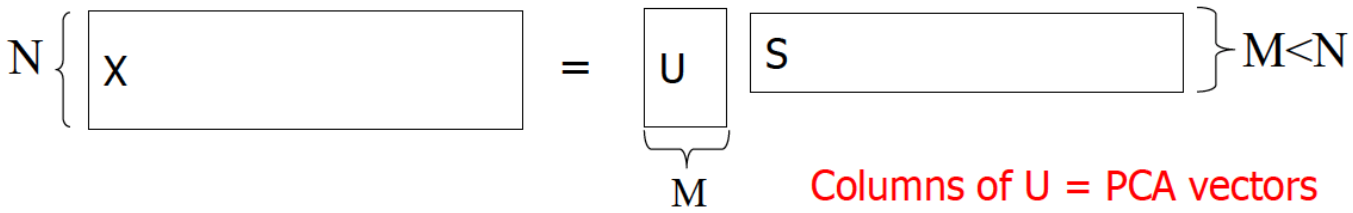


- PCA tries to find an orthogonal representation of the mixed data.
- ICA tries to disentangle the data sources.

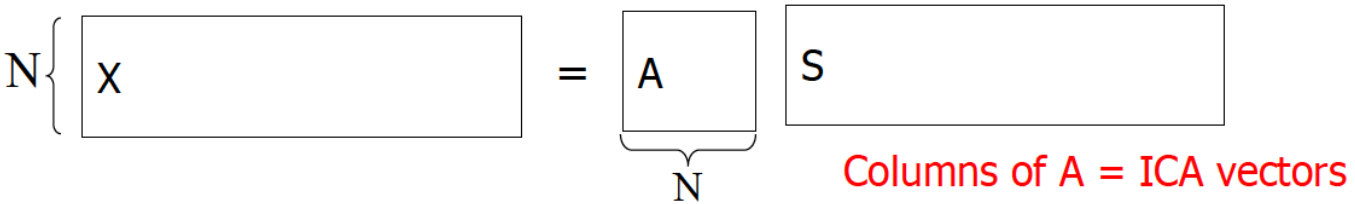
How Does ICA Work?

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



Comparing PCA and ICA

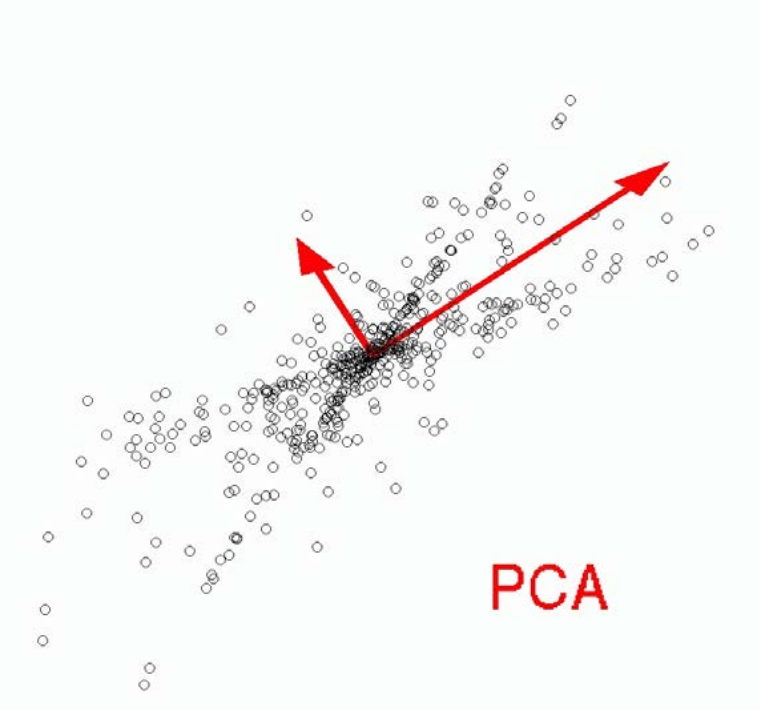
Let $\mathbf{S}$ denote our original data. It can be transformed to a new set of features $\mathbf{X}$:

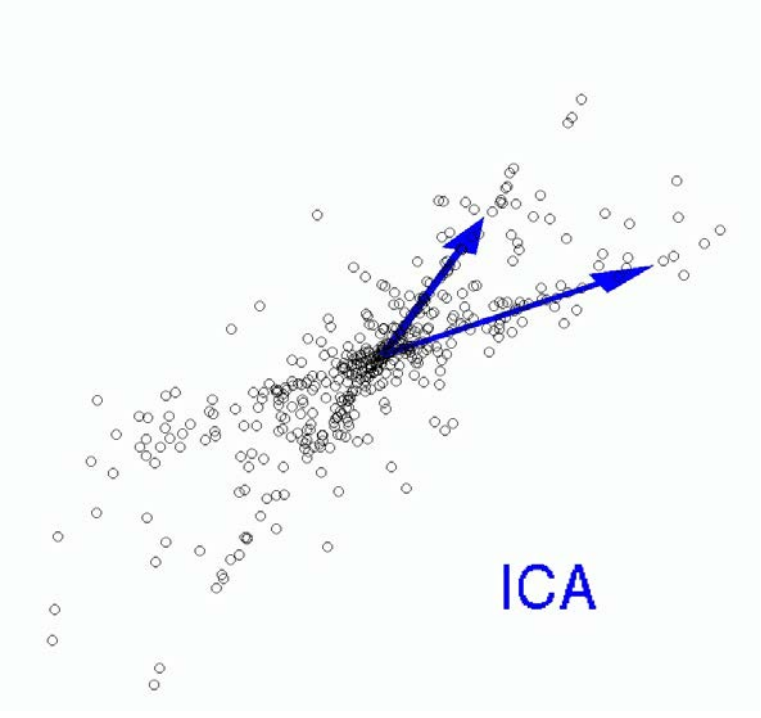
$$\text{PCA} : \mathbf{X} = \mathbf{US},\ \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

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

- 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 to exploit sparsity to speed up computations, and how to formulate the sparse PCA problem
- How PCA differs from ICA