

CS 760: Machine Learning **Unsupervised Learning II**

Ilias Diakonikolas

University of Wisconsin-Madison

Nov. 17, 2022

Announcements

- **Logistics:**

- HW6 Due Thursday. HW7 out today

- **Class roadmap:**

	Unsupervised Learning I
	Unsupervised Learning II
	Learning Theory
	RL I
	RL II

Outline

- **Clustering Review**

- k-means, hierarchical, spectral clustering

- **Gaussian Mixture Models**

- Mixtures, Expectation-Maximization algorithm

- **Principal Components Analysis**

- Definition, Algorithm, Interpretations, Analysis

Outline

- **Clustering Review**

- k-means, hierarchical, spectral clustering

- **Gaussian Mixture Models**

- Mixtures, Expectation-Maximization algorithm

- **Principal Components Analysis**

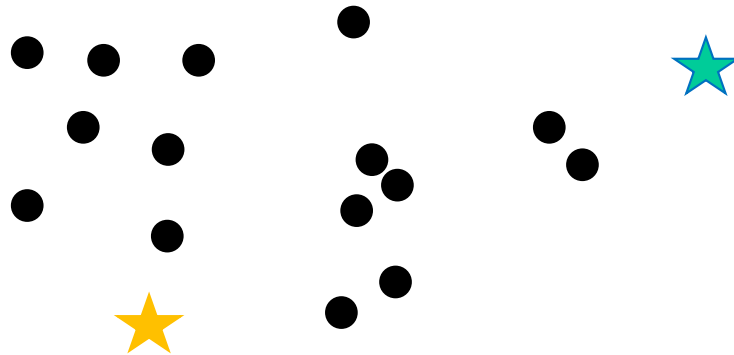
- Definition, Algorithm, Interpretations, Analysis

K-Means Clustering

k-means is a type of partitional **centroid-based clustering**

Algorithm:

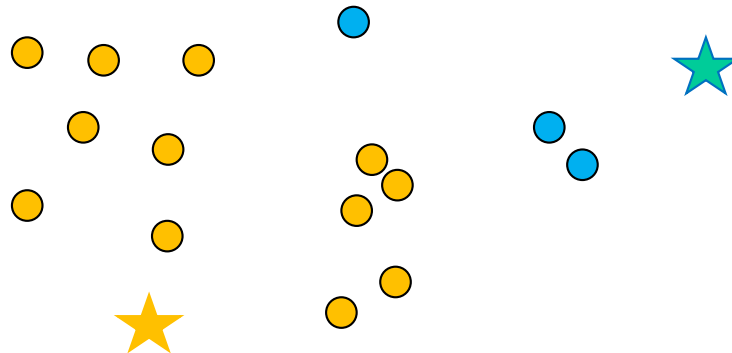
1. Randomly pick k cluster centers



K-Means Clustering: Algorithm

K-Means clustering

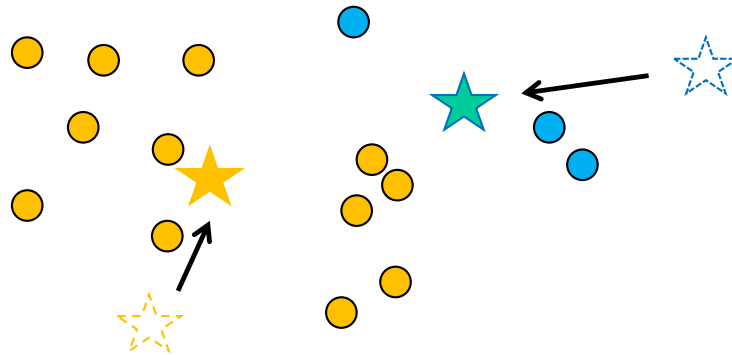
2. Find closest center for each point



K-Means Clustering: Algorithm

K-Means clustering

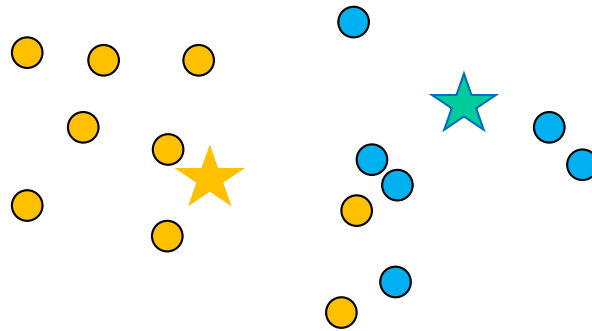
3. Update cluster centers by computing centroids



K-Means Clustering: Algorithm

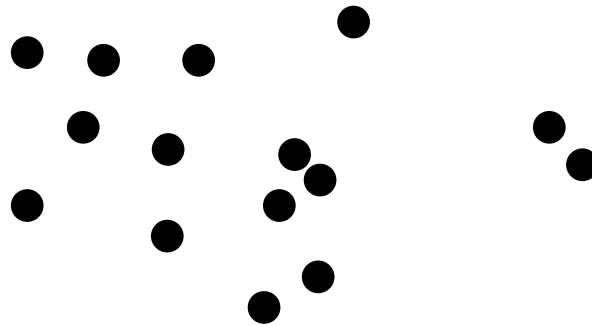
K-Means clustering

Repeat Steps 2 & 3 until convergence



HC: Agglomerative Clustering Example

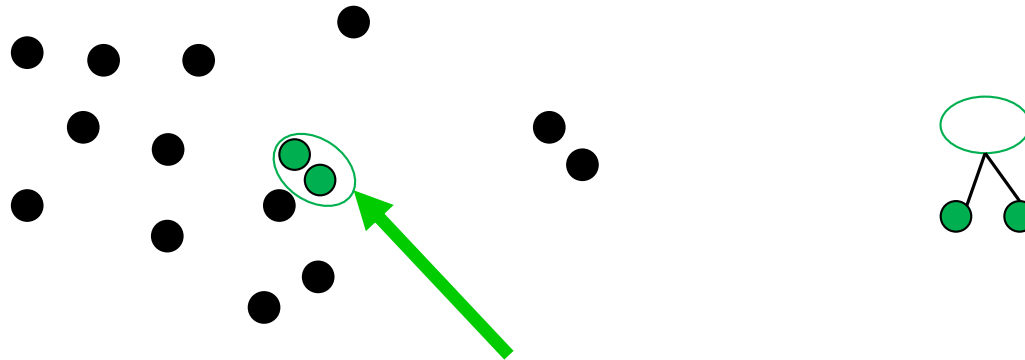
Agglomerative: Start: every point is its own cluster



HC: Agglomerative Clustering Example

Basic idea: build a “hierarchy”

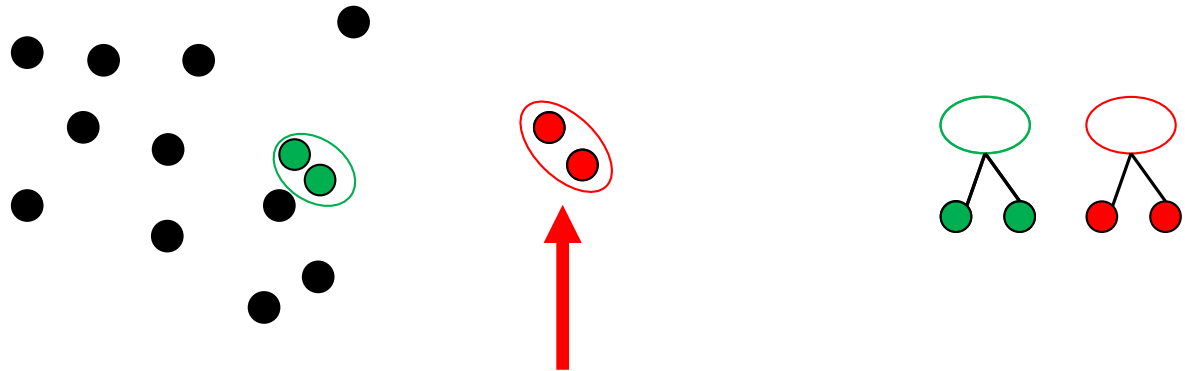
- Get pair of clusters that are closest and merge



HC: Agglomerative Clustering Example

Basic idea: build a “hierarchy”

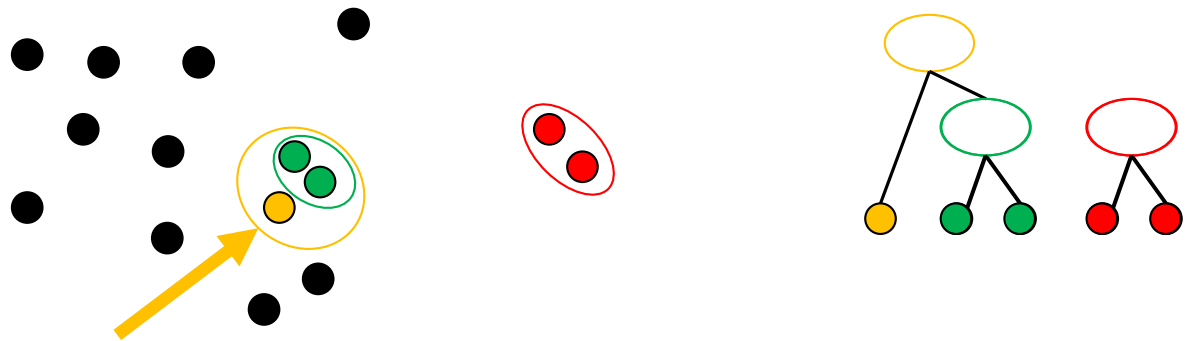
- **Repeat:** Get pair of clusters that are closest and merge



HC: Agglomerative Clustering Example

Basic idea: build a “hierarchy”

- **Repeat:** Get pair of clusters that are closest and merge



HC: Merging Criteria

Merge: use closest clusters. Define closest?

- Single-linkage

$$d(A, B) = \min_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

- Complete-linkage

$$d(A, B) = \max_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

- Average-linkage

$$d(A, B) = \frac{1}{|A||B|} \sum_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$



Break & Quiz

Break & Quiz

Q 2.1: You have seven 2-dimensional points. You run 3-means on it, with initial clusters

$$C_1 = \{(2, 2), (4, 4), (6, 6)\}, C_2 = \{(0, 4), (4, 0)\}, C_3 = \{(5, 5), (9, 9)\}$$

Cluster centroids at the next iteration are?

- A. $C_1: (4,4), C_2: (2,2), C_3: (7,7)$
- B. $C_1: (6,6), C_2: (4,4), C_3: (9,9)$
- C. $C_1: (2,2), C_2: (0,0), C_3: (5,5)$
- D. $C_1: (2,6), C_2: (0,4), C_3: (5,9)$

Break & Quiz

Q 2.1: You have seven 2-dimensional points. You run 3-means on it, with initial clusters

$$C_1 = \{(2, 2), (4, 4), (6, 6)\}, C_2 = \{(0, 4), (4, 0)\}, C_3 = \{(5, 5), (9, 9)\}$$

Cluster centroids at the next iteration are?

- **A. $C_1: (4,4)$, $C_2: (2,2)$, $C_3: (7,7)$**
- B. $C_1: (6,6)$, $C_2: (4,4)$, $C_3: (9,9)$
- C. $C_1: (2,2)$, $C_2: (0,0)$, $C_3: (5,5)$
- D. $C_1: (2,6)$, $C_2: (0,4)$, $C_3: (5,9)$

Break & Quiz

Q 2.2: If we do hierarchical clustering on n points, the maximum depth of the resulting tree is

- A. 2
- B. $\log n$
- C. $n/2$
- D. $n-1$

Break & Quiz

Q 2.2: If we do hierarchical clustering on n points, the maximum depth of the resulting tree is

- A. 2
- B. $\log n$
- C. $n/2$
- **D. $n-1$**

Outline

- **Clustering Review**

- k-means, hierarchical, spectral clustering

- **Gaussian Mixture Models**

- Mixtures, Expectation-Maximization algorithm

- **Principal Components Analysis**

- Definition, Algorithm, Interpretations, Analysis

Mixture Models

- Let's get back to modeling densities in unsupervised learning.
- Have dataset:

$$\{ (x^{(1)}, x^{(2)}, \dots, x^{(n)}) \}$$

- One type of model: **mixtures**
 - A function of the **latent variable** z
 - We did something similar with flows
 - Model:

$$p(x^{(i)} | z^{(i)}) p(z^{(i)})$$

Mixture Models: Gaussians

- Lots of different kinds of mixtures, but let's focus on Gaussians.
- What does this mean?
- Latent variable z has some multinomial distribution, $\sum_{i=1}^k \phi_i = 1$

$$z^{(i)} \sim \text{Multinomial}(\phi)$$

- Then, let's make x be conditional Gaussian

$$x^{(i)} | (z^{(i)} = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$$

 
Mean Covariance Matrix

Gaussian Mixture Models: Likelihood

- How should we learn the parameters? ϕ, μ_j, Σ_j
- Could try our usual way: maximum likelihood
 - Log likelihood:

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^n \log \sum_{z^{(i)}=1}^k p(x^{(i)} | z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi)$$

- Turns out to be **hard** to solve... inner sum leads to problems!

GMMs: Supervised Setting

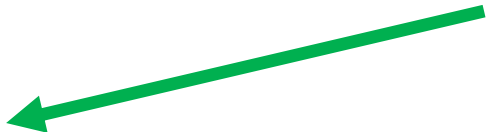
- What if we knew the z 's?
 - “Supervised” setting... very similar to Gaussian Naïve Bayes
- First, empirically estimate the z parameters:

$$\phi_j = \frac{1}{n} \sum_{i=1}^n 1\{z^{(i)} = j\}$$

- Next the Gaussian components:

$$\mu_j = \frac{\sum_{i=1}^n 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^n 1\{z^{(i)} = j\}}$$

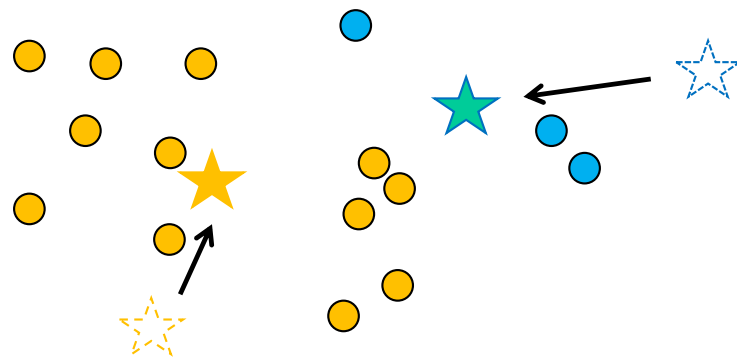
Average of x 's
where $z = j$



$$\Sigma_j = \frac{\sum_{i=1}^n 1\{z_j^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^n 1\{z_j^{(i)} = j\}}$$

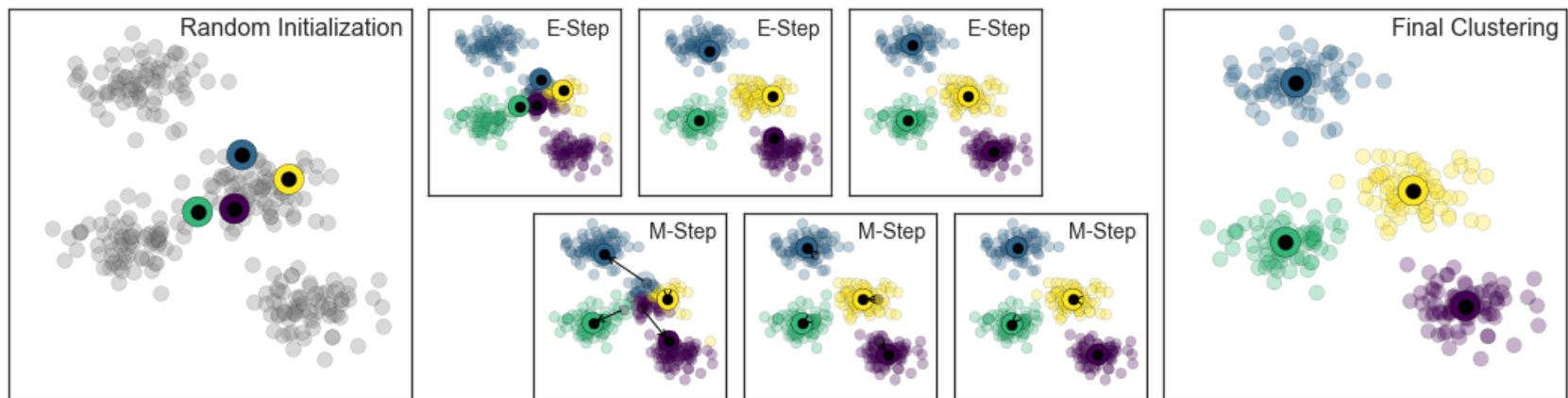
GMMs: Back to Latent Setting

- But, we don't get to see the z 's
 - Similar to the weak supervision setting from last time.
- What could we do instead?
- Recall our **k-means** approach: we don't know the centers, but we pretend we do, perform a clustering, re-center, iterate



GMMs: Expectation Maximization

- EM :an algorithm for dealing with latent variable problems
- Iterative, alternating between two steps:
 - **E**-step (expectation): guess the latent variables
 - **M**-step (maximization): update the parameters of the model
 - Note similarity to k-means clustering.



Jake VanderPlas

GMM EM: E-Step

- Let's write down the formulas.
- **E-step**: fix parameters, compute posterior:

$$w_j^{(i)} = p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

- These w 's are “soft” assignments of the z terms... probabilities over the values z could take. Concretely:


$$w_j^{(i)} = p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)} | z^{(i)} = j; \mu, \Sigma) p(z^{(i)} = j; \phi)}{\sum_{\ell=1}^k p(x^{(i)} | z^{(i)} = \ell; \mu, \Sigma) p(z^{(i)} = \ell; \phi)}$$

GMM EM: M-Step

- Let's write down the formulas.
- **M-step**: fix w , update parameters:

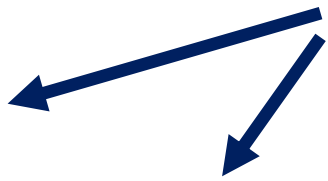
$$\phi_j = \frac{1}{n} \sum_{i=1}^n w_j^{(i)}$$

Soft version of our counting estimator for the supervised case.



$$\mu_j = \frac{\sum_{i=1}^n w_j^{(i)} x^{(i)}}{\sum_{i=1}^n w_j^{(i)}}$$

Soft version of our empirical mean and covariances.



$$\Sigma_j = \frac{\sum_{i=1}^n w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^n w_j^{(i)}}$$



Break & Quiz

Outline

- Clustering Review

- k-means, hierarchical, spectral clustering

- Gaussian Mixture Models

- Mixtures, Expectation-Maximization algorithm

- Principal Components Analysis

- Definition, Algorithm, Interpretations, Analysis

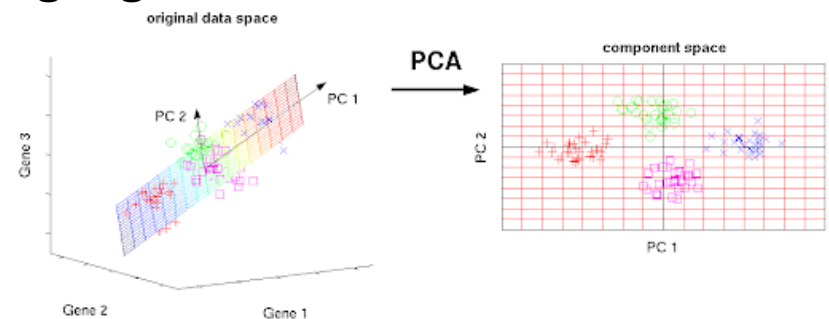
High-Dimensional Data

- High-dimensions = lots of features
- We've seen this repeatedly, but some examples:
- Document classification
 - Features per document = thousands of words/unigrams millions of bigrams, contextual information
- **Example:** Surveys - Netflix
 - 480189 users x 17770 movies

	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6
Tom	5	?	?	1	3	?
George	?	?	3	1	2	5
Susan	4	3	1	?	5	1
Beth	4	3	?	2	4	2

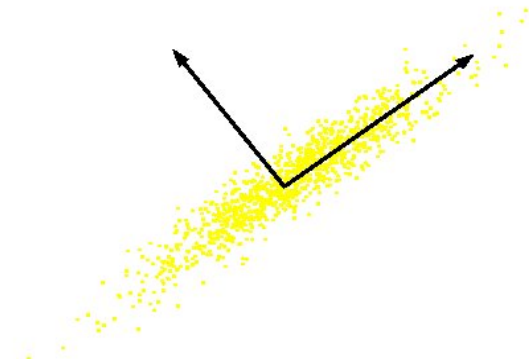
Dealing with Dimensionality

- **PCA, Kernel PCA, ICA:** Powerful unsupervised learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.
- Some uses:
 - Visualization
 - More efficient use of resources (e.g., time, memory, communication)
 - Noise removal (improving data quality)
 - Further processing by machine learning algorithms



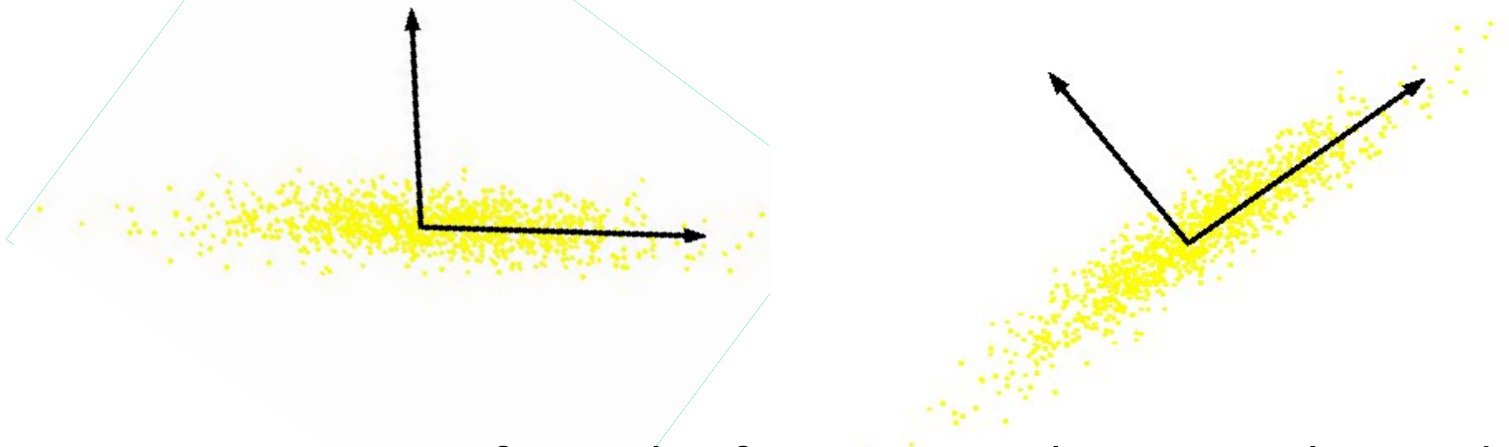
Principal Components Analysis

- Unsupervised technique for extracting variance structure from high dimensional datasets
 - And also reduces dimensionality
- PCA: orthogonal projection / transformation of the data
 - Into a (possibly lower dimensional) subspace
 - So that the variance of the projected data is maximized.



PCA Intuition

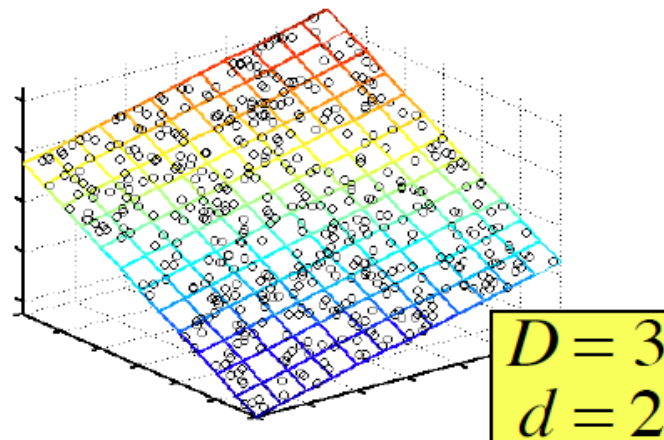
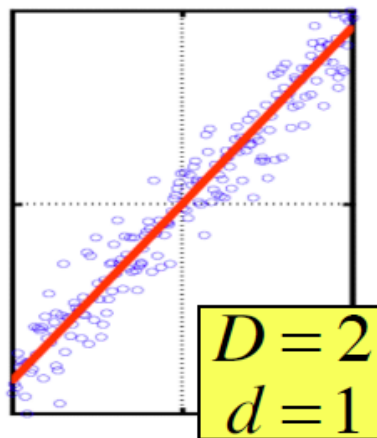
- The dimension of the ambient space (ie, \mathbb{R}^d) might be much higher than the **intrinsic** data dimension



- **Question:** Can we transform the features so that we only need to preserve one latent feature?
 - Or a few?

PCA Intuition

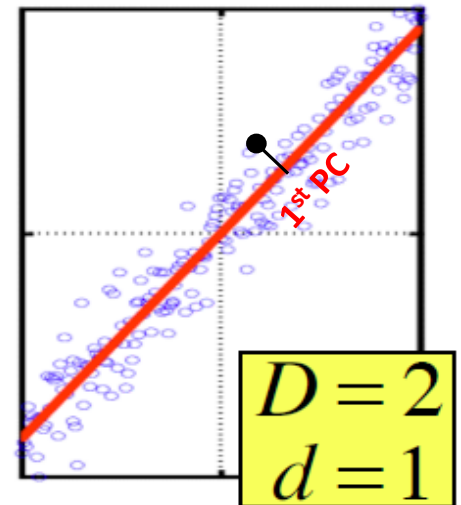
- Some more visualizations



- In case where data lies on or near a low d -dimensional linear subspace, axes of this subspace are an effective representation of the data.

PCA: Principal Components

- **Principal Components (PCs)** are orthogonal directions that capture most of the variance in the data.
- First PC – direction of greatest variability in data.
- Projection of data points along first PC discriminates data most along any one direction

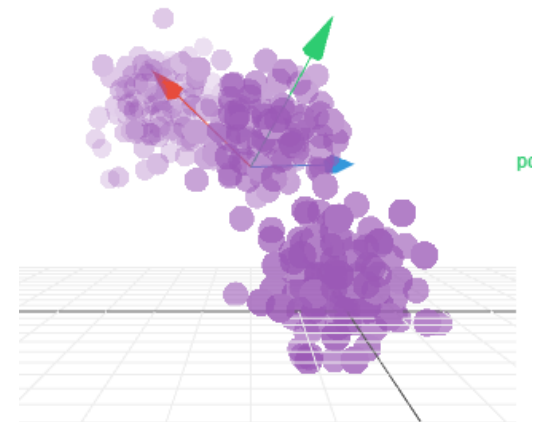


PCA: Principal Components and Projection

- How does dimensionality reduction work? From d dimensions to r dimensions:

- Get $v_1, v_2, \dots, v_r \in \mathbb{R}^d$
- Orthogonal!

- Maximizing variability
 - Equivalent to **minimizing reconstruction error**
- Then project data onto PCs \rightarrow d -dimensional



Victor Powell

PCA Approach Overview

- Want directions/components (unit vectors) so that
 - Projecting data maximizes variance
 - Specifically, for centered data

$$\sum_{i=1}^n \langle x_i, v \rangle = \|Xv\|^2$$

- Do this **recursively**
 - Get orthogonal directions

$$v_1, v_2, \dots, v_r \in \mathbb{R}^d$$

PCA First Step

- First component,

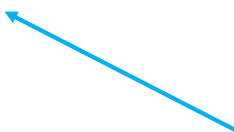
$$v_1 = \arg \max_{\|v\|=1} \sum_{i=1}^n \langle v, x_i \rangle^2$$

- Same as getting

$$v_1 = \arg \max_{\|v\|=1} \|Xv\|^2$$

PCA Recursion

- Once we have $k-1$ components, next?

$$\hat{X}_k = X - \sum_{i=1}^{k-1} X v_i v_i^T$$


Deflation

- Then do the same thing

$$v_k = \arg \max_{\|v\|=1} \|\hat{X}_k w\|^2$$

PCA Interpretations

- The v 's are eigenvectors of XX^T (**Gram matrix**)
 - We'll see why in a second
- XX^T (proportional to) sample covariance matrix
 - When data is 0 mean!
 - I.e., PCA is eigendecomposition of sample covariance
- Nested subspaces $\text{span}(v_1)$, $\text{span}(v_1, v_2)$, ...,



PCA Interpretations: First Component

- Two specific ways to think about the first component
- **Maximum variance direction**
 - What we saw so far

$$\sum_{i=1}^n (\mathbf{v}^T \mathbf{x}_i)^2 = \mathbf{v}^T \mathbf{X} \mathbf{X}^T \mathbf{v}$$

- **Minimum reconstruction error**
 - A direction so that projection yields minimum MSE in reconstruction

$$\sum_{i=1}^n \|\mathbf{x}_i - (\mathbf{v}^T \mathbf{x}_i) \mathbf{v}\|^2$$

PCA Interpretations: Equivalence

- Interpretation 1.

Maximum variance direction

$$\sum_{i=1}^n (\mathbf{v}^T \mathbf{x}_i)^2 = \mathbf{v}^T \mathbf{X} \mathbf{X}^T \mathbf{v}$$

- Interpretation 2.

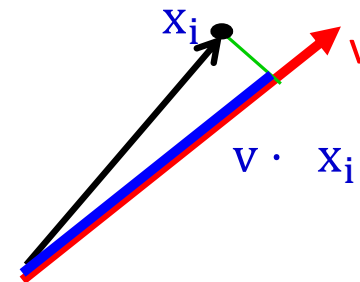
Minimum reconstruction error

$$\sum_{i=1}^n \|\mathbf{x}_i - (\mathbf{v}^T \mathbf{x}_i) \mathbf{v}\|^2$$

- Why are these equivalent?

- Use Pythagorean theorem.

- Maximizing **blue** segment is the same as minimizing the **green**



PCA Gram Matrix Interpretation

- Recall our first PC, maximized variance:

$$\max_{\mathbf{v}} \mathbf{v}^T \mathbf{X} \mathbf{X}^T \mathbf{v} \quad \text{s.t.} \quad \mathbf{v}^T \mathbf{v} = 1$$

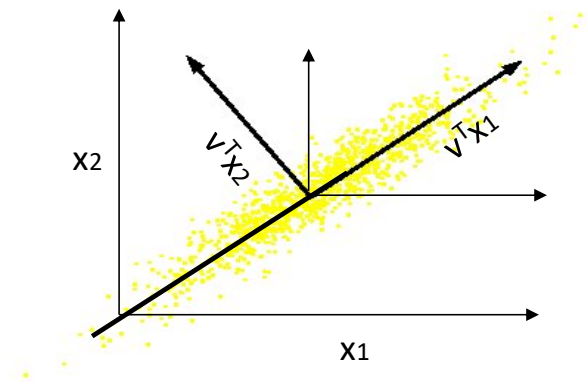
- Constrained optimization
 - Recall our usual approach: Lagrangian + KKT conditions

$$\text{Lagrangian: } \max_{\mathbf{v}} \mathbf{v}^T \mathbf{X} \mathbf{X}^T \mathbf{v} - \lambda \mathbf{v}^T \mathbf{v}$$

$$\partial / \partial \mathbf{v} = 0 \quad (\mathbf{X} \mathbf{X}^T - \lambda \mathbf{I}) \mathbf{v} = 0 \quad \Rightarrow \quad (\mathbf{X} \mathbf{X}^T) \mathbf{v} = \lambda \mathbf{v}$$

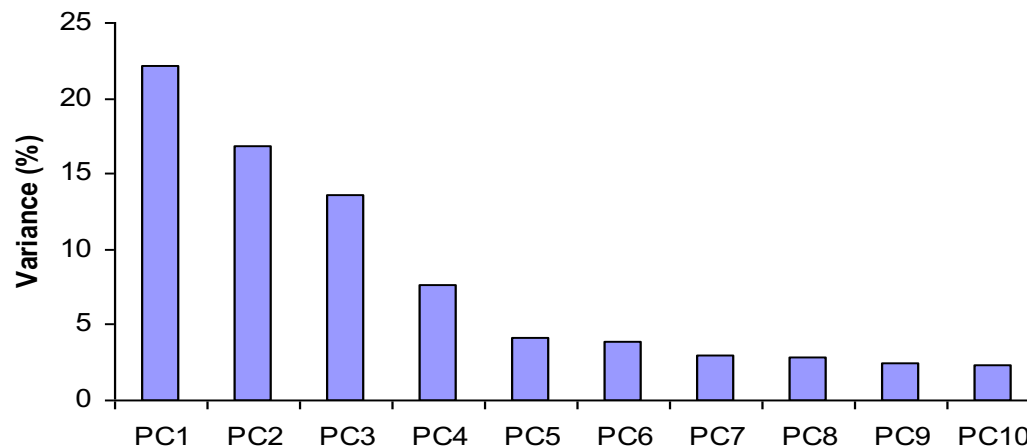
PCA Covariance Matrix Interpretation

- So... $\Rightarrow (XX^T)v = \lambda v$
- Means that v (the first PC) is an eigenvector of XX^T
- Its eigenvalue λ denotes the amount of variability captured along that dimension
- PCs are just the eigenvectors...
 - How to find them? Eigendecomposition
- Don't need to keep all eigenvectors
 - Just the ones for largest eigenvalues



PCA Dimensionality Reduction

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



Application: Image Compression

- Start with image; divide into 12x12 patches
 - I.E., 144-D vector
- **Original image:**



Application: Image Compression

- Project to 6D,



Compressed



Original

Q2-2: Are these statements true or false?

(A) The principal component with the largest eigenvalue maximizes the reconstruction error.

(B) The dimension of original data representation is always higher than the dimension of transformed representation of PCA.

1. True, True
2. True, False
3. False, True
4. False, False

Q2-2: Are these statements true or false?

(A) The principal component with the largest eigenvalue maximizes the reconstruction error.

(B) The dimension of original data representation is always higher than the dimension of transformed representation of PCA.

1. True, True

2. True, False

3. False, True

4. False, False



(A) The principal component with the largest eigenvalue captures the maximum amount of variability which is equivalent to minimum reconstruction error.

(B) If the matrix XX^T is full-rank, they can be of the same dimension.



Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Fred Sala