Introduction to Audio Content Analysis

Module 3.7.1: Feature Post-Processing

alexander lerch

Georgia Tech Center for Music Technology
College of Design

corresponding textbook section

Section 3.7.1–3.7.3

- **lecture content**
  - derived features
  - feature aggregation
  - feature normalization

- **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

corresponding textbook section

Section 3.7.1–3.7.3

- **lecture content**
  - derived features
  - feature aggregation
  - feature normalization

- **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

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

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

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

- **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.)

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