Introduction to Audio Content Analysis

Module 5.0: Data, Data Splits, and Augmentation

alexander lerch

introduction
overview

---

corresponding textbook section

Section 5

- **lecture content**
    - data requirements
    - data splits for train and test
    - N-Fold cross-validation
    - data augmentation

- **learning objectives**
    - understand the importance of data in machine learning
    - define task-specific data requirements
    - discuss possibilities of data augmentation
    - implement N-Fold cross-validation in Python

introduction
overview

corresponding textbook section

Section 5

- **lecture content**
  - data requirements
  - data splits for train and test
  - N-Fold cross-validation
  - data augmentation

- **learning objectives**
  - understand the importance of data in machine learning
  - define task-specific data requirements
  - discuss possibilities of data augmentation
  - implement N-Fold cross-validation in Python

machine learning
data-driven

- derive classification parameters from data, e.g.,
$\Rightarrow$ learn feature distributions/separation metrics per class

- typical 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.)

overview
○

intro
●

data
○○○○○

augmentation
○

summary
○

machine learning
data-driven

Georgia Tech | Center for Music Technology
College of Design

- derive classification parameters from data, e.g.,
- ⇒ learn feature distributions/separation metrics per class

- typical 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.)

overview

intro
●

data
○○○○○

augmentation
○

summary
○

machine learning
data-driven

- derive classification parameters from data, e.g.,
$\Rightarrow$ learn feature distributions/separation metrics per class

- typical 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.)

machine learning
data-driven

- derive classification parameters from data, e.g.,

$\Rightarrow$ learn feature distributions/separation metrics per class

- typical 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.)

overview
○

intro
●

data
○○○○○

augmentation
○

summary
○

machine learning
data-driven

- derive classification parameters from data, e.g.,
$\Rightarrow$ learn feature distributions/separation metrics per class

- typical 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.)

overview
○

intro
○

data
●○○○○

augmentation
○

summary
○

data
requirements

Georgia | Center for Music
Tech | Technology
College of Design

what are important properties of our data

data
requirements

what are important properties of our data

- **representative**
  - represent all necessary factors of input data (e.g., range of genres, audio qualities, musical complexity, etc.)
  - unbiased representation of class balance/label distribution
- **clean**, non-noisy
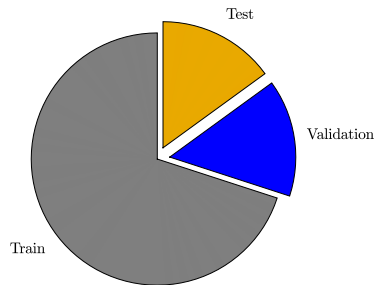 - potential issues with subjective tasks
- **sufficient**
  - complex tasks/systems require lots of data

overview · intro · data ○●○○○ · augmentation ○ · summary ○

data
data split

Georgia **|** Center for Music
Tech **|** Technology
College of Design

- a bigger data set is commonly split in subsets
  - **training data** ($\approx 70 - 80\%$)
    - ▶ used to build the machine learning model
  - **validation data** ($\approx 10 - 15\%$)
    - ▶ used to tweak model parameters
  - **testing data** ($\approx 10 - 15\%$)
    - ▶ used to evaluate the model
    - ▶ needs to be **unseen**!

- no overlap between subsets!
  - also make sure that similar content (from one recording, album, artist, ...) is grouped into **one subset only**
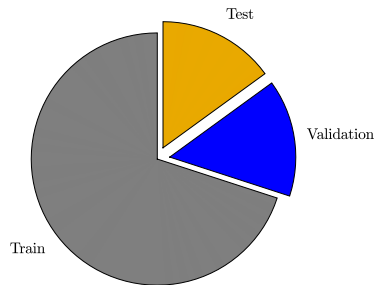

matlab source: plotDataSplit.m

data
data split

Georgia Tech | **Center for Music Technology**
College of Design

- a bigger data set is commonly split in subsets
  - **training data** ($\approx 70 - 80\%$)
    - ▶ used to build the machine learning model
  - **validation data** ($\approx 10 - 15\%$)
    - ▶ used to tweak model parameters
  - **testing data** ($\approx 10 - 15\%$)
    - ▶ used to evaluate the model
    - ▶ needs to be **unseen**!

- no overlap between subsets!
  - also make sure that similar content (from one recording, album, artist, ...) is grouped into **one subset only**

matlab source: plotDataSplit.m

data
N-Fold cross validation 1/2

- trying to utilize ALL data as both training and testing data
- special case: Leave One Out CV
- tends to be time-consuming

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

Georgia **Center for Music**
Tech **Technology**
College of Design

- trying to utilize ALL data as both training and testing data
- special case: Leave One Out CV
- tends to be time-consuming

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

data
N-Fold cross validation 1/2

Georgia **Center for Music**
Tech **Technology**
College of Design

- trying to utilize ALL data as both training and testing data
- special case: Leave One Out CV
- tends to be time-consuming

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

data
N-Fold cross validation 1/2

Georgia | Center for Music
Tech | Technology
College of Design

- trying to utilize ALL data as both training and testing data
- special case: Leave One Out CV
- tends to be time-consuming

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

overview
○

intro
○

data
○○●○○

augmentation
○

summary
○

data
N-Fold cross validation 1/2

- trying to utilize ALL data as both training and testing data
- special case: Leave One Out CV
- tends to be time-consuming

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

data
N-Fold cross validation 1/2

- trying to utilize ALL data as both training and testing data
- special case: Leave One Out CV
- tends to be time-consuming

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

overview

intro

data
00●00

augmentation

summary

data
N-Fold cross validation 1/2

Georgia | Center for Music
Tech | Technology
College of Design

- trying to utilize ALL data as both training and testing data
- special case: Leave One Out CV
- tends to be time-consuming

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

data
N-Fold cross validation 2/2

Data			

1.	Test	Train	Train	Train
2.	Train	Test	Train	Train
3.	Train	Train	Test	Train
4.	Train	Train	Train	Test

classification
interaction of data, features, and classifier

Georgia Tech | **Center for Music Technology**
College of Design

- **training set**
  - 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*

- **features**
  - poor features
    ⇒ *bad classification performance*

overview

intro

data
○○○○●

augmentation

summary

classification
interaction of data, features, and classifier

- **training set**
  - 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*

- **features**
  - poor features
    $\Rightarrow$ *bad classification performance*

classification
interaction of data, features, and classifier

- **training set**
  - 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*

- **features**
  - **poor features**
    ⇒ *bad classification performance*

data
augmentation

Georgia Tech | **Center for Music Technology**
College of Design

- if annotated data is insufficient, we can 'cheat' by increasing the amount of training data
- ⇒ **data augmentation**: apply irrelevant transforms to audio data
  - *data segmentation*
    - ▶ treat audio snippets as separate observations
  - *quality degradation*
    - ▶ add noise and distortion, limit bandwidth, etc.
  - *audio effects*
    - ▶ apply reverb, etc.
  - *changing pitch/tempo*
  - *combine data*
    - ▶ mix different audio inputs together (if labels can be "mixed")
  - *mask out parts of the signal*

data
augmentation

- if annotated data is insufficient, we can 'cheat' by increasing the amount of training data
- ⇒ **data augmentation**: apply irrelevant transforms to audio data
  - *data segmentation*
    - ▶ treat audio snippets as separate observations
  - *quality degradation*
    - ▶ add noise and distortion, limit bandwidth, etc.
  - *audio effects*
    - ▶ apply reverb, etc.
  - *changing pitch/tempo*
  - *combine data*
    - ▶ mix different audio inputs together (if labels can be "mixed")
  - *mask out parts of the signal*

data
augmentation

- if annotated data is insufficient, we can 'cheat' by increasing the amount of training data

⇒ **data augmentation**: apply irrelevant transforms to audio data
  - *data segmentation*
    - ▶ treat audio snippets as separate observations
  - *quality degradation*
    - ▶ add noise and distortion, limit bandwidth, etc.
  - *audio effects*
    - ▶ apply reverb, etc.
  - *changing pitch/tempo*
  - *combine data*
    - ▶ mix different audio inputs together (if labels can be "mixed")
  - *mask out parts of the signal*

overview
○
intro
○
data
○○○○○
augmentation
●
summary
○

data
augmentation

Georgia | Center for Music
Tech ⬛ Technology
College of Design

- if annotated data is insufficient, we can 'cheat' by increasing the amount of training data
- ⇒ **data augmentation**: apply irrelevant transforms to audio data
  - *data segmentation*
    - ▶ treat audio snippets as separate observations
  - *quality degradation*
    - ▶ add noise and distortion, limit bandwidth, etc.
  - *audio effects*
    - ▶ apply reverb, etc.
  - *changing pitch/tempo*
  - *combine data*
    - ▶ mix different audio inputs together (if labels can be "mixed")
  - *mask out parts of the signal*

data
augmentation

- if annotated data is insufficient, we can 'cheat' by increasing the amount of training data
- ⇒ **data augmentation**: apply irrelevant transforms to audio data
  - *data segmentation*
    - ▶ treat audio snippets as separate observations
  - *quality degradation*
    - ▶ add noise and distortion, limit bandwidth, etc.
  - *audio effects*
    - ▶ apply reverb, etc.
  - *changing pitch/tempo*
  - *combine data*
    - ▶ mix different audio inputs together (if labels can be "mixed")
  - *mask out parts of the signal*

Georgia | Center for Music
Tech | Technology
College of Design

- if annotated data is insufficient, we can 'cheat' by increasing the amount of training data
- ⇒ **data augmentation**: apply irrelevant transforms to audio data
  - *data segmentation*
    - ▶ treat audio snippets as separate observations
  - *quality degradation*
    - ▶ add noise and distortion, limit bandwidth, etc.
  - *audio effects*
    - ▶ apply reverb, etc.
  - *changing pitch/tempo*
  - *combine data*
    - ▶ mix different audio inputs together (if labels can be "mixed")
  - *mask out parts of the signal*

overview
○

intro
○

data
○○○○○

augmentation
●

summary
○

data
augmentation

- if annotated data is insufficient, we can 'cheat' by increasing the amount of training data
- ⇒ **data augmentation**: apply irrelevant transforms to audio data
    - *data segmentation*
        - ▶ treat audio snippets as separate observations
    - *quality degradation*
        - ▶ add noise and distortion, limit bandwidth, etc.
    - *audio effects*
        - ▶ apply reverb, etc.
    - *changing pitch/tempo*
    - *combine data*
        - ▶ mix different audio inputs together (if labels can be "mixed")
    - *mask out parts of the signal*

summary
lecture content

- **data**
  - representative
  - clean, non-noisy
  - sufficient

- **data split**
  - train
  - validation
  - test

- **cross validation**
  - multiple runs with varying data splits
  - maximum data utilization