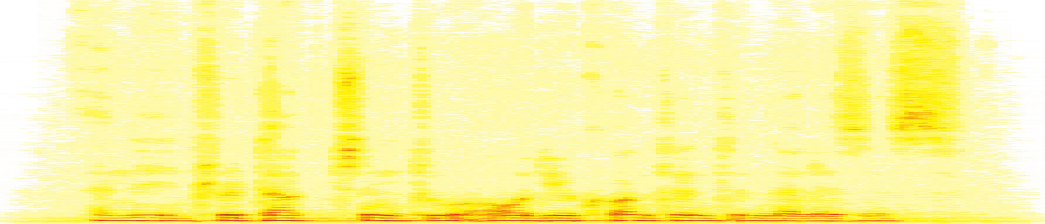
Introduction to Audio Content Analysis

Module 8.0: Classifiers

alexander lerch



Georgia Tech Center for Music Technology
College of Design

introduction
overview

corresponding textbook section

Chapter 8: Musical Genre, Similarity, and Mood (pp. 155)

- **lecture content**
  - training set and test set
  - intuitive intro to machine learning
  - classifier examples

- **learning objectives**
  - describe the basic principles and challenges of data-driven machine learning approaches
  - implement a kNN classifier in Matlab

introduction
overview

corresponding textbook section

Chapter 8: Musical Genre, Similarity, and Mood (pp. 155)

- **lecture content**
  - training set and test set
  - intuitive intro to machine learning
  - classifier examples
- **learning objectives**
  - describe the basic principles and challenges of data-driven machine learning approaches
  - implement a kNN classifier in Matlab

classification

machine learning

- computer runs a 'generic' program
- adapts parameters to training data

⇒ **data driven** approach

data and its representation defines much of the outcome

- *validity*: is data representative sample and do features focus on important characteristics
- *reliability*: does data lead to accurate and consistent results
- *reproducibility*: will multiple runs result in similar results

classification
machine learning

- computer runs a 'generic' program
- adapts parameters to training data

$\Rightarrow$ **data driven** approach

data and its representation defines much of the outcome

- *validity*: is data representative sample and do features focus on important characteristics
- *reliability*: does data lead to accurate and consistent results
- *reproducibility*: will multiple runs result in similar results

classification

general steps

1. **define training set**: annotated results
2. **normalize** training set
3. **train** classifier
4. **evaluate** classifier with test (or validation) set
5. (**adjust** classifier settings, return to 4.)

classification

general steps

1. **define training set**: annotated results
2. **normalize** training set
3. **train** classifier
4. **evaluate** classifier with test (or validation) set
5. (**adjust** classifier settings, return to 4.)

classification

general steps

1. **define training set**: annotated results
2. **normalize** training set
3. **train** classifier
4. **evaluate** classifier with test (or validation) set
5. (**adjust** classifier settings, return to 4.)

classification

general steps

1. **define training set**: annotated results
2. **normalize** training set
3. **train** classifier
4. **evaluate** classifier with test (or validation) set
5. (**adjust** classifier settings, return to 4.)

classification

general steps

1. **define training set**: annotated results
2. **normalize** training set
3. **train** classifier
4. **evaluate** classifier with test (or validation) set
5. (**adjust** classifier settings, return to 4.)

classification

rules of thumb

- **training set**
    - training set size vs. number of features
        - training set too small,feature number too large $\Rightarrow$ *overfitting*
    - training set **too noisy** $\Rightarrow$ *underfitting*
    - training set **not representative** $\Rightarrow$ *bad classification performance*
- **classifier**
    - classifier too complex $\Rightarrow$ *overfitting*
    - **poor classifier** $\Rightarrow$ *bad classification performance*
        - $\rightarrow$ different classifier
- **features**
    - **poor features** $\Rightarrow$ *bad classification performance*
        - $\rightarrow$ new, better features
    - features **not normalized** $\Rightarrow$ possibly *bad classification performance*
        - feature distribution (range, mean, symmetry)

classification

rules of thumb

Georgia Tech Center for Music Technology College of Design

- **training set**
  - training set size vs. number of features
    - training set too small,feature number too large ⇒ *overfitting*
  - training set **too noisy** ⇒ *underfitting*
  - training set **not representative** ⇒ *bad classification performance*
- **classifier**
  - classifier too complex ⇒ *overfitting*
  - **poor classifier** ⇒ *bad classification performance*

    → different classifier
- **features**
  - **poor features** ⇒ *bad classification performance*

    → new, better features
  - features **not normalized** ⇒ possibly *bad classification performance*
    - feature distribution (range, mean, symmetry)

classification

rules of thumb

- **training set**
  - training set size vs. number of features
    - training set too small,feature number too large ⇒ *overfitting*
  - training set **too noisy** ⇒ *underfitting*
  - training set **not representative** ⇒ *bad classification performance*
- **classifier**
  - classifier too complex ⇒ *overfitting*
  - **poor classifier** ⇒ *bad classification performance*

    → different classifier
- **features**
  - **poor features** ⇒ *bad classification performance*

    → new, better features
  - features **not normalized** ⇒ possibly *bad classification performance*
    - feature distribution (range, mean, symmetry)

classification
evaluation

- define **test set** for evaluation
  - test set *different* from training set
  - otherwise, same requirements

- example: **N-fold cross validation**
  1. split training set into N parts (randomly, but preferably identical number per class)
  2. select one part as test set
  3. train the classifier with all observations from remaining $N-1$ parts
  4. compute the classification rate for the test set
  5. repeat until all N parts have been tested
  6. overall result: *average* classification rate

classification

evaluation

- define **test set** for evaluation
  - test set *different* from training set
  - otherwise, same requirements

- example: ***N*-fold cross validation**
  1. split training set into N parts (randomly, but preferably identical number per class)
  2. select one part as test set
  3. train the classifier with all observations from remaining $N - 1$ parts
  4. compute the classification rate for the test set
  5. repeat until all N parts have been tested
  6. overall result: *average* classification rate

classification
evaluation

- define **test set** for evaluation
  - test set *different* from training set
  - otherwise, same requirements

- example: ***N*-fold cross validation**
  1. split training set into N parts (randomly, but preferably identical number per class)
  2. select one part as test set
  3. train the classifier with all observations from remaining $N - 1$ parts
  4. compute the classification rate for the test set
  5. repeat until all N parts have been tested
  6. overall result: *average* classification rate

classification

evaluation

- define **test set** for evaluation
  - test set *different* from training set
  - otherwise, same requirements

- example: **N-fold cross validation**
  1. split training set into N parts (randomly, but preferably identical number per class)
  2. select one part as test set
  3. train the classifier with all observations from remaining $N-1$ parts
  4. compute the classification rate for the test set
  5. repeat until all N parts have been tested
  6. overall result: *average* classification rate

classification

evaluation

- define **test set** for evaluation
  - test set *different* from training set
  - otherwise, same requirements
- example: **N-fold cross validation**
  1. split training set into N parts (randomly, but preferably identical number per class)
  2. select one part as test set
  3. train the classifier with all observations from remaining $N-1$ parts
  4. compute the classification rate for the test set
  5. repeat until all N parts have been tested
  6. overall result: *average* classification rate

classification

evaluation

- define **test set** for evaluation
  - test set *different* from training set
  - otherwise, same requirements

- example: ***N*-fold cross validation**
  1. split training set into N parts (randomly, but preferably identical number per class)
  2. select one part as test set
  3. train the classifier with all observations from remaining $N-1$ parts
  4. compute the classification rate for the test set
  5. repeat until all N parts have been tested
  6. overall result: *average* classification rate

classification

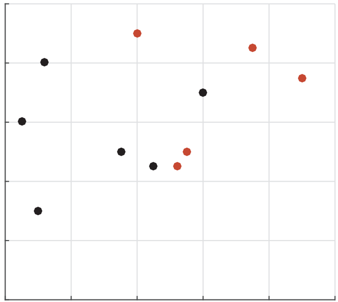
evaluation

- define **test set** for evaluation
  - test set *different* from training set
  - otherwise, same requirements

- example: **N-fold cross validation**
  1. split training set into N parts (randomly, but preferably identical number per class)
  2. select one part as test set
  3. train the classifier with all observations from remaining $N - 1$ parts
  4. compute the classification rate for the test set
  5. repeat until all N parts have been tested
  6. overall result: *average* classification rate

classifier examples
k-Nearest Neighbor (kNN)

- **training**: extract reference vectors from training set (keep class labels)
- **classification**: extract test vector and set class to majority of k nearest reference vectors
- **classifier data**: all training vectors



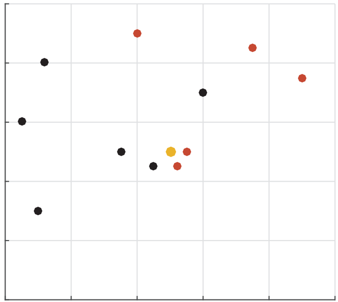
matlab source: matlab/displayKnn.m

classifier examples
k-Nearest Neighbor (kNN)

Georgia Tech Center for Music Technology College of Design

- **training**: extract reference vectors from training set (keep class labels)
- **classification**: extract test vector and set class to majority of k nearest reference vectors

- **classifier data**: all training vectors

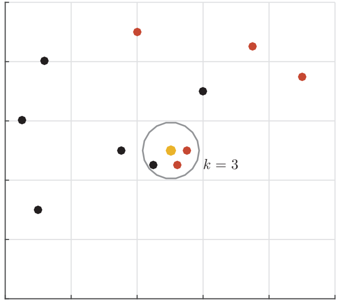


matlab source: matlab/displayKnn.m

classifier examples
k-Nearest Neighbor (kNN)

- **training**: extract reference vectors from training set (keep class labels)
- **classification**: extract test vector and set class to majority of k nearest reference vectors

- **classifier data**: all training vectors



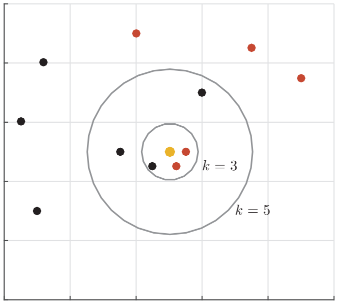
matlab source: matlab/displayKnn.m

$k = 3 \Rightarrow$ red majority

classifier examples
k-Nearest Neighbor (kNN)

- **training**: extract reference vectors from training set (keep class labels)
- **classification**: extract test vector and set class to majority of k nearest reference vectors

- **classifier data**: all training vectors



matlab source: matlab/displayKnn.m

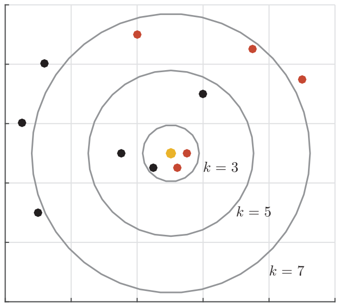
$k = 5 \Rightarrow$ black majority

classifier examples
k-Nearest Neighbor (kNN)

Georgia Tech Center for Music Technology
College of Design

- **training**: extract reference vectors from training set (keep class labels)
- **classification**: extract test vector and set class to majority of k nearest reference vectors

- **classifier data**: all training vectors



$k = 7 \Rightarrow$ red majority

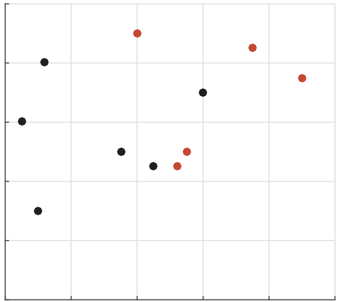
matlab source: matlab/displayKnn.m

classifier examples
k-Nearest Neighbor (kNN)

Georgia Tech Center for Music Technology College of Design

- **training**: extract reference vectors from training set (keep class labels)
- **classification**: extract test vector and set class to majority of k nearest reference vectors

- **classifier data**: all training vectors



matlab source: matlab/displayKnn.m

classifier examples

Gaussian Mixture Model (GMM)

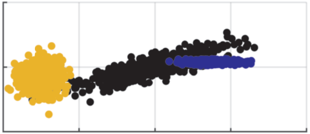
Georgia Tech Center for Music Technology College of Design

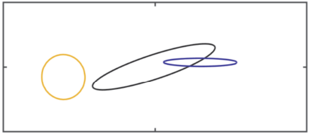
- **training**: build model of each class distribution as superposition of Gaussian distributions
- **classification**: compute output of each Gaussian and select class with highest probability
- **classifier data**: per class per Gaussian: μ and covariance, mixture weight?

classifier examples
Gaussian Mixture Model (GMM)

Georgia Tech Center for Music Technology College of Design

- **training**: build model of each class distribution as superposition of Gaussian distributions
- **classification**: compute output of each Gaussian and select class with highest probability





matlab source: matlab/displayGmm.m

- **classifier data**: per class per Gaussian: μ and covariance, mixture weight?

classifier examples

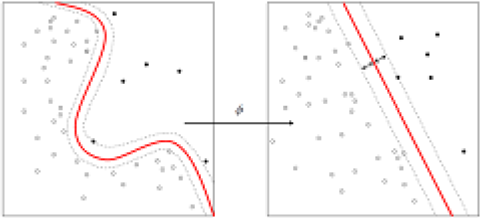
Gaussian Mixture Model (GMM)

- **training**: build model of each class distribution as superposition of Gaussian distributions
- **classification**: compute output of each Gaussian and select class with highest probability
- **classifier data**: per class per Gaussian: μ and covariance, mixture weight?

classifier examples

Support Vector Machine (SVM)

- **training**:
  - map features to high dimensional space



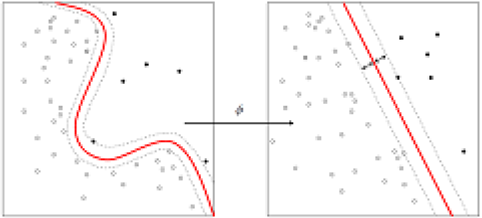
  - find separating hyperplane through maximum distance of support vectors (data points)
- **classification**: apply feature transform and proceed with 'linear' classification
- **classifier data**: support vectors, kernel, kernel parameters

https://en.wikipedia.org/wiki/Support_vector_machine

classifier examples
Support Vector Machine (SVM)

Georgia Tech Center for Music Technology College of Design

- **training**:
  - map features to high dimensional space



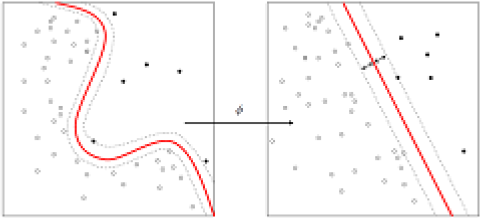
  - find separating hyperplane through maximum distance of support vectors (data points)
- **classification**: apply feature transform and proceed with 'linear' classification
- **classifier data**: support vectors, kernel, kernel parameters

https://en.wikipedia.org/wiki/Support_vector_machine

classifier examples

Support Vector Machine (SVM)

- **training**:
  - map features to high dimensional space



  - find separating hyperplane through maximum distance of support vectors (data points)
- **classification**: apply feature transform and proceed with 'linear' classification
- **classifier data**: support vectors, kernel, kernel parameters

https://en.wikipedia.org/wiki/Support_vector_machine

summary
lecture content

- **data-driven approach**
  - general systems that learn behavior from data
  - human interaction through
    - parametrization and procedures
    - data selection

- **training & test set**
  1. must not overlap
  2. must be representative

- **fine balance of inputs**
  1. number of features
  2. classifier complexity
  3. amount and variability of data