Clustering and K-means algorithm

Oliver W. Layton

CS251: Data analysis and visualization

Lecture 22, Fall 2018

Monday April 3
Plan

- Finish PCA
- PCA Application: Eigenfaces
- Clustering
- Distance metrics
- K-means algorithm
- Leader algorithm
Application: Eigenfaces

- Dataset: MIT Faces Recognition Project database.
  - > 1000 face images.
- Simulation images from https://sandipanweb.wordpress.com/.
Random sample of 500 faces
Eigenface algorithm: PCA on face images

1. Load in grayscale images with same x-y dimensions $I_1, I_2, \ldots, I_M$.

2. Collapse each 2D image into 1D vectors $\Gamma_i$ (e.g. 16x16 2D image $\Rightarrow$ 256 1D vector)

3. Center the images (subtract grand mean image). $\Phi_i = \Gamma_i - \frac{1}{M} \sum_{j=0}^{M-1} \Gamma_j$

4. Compute SVD.

5. Project images onto a subset $k$ of principal components.
Grand mean of 500 faces
Variance explained by top eigenvalues/PCs
The "facial dimensions": Resizing (1D vectors -> 2D) top eVecs
Project one face image onto top $K$ PCs
Clustering
Iris data in PCA space
**Identifying clusters in data**

- **Goal:** Assign each data point to "groups" based on similarity to other data points (clusters).

- Grouping means "tagging" or "coloring" each point. Like adding another feature (dimension) with an int that represents group membership (e.g. group 1, group 2, etc.).

- Algorithms differ in:
  - how similarity defined.
  - whether group membership is exclusive (1 group per point) or **fuzzy** (point belongs to multiple groups, to varying degrees).
  - how many groups/clusters to use.
  - whether grouping happens **top-down** (big picture first) or **bottom-up** (data point level).
Input: Iris data
Output: Iris data assigned to 2 clusters
Distance metric (1/2)

• What's the most common way to measure distance?
  • Euclidean distance. But there are other ways...

• **Distance metric**: assign a *scalar value* between 2 pts $\vec{x}$ and $\vec{y}$ that represents how far they are from one another.

• Function that yields this scalar value: $d(\vec{x}, \vec{y})$. 
Distance metric (2/2)

- Properties of $d(\vec{x}, \vec{y})$:
  - $d(\vec{x}, \vec{y}) > 0$ for any 2 pts $\vec{x}$ and $\vec{y}$.
  - $d(\vec{x}, \vec{x}) = 0$
  - Obey triangle inequality: $d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z})$.
    - Intuitively: Direct path between 2 pts ("straight line") at least as short a path as any kind of triangulation with another waypoint.
  - Order of args doesn't matter: $d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x})$
\[ L^2 \] norm

- **Norm**: Way to measure distance.

- Euclidean distance called \( L^2 \) norm.

\[
\left\| \vec{x} - \vec{y} \right\|_2 = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}
\]

- Advantages: Intuitive geometry, can do calculus on it (e.g. take derivatives, integrate, etc.)
\( L^1 \) norm

- **City block distance** is the \( L^1 \) ("elle one") norm.

\[
d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} |x_i - y_i| = |\mathbf{x} - \mathbf{y}|_1
\]

- | \cdot | means absolute value.

- Advantages: Easier computations, runs faster on computers.
\( L^p \) norm

- In general, \( L^p \) norm defined for some positive number \( p \):
  \[
  |x|_p = \left( \sum_{i=1}^{N} |x_i|^p \right)^{\frac{1}{p}}
  \]

- \( L^\infty \) norm: Limiting case when \( p \) gets really, really large:
  \[
  |x|_\infty = \lim_{p \to \infty} |x|_p = \max_i |x_i|
  \]

- Let's compare all the \( L^p \) norm geometries on the board.