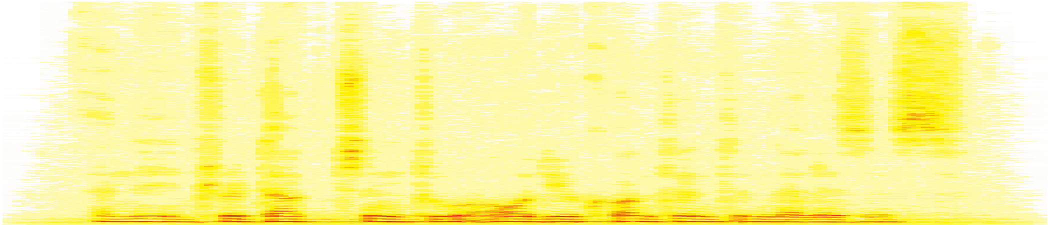
Introduction to Audio Content Analysis

Module 8.2: Music Similarity

alexander lerch



Georgia Tech Center for Music Technology
College of Design

corresponding textbook section

Chapter 8: Musical Genre, Similarity, and Mood (pp. 156–157)

- **lecture content**
  - music similarity and its relation to musical genre
  - clustering and visualization of feature space

- **learning objectives**
  - describe potential issues with algorithms for measuring music similarity
  - implement a simple k-Means algorithm

corresponding textbook section

Chapter 8: Musical Genre, Similarity, and Mood (pp. 156–157)

- **lecture content**
  - music similarity and its relation to musical genre
  - clustering and visualization of feature space

- **learning objectives**
  - describe potential issues with algorithms for measuring music similarity
  - implement a simple k-Means algorithm

- genre classification is just a grouping by specific interpretation of similarity
  - similar set of features
  - ambiguous 'ground truth'
  - unclear value/impact of low level and high level features

- **differences** to genre classification
  - *similarity*: distance measure instead of categorizing into classes

- genre classification is just a grouping by specific interpretation of similarity
    - similar set of features
    - ambiguous 'ground truth'
    - unclear value/impact of low level and high level features
- **differences** to genre classification
    - *similarity*: distance measure instead of categorizing into classes

- **perception** of music similarity
  - multi-dimensional (melodic, rhythmic, sound quality, ...)
  - user dependent
  - associative, may also depend on editorial data
  - may be context dependent

- genres are **clusters of musical similarity**

$\Rightarrow$ genre classification is a *special case* of audio similarity measures

- instead of assigning (genre) labels, the similarity/distance between (pairs) of files is measured

- **perception** of music similarity
  - multi-dimensional (melodic, rhythmic, sound quality, ...)
  - user dependent
  - associative, may also depend on editorial data
  - may be context dependent

- genres are **clusters of musical similarity**

$\Rightarrow$ genre classification is a *special case* of audio similarity measures

- instead of assigning (genre) labels, the similarity/distance between (pairs) of files is measured

- **perception** of music similarity
  - multi-dimensional (melodic, rhythmic, sound quality, ...)
  - user dependent
  - associative, may also depend on editorial data
  - may be context dependent

- genres are **clusters of musical similarity**

⇒ genre classification is a *special case* of audio similarity measures

- instead of assigning (genre) labels, the similarity/distance between (pairs) of files is measured

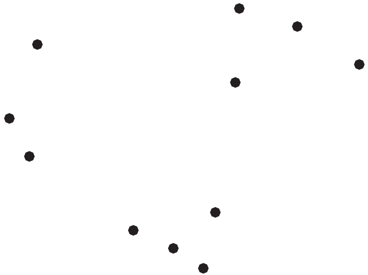
- simple k-means example
  - **goal**: minimize intra-cluster variance
  - **distance**: Euclidean
  - **procedure**:
    1. *initialization*:
       randomly select K points in the feature space as initialization.
    2. *assignment*:
       assign each observation to the cluster with the mean/centroid of the closest cluster.
    3. *update*:
       compute mean/centroid for each cluster.
    4. *iteration*:
       go to step 2 until the clusters converge.

- simple k-means example
  - **goal**: minimize intra-cluster variance
  - **distance**: Euclidean
  - **procedure**:
    1. *initialization*:
       randomly select K points in the feature space as initialization.
    2. *assignment*:
       assign each observation to the cluster with the mean/centroid of the closest cluster.
    3. *update*:
       compute mean/centroid for each cluster.
    4. *iteration*:
       go to step 2 until the clusters converge.

- simple k-means example
  - **goal**: minimize intra-cluster variance
  - **distance**: Euclidean
  - **procedure**:
    1. *initialization*:
       randomly select K points in the feature space as initialization.
    2. *assignment*:
       assign each observation to the cluster with the mean/centroid of the closest cluster.
    3. *update*:
       compute mean/centroid for each cluster.
    4. *iteration*:
       go to step 2 until the clusters converge.

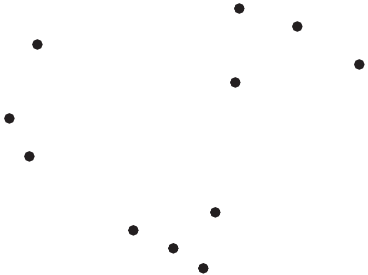
- simple k-means example
  - **goal**: minimize intra-cluster variance
  - **distance**: Euclidean
  - **procedure**:
    1. *initialization*:
       randomly select K points in the feature space as initialization.
    2. *assignment*:
       assign each observation to the cluster with the mean/centroid of the closest cluster.
    3. *update*:
       compute mean/centroid for each cluster.
    4. *iteration*:
       go to step 2 until the clusters converge.

- data points
- init randomly
- iteration
  - assign data points to closest centroid
  - update centroids
- terminate when convergence



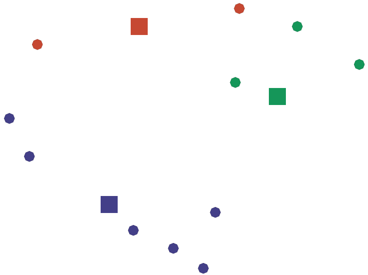
matlab source: matlab/displayKMeans.m

- data points
- init randomly
- iteration
  - assign data points to closest centroid
  - update centroids
- terminate when convergence



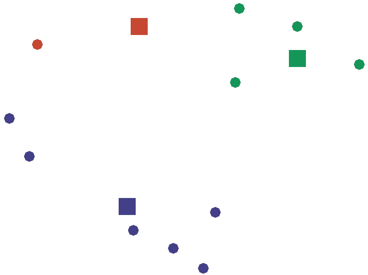
matlab source: matlab/displayKMeans.m

- data points
- init randomly
- iteration
  - assign data points to closest centroid
  - update centroids
- terminate when convergence



matlab source: matlab/displayKMeans.m

- data points
- init randomly
- iteration
  - assign data points to closest centroid
  - update centroids
- terminate when convergence



matlab source: matlab/displayKMeans.m

- data points
- init randomly
- iteration
  - assign data points to closest centroid
  - update centroids
- terminate when convergence

matlab source: matlab/displayKMeans.m

- data points
- init randomly
- iteration
  - assign data points to closest centroid
  - update centroids
- terminate when convergence

matlab source: matlab/displayKMeans.m

- data points
- init randomly
- iteration
    - assign data points to closest centroid
    - update centroids
- terminate when convergence

matlab source: matlab/displayKMeans.m

- **problem**
  - feature space is high-dimensional
  - → cannot be visualized

- find **mapping** to 2D "preserving" (high-dimensional) distance metrics
  example:
  - Self-Organizing Maps

- **problem**
  - feature space is high-dimensional

  → cannot be visualized

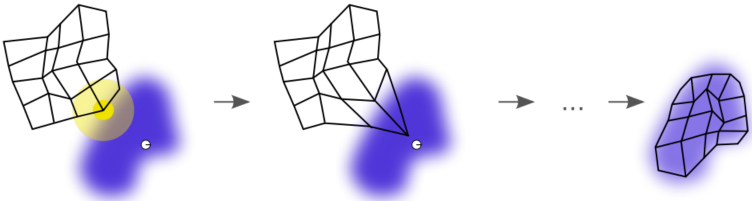
- find **mapping** to 2D "preserving" (high-dimensional) distance metrics

  example:
  - Self-Organizing Maps

1. create a map with 'neurons'
2. train
   - for each training sample find BMU (best matching unit)
   - adapt BMU and neighbors toward training sample

$$W_v(t+1) = W_v(t) + \theta(u,v,t)\alpha(t)\big(D(t) - W_v(t)\big)$$

- $\theta(u,v,t)$: depends on neighborhood distance from BMU
- $\alpha(t)$: learning restraint
- $D(t)$ training sample



https://en.wikipedia.org/wiki/Self-organizing_map

audio similarity
SOM 2/2

from[1]

[1] **pampalk˙islands˙2001**.

- **music similarity**
  - even less clearly defined than music genre
- **processing steps**
  1. extract features
  2. define some distance metric in feature space
- **clustering algorithms**
  - work to a certain degree with traditional features