Introduction to Audio Content Analysis

Module 3.7: Feature Postprocessing

alexander lerch

Georgia Tech Center for Music Technology
College of Design

corresponding textbook section

Section 3.7

- **lecture content**
  - derived features
  - feature aggregation
  - feature normalization
  - problems of dimensionality
  - feature selection
  - feature transformation/mapping

- **learning objectives**
  - discuss the advantages of specific derived features
  - summarize the principles of feature aggregation
  - list two forms of feature normalization and explain their usefulness
  - describe potential challenges with high-dimensional feature spaces

corresponding textbook section

Section 3.7

- **lecture content**
  - derived features
  - feature aggregation
  - feature normalization
  - problems of dimensionality
  - feature selection
  - feature transformation/mapping

- **learning objectives**
  - discuss the advantages of specific derived features
  - summarize the principles of feature aggregation
  - list two forms of feature normalization and explain their usefulness
  - describe potential challenges with high-dimensional feature spaces

feature post-processing

introduction 1/2

- extracting multiple instantaneous features leads to
  - $\rightarrow$ one feature vector per block, or
  - $\rightarrow$ one feature matrix per audio file

$$
\begin{aligned}
\boldsymbol{V} &= [\boldsymbol{v}(0)\ \boldsymbol{v}(1)\ \ldots\ \boldsymbol{v}(\mathcal{N}-1)] \\
&= \begin{bmatrix} v_0(0) & v_0(1) & \ldots & v_0(\mathcal{N}-1) \\ v_1(0) & v_1(1) & \ldots & v_1(\mathcal{N}-1) \\ \vdots & \vdots & \ddots & \vdots \\ v_{\mathcal{F}-1}(0) & v_{\mathcal{F}-1}(1) & \ldots & v_{\mathcal{F}-1}(\mathcal{N}-1) \end{bmatrix}
\end{aligned}
$$

dimensions: $\mathcal{F} \times \mathcal{N}$ (number of features and number of blocks, resp.)

feature post-processing
introduction 2/2

multiple options for feature matrix processing:

1. derive additional features
2. aggregate existing features (e.g., one feature vector per file)
3. ensure similar scale and distribution

feature post-processing

examples of derived features

- **diff**: use the change in value

$$v_{j,\Delta}(n) = v_j(n) - v_j(n-1)$$

- **smoothed**: remove high frequency content by low-pass filtering
  - (anticausal) single-pole

$$v_{j,\mathrm{LP}}(n) = (1-\alpha)\cdot v_j(n) - \alpha \cdot v_{j,\mathrm{LP}}(n-1)$$

  - moving average

feature post-processing

examples of derived features

- **diff**: use the change in value

$$v_{j,\Delta}(n) = v_j(n) - v_j(n-1)$$

- **smoothed**: remove high frequency content by low-pass filtering
    - (anticausal) single-pole

$$v_{j,\mathrm{LP}}(n) = (1-\alpha)\cdot v_j(n) - \alpha \cdot v_{j,\mathrm{LP}}(n-1)$$

    - moving average

feature post-processing

feature normalization

- **reasons**
  - features have different ranges and distributions
  - ensure that one feature does not have outsized impact
- **z-score normalization**

$$v_{j,\mathrm{N}}(n) = \frac{v_j(n) - \mu_{v_j}}{\sigma_{v_j}}.$$

- **min-max normalization**

$$v_{j,\mathrm{N}}(n) = \frac{v_j(n) - \min(v_j)}{\max(v_j) - \min(v_j)}.$$

T

he normalization constants $\mu_{v_j}, \sigma_{v_j}, \max(v_j), \min(v_j)$ have to be estimated from the *Training Set*. The same (training) constants are then applied during inference. Extracting constants from the *Test Set* is meaningless as the system has to infer with

feature post-processing

feature aggregation

feature aggregation:[1] compute *summary features* from feature series ⇒ **subfeatures**

- **reasons**
  - only one feature vector required per file
  - data reduction
  - characteristics of distribution or change over time contain additional info
- **examples**
  - *statistical descriptors*
    - mean, median, max, standard deviation
  - *hand crafted*
    - anything that might be meaningful — periodicity, slope, ...

[1] also compare *pooling* operation in machine learning

feature post-processing
feature aggregation

- could be for whole file or **texture window**:
  split feature series in overlapping blocks of a few seconds length
- could be **hierarchical** process:
  1. compute subfeatures per window
  2. compute subfeatures of subfeature series
  3. (go to 1.)

feature post-processing
feature aggregation

- could be for whole file or **texture window**:
  split feature series in overlapping blocks of a few seconds length

- could be **hierarchical** process:
  1. compute subfeatures per window
  2. compute subfeatures of subfeature series
  3. (go to 1.)

- **problem**
  - many ML approaches cannot cope with large amounts of irrelevant features
  - ML algorithms might degrade in performance

- **advantages**
  - reducing storage requirements
  - reducing training complexity
  - defying the "curse of dimensionality"

- **disadvantages**
  - additional workload for reduction
  - adding an additional layer of model complexity

introduction
dimensionality reduction

- **problem**
  - many ML approaches cannot cope with large amounts of irrelevant features
  - ML algorithms might degrade in performance
- **advantages**
  - reducing storage requirements
  - reducing training complexity
  - defying the "curse of dimensionality"
- **disadvantages**
  - additional workload for reduction
  - adding an additional layer of model complexity

introduction
dimensionality reduction

- **problem**
  - many ML approaches cannot cope with large amounts of irrelevant features
  - ML algorithms might degrade in performance
- **advantages**
  - reducing storage requirements
  - reducing training complexity
  - defying the "curse of dimensionality"
- **disadvantages**
  - additional workload for reduction
  - adding an additional layer of model complexity

introduction
dimensionality issues

problems of high-dimensional data:

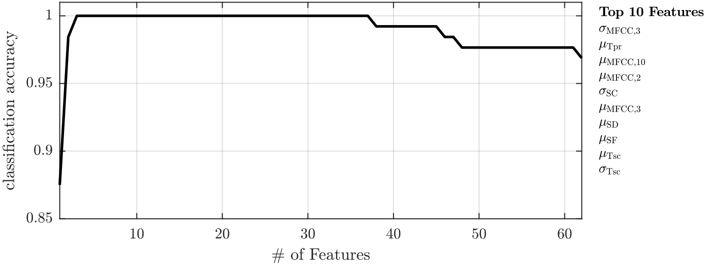
- increase in run-time
- overfitting
- curse of dimensionality
- required amount of training samples

introduction
dimensionality issues

problems of high-dimensional data:

- increase in run-time
- overfitting
- curse of dimensionality
- required amount of training samples

$\Rightarrow$ increasing number of input features may *decrease* classification performance



matlab source: plotSequentialForwardSelection.m

dimensionality issues
overfitting

Georgia Tech Center for Music Technology College of Design

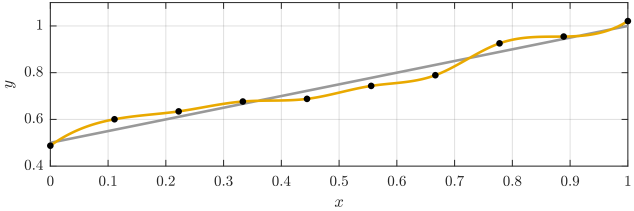
- **overfitting**:
  - lack of training data
  - overly complex model

$\Rightarrow$ model cannot be estimated properly

- required training set size depends on
  - classifier and its parametrization
  - number of classes
  - ...
- *rule of thumb*:
  don't bother with training sets smaller than $\mathcal{F}^2$

dimensionality issues
overfitting

- **overfitting**:
  - lack of training data
  - overly complex model
  - ⇒ model cannot be estimated properly



- required training set size depends on
  - classifier and its parametrization
  - number of classes

matlab source: plotOverfitting.m

dimensionality issues
overfitting

- **overfitting**:
  - lack of training data
  - overly complex model

  $\Rightarrow$ model cannot be estimated properly

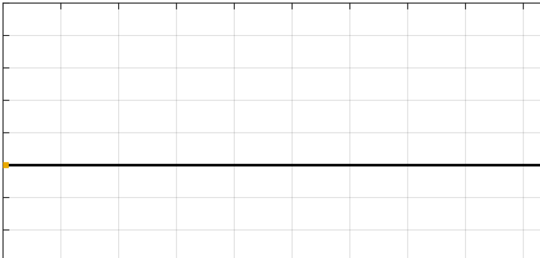
  - required training set size depends on
    - classifier and its parametrization
    - number of classes
    - ...
  - *rule of thumb*:
    don't bother with training sets smaller than $\mathcal{F}^2$

dimensionality issues

curse of dimensionality

- **curse of dimensionality**:
  - increasing dimensionality leads to sparse training data
  - neighborhoods of data points become less concentrated
  - model tends to be harder to estimate in higher-dimensional space
  - applies to distance-based algorithms

- **example** (uniformly distributed data)
  - identify region on axis covering **1% of data**
    - 1-D: 1% of x-axis
    - 2-D: 10% of x-axis/y-axis
    - 3-D: 21.5% of x-axis/y-axis/z-axis
    - 10-D: 63%
    - 100-D: 95%



'seOfDimensionality.m

dimensionality reduction
introduction

- **feature subset selection**:
  discard least helpful features
  - high "discriminative" or descriptive power
  - non-correlation to other features
  - invariance to irrelevancies

- **feature space transformation**:
  map feature space

dimensionality reduction

introduction

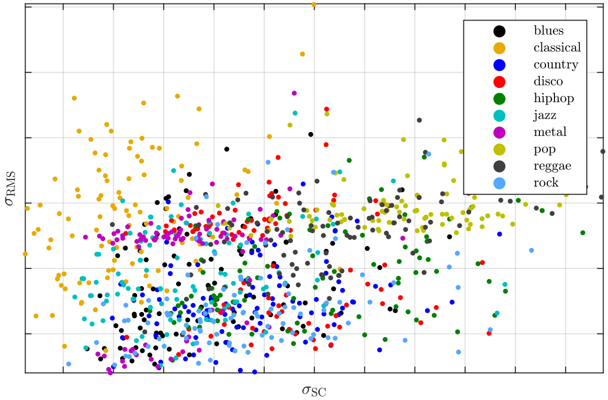
- **feature subset selection**:
  discard least helpful features
  - high "discriminative" or descriptive power
  - non-correlation to other features
  - invariance to irrelevancies

- **feature space transformation**:
  map feature space

feature subset selection

manual feature selection

example scatter plots of pairs of features in a multi-class scenario



matlab source: plotFeatureScatter.m

feature subset selection

introduction

1. **wrapper methods**:
   - *description*
     - use the "classifier" itself to evaluate feature performance
   - *advantages*
     - taking into account feature dependencies
     - model dependency
   - *disadvantages*
     - complexity
     - risk of overfitting
2. **filter methods**:
   - *description*
     - use an objective function
   - *advantages*
     - easily scalable
     - independent of classification algorithm
   - *disadvantages*
     - no interaction with classifier
     - no feature dependencies

feature subset selection
introduction

1. **wrapper methods**:
   - *description*
     - use the "classifier" itself to evaluate feature performance
   - *advantages*
     - taking into account feature dependencies
     - model dependency
   - *disadvantages*
     - complexity
     - risk of overfitting
2. **filter methods**:
   - *description*
     - use an objective function
   - *advantages*
     - easily scalable
     - independent of classification algorithm
   - *disadvantages*
     - no interaction with classifier
     - no feature dependencies

feature subset selection
wrapper methods 1/2

1. **single variable classification**:
   - *procedure*
     - evaluate each feature individually
     - choose the top N
   - *complexity*
     - subsets to test: $\mathcal{F}$
   - *challenges*
     - inter-feature correlation is not considered
     - feature combinations are not considered
2. **brute force subset selection**
   - *procedure*
     - evaluate all possible feature combinations
     - choose the optimal combination
   - *complexity*
     - subsets to test: $2^{\mathcal{F}}$

feature subset selection

wrapper methods 1/2

1. **single variable classification**:
   - *procedure*
     - evaluate each feature individually
     - choose the top N
   - *complexity*
     - subsets to test: $\mathcal{F}$
   - *challenges*
     - inter-feature correlation is not considered
     - feature combinations are not considered
2. **brute force subset selection**
   - *procedure*
     - evaluate all possible feature combinations
     - choose the optimal combination
   - *complexity*
     - subsets to test: $2^{\mathcal{F}}$

feature subset selection
wrapper methods 2/2

4 sequential forward selection

- *procedure*
  1. init: empty feature subset $\mathcal{V}_\mathrm{s} = \emptyset$
  2. find feature v_j maximizing objective function
$$v_j = \operatorname*{argmax}_{\forall j | v_j \notin \mathcal{V}_\mathrm{s}} J(\mathcal{V}_\mathrm{s} \bigcup v_j)$$
  3. add feature v_j to $\mathcal{V}_\mathrm{s}$
  4. go to step 2

5 sequential backward elimination

- *procedure*
  1. init: full feature set
  2. find feature v_j with the least impact on objective function
  3. discard feature v_j
  4. go to step 2

feature subset selection
wrapper methods 2/2

4 sequential forward selection

- *procedure*
  1. init: empty feature subset $\mathcal{V}_\mathrm{s} = \emptyset$
  2. find feature v_j maximizing objective function

$$v_j = \underset{\forall j | v_j \notin \mathcal{V}_\mathrm{s}}{\mathrm{argmax}}\, J(\mathcal{V}_\mathrm{s} \bigcup v_j)$$

  3. add feature v_j to $\mathcal{V}_\mathrm{s}$
  4. go to step 2

5 sequential backward elimination

- *procedure*
  1. init: full feature set
  2. find feature v_j with the least impact on objective function
  3. discard feature v_j
  4. go to step 2

feature space transformation
PCA introduction

- **objective**
  - map features to new coordinate system

$$\boldsymbol{u}(n) = \boldsymbol{T}^{\mathrm{T}} \cdot \boldsymbol{v}(n)$$

    - $\boldsymbol{u}(n)$: transformed features (same dimension as $\boldsymbol{v}(n)$)
    - $\boldsymbol{T}$: transformation matrix ($\mathcal{F} \times \mathcal{F}$)

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{c}_0 & \boldsymbol{c}_1 & \dots & \boldsymbol{c}_{\mathcal{F}-1} \end{bmatrix}$$

- **properties**
  - $\boldsymbol{c}_0$ points in the direction of highest *variance*
  - variance concentrated in as few output components as possible
  - $\boldsymbol{c}_i$ orthogonal

$$\boldsymbol{c}_i^{\mathrm{T}} \cdot \boldsymbol{c}_j = 0 \quad \forall\ i \neq j$$

  - transformation is invertible

$$\boldsymbol{v}(n) = \boldsymbol{T} \cdot \boldsymbol{u}(n)$$

feature space transformation

PCA introduction

- **objective**
  - map features to new coordinate system

$$\boldsymbol{u}(n) = \boldsymbol{T}^{\mathrm{T}} \cdot \boldsymbol{v}(n)$$

    - $\boldsymbol{u}(n)$: transformed features (same dimension as $\boldsymbol{v}(n)$)
    - $\boldsymbol{T}$: transformation matrix $(\mathcal{F} \times \mathcal{F})$

$$\boldsymbol{T} = \left[\begin{array}{cccc} \boldsymbol{c}_0 & \boldsymbol{c}_1 & \ldots & \boldsymbol{c}_{\mathcal{F}-1} \end{array} \right]$$

- **properties**
  - $\boldsymbol{c}_0$ points in the direction of highest *variance*
  - variance concentrated in as few output components as possible
  - $\boldsymbol{c}_i$ orthogonal

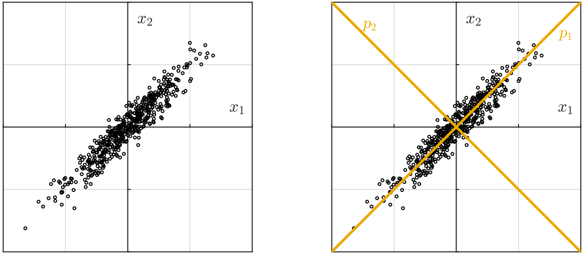
$$\boldsymbol{c}_i^{\mathrm{T}} \cdot \boldsymbol{c}_j = 0 \quad \forall \; i \neq j$$

  - transformation is invertible

$$\boldsymbol{v}(n) = \boldsymbol{T} \cdot \boldsymbol{u}(n)$$

feature space transformation

PCA visualization



calculation of the transformation matrix

1. compute covariance matrix $\boldsymbol{R}$

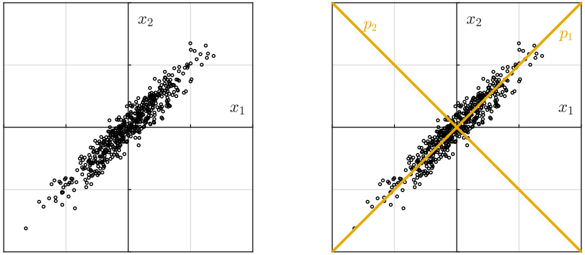
$$\boldsymbol{R} = \mathcal{E}\{(V - \mathcal{E}\{V\})(V - \mathcal{E}\{V\})\}$$

2. choose eigenvectors as axes for the new coordinate system

matlab source: plotPca.m

feature space transformation

PCA visualization



calculation of the transformation matrix

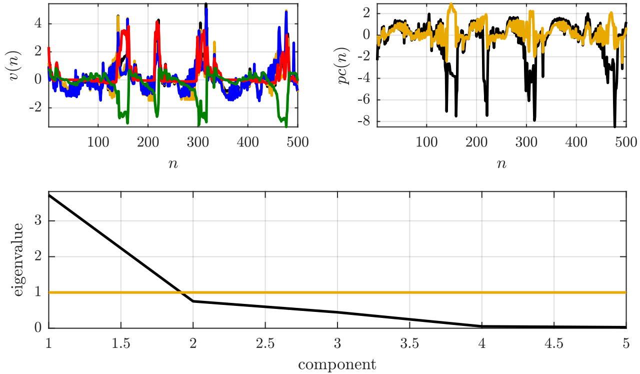
1. compute covariance matrix $\boldsymbol{R}$

$$\boldsymbol{R} = \mathcal{E}\{(V - \mathcal{E}\{V\})(V - \mathcal{E}\{V\})\}$$

2. choose eigenvectors as axes for the new coordinate system

matlab source: plotPca.m

introduction
PCA example



matlab source: plotPcaExample.m

pca transformation matrix

$$\begin{bmatrix} -0.4187 & 0.3467 & -0.4569 & 0.4143 & -0.1271 & -0.5549 \\ -0.3908 & 0.1815 & 0.8136 & -0.0289 & 0.2060 & -0.3304 \\ -0.4516 & 0.3384 & 0.0859 & 0.2413 & -0.2919 & 0.7285 \\ -0.4337 & 0.1699 & -0.3337 & -0.7243 & 0.3747 & 0.0816 \\ 0.3802 & 0.5599 & -0.0381 & 0.2808 & 0.6622 & 0.1524 \\ 0.3679 & 0.6245 & 0.0956 & -0.4071 & -0.5267 & -0.1495 \end{bmatrix}$$

introduction
PCA example

introduction
PCA example

pca transformation matrix

$$\begin{bmatrix} -0.4187 & 0.3467 & -0.4569 & 0.4143 & -0.1271 & -0.5549 \\ -0.3908 & 0.1815 & 0.8136 & -0.0289 & 0.2060 & -0.3304 \\ -0.4516 & 0.3384 & 0.0859 & 0.2413 & -0.2919 & 0.7285 \\ -0.4337 & 0.1699 & -0.3337 & -0.7243 & 0.3747 & 0.0816 \\ 0.3802 & 0.5599 & -0.0381 & 0.2808 & 0.6622 & 0.1524 \\ 0.3679 & 0.6245 & 0.0956 & -0.4071 & -0.5267 & -0.1495 \end{bmatrix}$$

summary

lecture content

- **feature matrix should be processed to adapt to task and classifier**
  - derive additional features
  - aggregate features
  - normalize features
- **derived features**
  - take existing features and "create" new ones
- **aggregate features: subfeatures**
  - combine blocks of features by computing, e.g., statistical features from them (mean, standard deviation, ...)
  - subfeature vector is used as classifier input or as intermediate feature series
- **feature normalization**
  - avoid different value ranges might impacting classifier
  - handle different feature distributions