ECEN 4532: Digital Signal Processing Lab

Lecture Notes: Lab 3

Instructor: Prof. Farhad Pourkamali-Anaraki

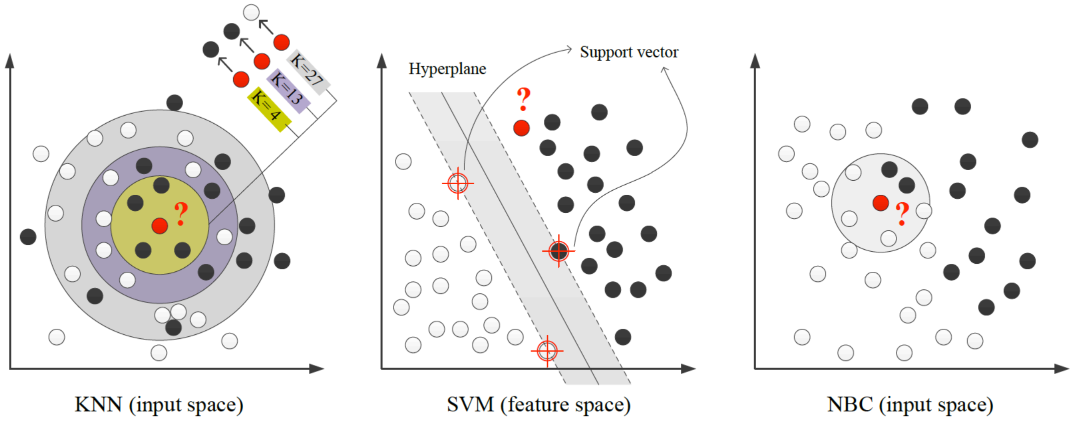
University of Colorado at Boulder

Spring 2017

University of Colorado
Boulder

Introduction

- In this lab, we will use classification methods to automatically detect the genre of a track.



- Database of 150 training samples (6x25)
- Please download these tracks at: (valid for 7 days)

https://www.dropbox.com/s/rcr3ps100eu1hgg/Lab3-audio-tracks.zip?dl=0

Introduction

- Therefore, we need to develop distances to compare audio tracks.

- The distances are based on the features that we developed in the first and second labs:

    - MFCC coefficients
    - Chroma (Normalized Pitch Class Profile)

Genre Classification

- We will consider the following genres:

1. classical
2. electronic
3. jazz/blues
4. metal/punk
5. pop/rock
6. world

Statistical Evaluation of Algorithms

- **Confusion Matrix:** The accuracy of the classification will be quantified using confusion matrices.

- For a given classification experiment, you will construct a **6 × 6 matrix** where the **rows** are the true genres, and the **columns** are the genres classified by your algorithm.

Confusion Matrices

- The entry $R(i,j)$ of the confusion matrix R is the number of songs of genre i that were classified as genre j.

predicted genre by your algorithm

true genre

	classical	electronic	jazz	punk	rock	world
classical	21	3	1	0	0	0
electronics	3	22	0	0	0	0
jazz	0	1	13	2	8	1
punk	0	0	0	23	1	1
rock	0	2	3	6	9	5
world	3	7	1	0	4	10

Confusion Matrices

- Diagonal entries of the confusion matrix:

	classical	electronic	jazz	punk	rock	world
classical	21	3	1	0	0	0
electronics	3	22	0	0	0	0
jazz	0	1	13	2	8	1
punk	0	0	0	23	1	1
rock	0	2	3	6	9	5
world	3	7	1	0	4	10

- The **correct classification rate** per class was:

$$[0.84 \quad 0.88 \quad 0.52 \quad 0.92 \quad 0.36 \quad 0.40]$$

Cross-Validation

- Cross-validation is a technique for assessing how the results of a statistical analysis will **generalize** to an independent data set.

- In this lab, you have access to 25 songs per genre, so 25x6=150 total songs.

$$\mathcal{S} = \bigcup_{i=1}^{6} \mathcal{G}_i$$

Cross-Validation

- Randomly divide each genre into 5 subsets of size 5:

$$\mathcal{G}_i = \bigcup_{k=1}^{5} \mathcal{G}_i^k$$

- where: $|\mathcal{G}_i^k| = |\mathcal{G}_i|/5 = 5$

Cross-Validation

for $k = 1$ to 5 `// round robin over the testing songs`

– form the set of test songs

$$\mathcal{U} = \bigcup_{i=1}^{6} \{\mathcal{G}_i^k,\}$$

and the set of training songs

$$\mathcal{L} = \bigcup_{i=1}^{6} \{\mathcal{G}_i^l,\ l = 1, \cdots, 5, l \neq k\}$$

– test the performance of your algorithm on the 30 songs in $\mathcal{U}$ using the 120 training songs $\mathcal{L}$

– record the confusion matrix

Cross-Validation

- Finally, repeat this procedure 10 different times over various randomizations.

- You will compute the **mean** and **standard** deviation for all the entries in the confusion matrices

- for each entry (i, j) of the confusion matrix:
  - compute the average $\overline{R}(i, j)$
  - compute the standard deviation $\sigma_R(i, j)$

Random Permutation in MATLAB

Description

`p = randperm(n)` returns a row vector containing a random permutation of the integers from 1 to n inclusive.

```
>> randperm(10)

ans =

     6     3     7     8     5     1     2     4     9    10
```

`rng(seed)` seeds the random number generator using the nonnegative integer `seed` so that `rand`, `randi`, and `randn` produce a predictable sequence of numbers.

Cross-Validation in MATLAB

crossvalind

Generate cross-validation indices

Syntax

```
Indices = crossvalind('Kfold', N, K)
[Train, Test] = crossvalind('HoldOut', N, P)
[Train, Test] = crossvalind('LeaveMOut', N, M)
[Train, Test] = crossvalind('Resubstitution', N, [P,Q])
[...] = crossvalind(Method, Group, ...)
[...] = crossvalind(Method, Group, ..., 'Classes', C)
[...] = crossvalind(Method, Group, ..., 'Min', MinValue)
```

Cross-Validation in MATLAB

```
>> N = 25; [Train, Test] = crossvalind('LeaveMOut', N, 5);
>> size(Train)
        ans =   25    1
>> size(Test)
        ans =   25    1
>> size(find(Train==1),1)
        ans =   20
>> size(find(Test==1),1)
        ans =    5
```

Property of a distance between tracks

- The distance should provide a natural **partitioning** of the dataset.
- Let X_i be the set of features that you compute for track i.

1. for a given genre g, we define its centroid as

$$\overline{\boldsymbol{X}}_g = \frac{1}{25} \sum_{k \in \text{genre } g} \boldsymbol{X}_k$$

2. We also define the radius ρ_g of a genre g as the standard deviation of the genre

$$\rho_g = \frac{1}{25} \sum_{k \in \text{genre } g} d(\boldsymbol{X}_k, \overline{\boldsymbol{X}}_g)$$

Property of a distance between tracks

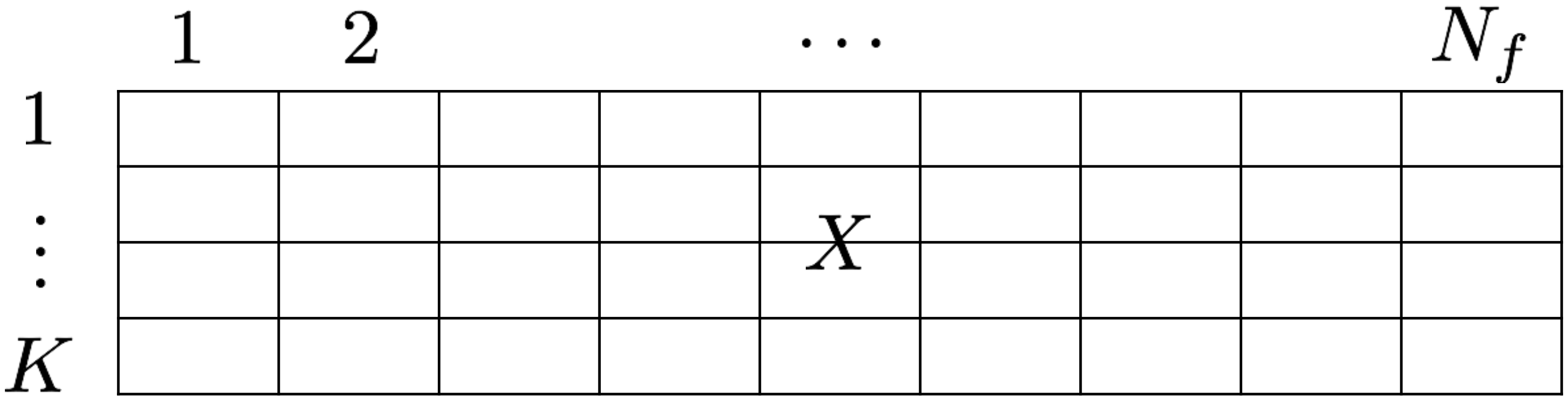
3. Finally, we require that the genres do not overlap, and therefore if we consider two genres, g and g',

$$d(\overline{\boldsymbol{X}}_g, \overline{\boldsymbol{X}}_{g'}) > \rho_g + \rho_{g'} \tag{4}$$

Distances Between Tracks

- The key ingredient of classification methods is a distance that quantifies the similarity between tracks in terms of musical features.

- We consider the following features:
  - MFCC coefficients
  - Chroma (Normalized Pitch Class Profile)



Resizing MFCC Coefficient

- In order to obtain more reliable classification results and speed up the computation, we merge some of the Mel banks:

```
t = zeros(1,36); (2.21)
t(1) =1;t(7:8)=5;t(15:18)= 9;
t(2)   = 2; t( 9:10) = 6; t(19:23) = 10;
t(3:4) = 3; t(11:12) = 7; t(24:29) = 11;
t(5:6) = 4; t(13:14) = 8; t(30:36) = 12;

mel2 = zeros(12,size(mfcc,2));

for i=1:12,
    mel2(i,:) = sum(mfcc(t==i,:),1);
end
```

Resizing MFCC Coefficient

```
>> t = [1,1,1,1,2];
>> t == 1

ans =

     1     1     1     1     0

>> t(ans)

ans =

     1     1     1     1
```

Resizing MFCC Coefficient

```
>> A = [1 3 2; 4 2 5; 6 1 4]

A =

     1     3     2
     4     2     5
     6     1     4

>> sum(A,2)

ans =

     6
    11
    11

>> sum(A,1)

ans =

    11     6    11
```

A Probabilistic Model

- For each track, the distribution of vectors X(:, n), n = 1,..., N_f is modeled as a multivariate Gaussian distribution in $\mathbb{R}^K, K = 12$.

- Therefore, this approach collapses all the frames together and summarizes each track by a mean and a covariance matrix.

A Probabilistic Model

```
>> mu = mean(mfcc,2);
>> Cov = cov(mfcc');
```

A Probabilistic Model

```
>> help cov
 cov Covariance matrix.
    cov(X), if X is a vector, returns the variance.  For matrices,
    where each row is an observation, and each column a variable,
    cov(X) is the covariance matrix.  DIAG(cov(X)) is a vector of
    variances for each column, and SQRT(DIAG(cov(X))) is a vector
    of standard deviations. cov(X,Y), where X and Y are matrices with
    the same number of elements, is equivalent to cov([X(:) Y(:)]).
```

Distance in Probabilistic Model

- First, we compute the Kullback-Leibler divergence:

$$KL(G^s, G^{s'}) = \frac{1}{2}\left(\text{tr}(\Sigma_{s'}^{-1}\Sigma_s) + (\mu_{s'} - \mu_s)^T \Sigma_{s'}^{-1}(\mu_{s'} - \mu_s) - K + \log\left(\frac{\det \Sigma_{s'}}{\det \Sigma_s}\right)\right)$$

Symmetric KL Divergence

$$KL(G^s, G^{s'}) = \frac{1}{2}\mathrm{tr}(\Sigma_{s'}^{-1}\Sigma_s + \Sigma_s^{-1}\Sigma_{s'}) - K + \frac{1}{2}(\mu_s - \mu_{s'})^T \left(\Sigma_{s'}^{-1} + \Sigma_s^{-1}\right)(\mu_s - \mu_{s'})$$

Distance in Probabilistic Model

- Second, the KL distance is rescaled using an exponential kernel, and we define the distance between two tracks as follows:

$$d(s, s') = \exp\left(-\gamma KL(G^s, G^{s'})\right)$$

- In this kernel, the parameter γ is chosen in [0,1] to optimize the classification.

Distance Matrix

- We Compute the 6x6 average distance matrix between the genres, defined by:

$$\overline{D}(i,j) = \frac{1}{25^2} \sum_{s \in \text{genre } i,\ s' \in \text{genre } j} d(s, s'), \quad i, j = 1, \ldots, 6$$

- Now, you find a value of γ that maximizes the separation between the different genres.

Appendix 1: Singular Value Decomposition (SVD)

- There exists a factorization of matrix M:

$$M = U \underbrace{\Sigma}_{\text{diagonal matrix}} V^T$$

Appendix 1: Singular Value Decomposition (SVD)

- What is the application of SVD?

```
M =

     1     2     3
     0     0     0
     0     0     0
```

```
[U,Sigma,V] = svd(M);
```

```
Sigma =

    3.7417         0         0
         0         0         0
         0         0         0
```

```
U =

    1.0000         0         0
         0         0    1.0000
         0    1.0000         0
```

Appendix 2: Pseudo Inverse

- The SVD can be used for computing the pseudo inverse of a matrix:

$$M^{+} = V\Sigma^{+}U^{T}$$

replacing every non-zero diagonal entry by its inverse