RUHR-UNIVERSITÄT BOCHUM

RUB

Music Classification Using Constant-Q Based Features

a library for mobile devices

Lena Brüder

January 5, 2013

Institute of
Communication Acoustics

Outline

Plan

Objectives

- Create a program that helps exploring music collections
- Derive all classification features from the Constant Q transform
- Design program as a library that runs on both a PC and on embedded devices ($\rightarrow$ Blackberry Playbook)

General approach to music classification

signal → frequency domain → signal features → classification → result

MP3,
WAV,
FLAC,
...

Constant Q transform

Length,
dynamic range,
tempo,
timbre,
chroma

Gaussian model

General approach to music classification

signal → frequency domain → signal features → classification → result

MP3,
WAV,
FLAC,
...

Constant Q transform

Length,
dynamic range,
tempo,
timbre,
chroma

Gaussian model

General approach to music classification

signal → frequency domain → signal features → classification → result

MP3, WAV, FLAC, ...	Constant Q transform	Length, dynamic range, tempo, timbre, chroma	Gaussian model

General approach to music classification

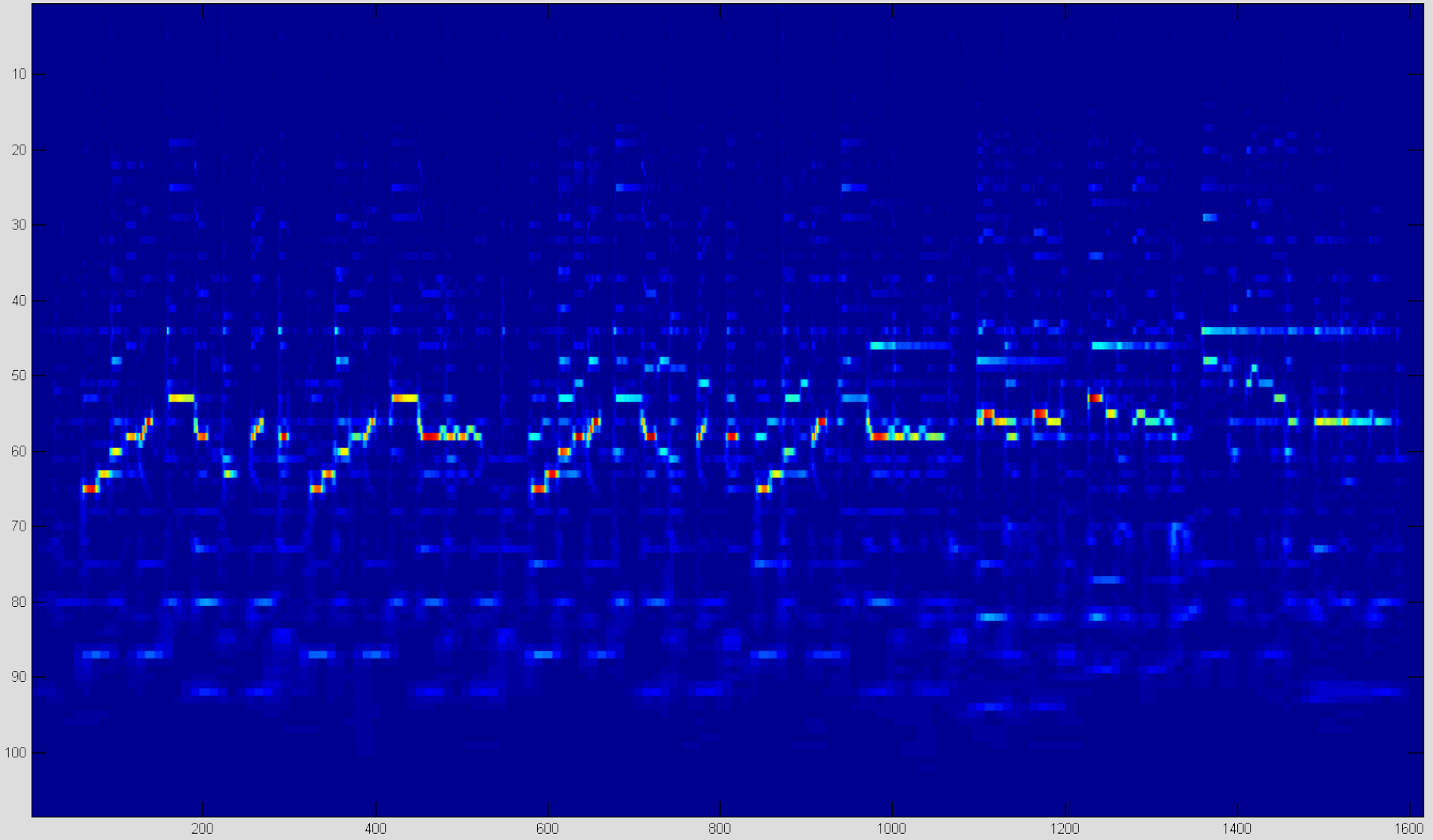
MP3, WAV, FLAC, ...	Constant Q transform	Length, dynamic range, tempo, timbre, chroma	Gaussian model

General approach to music classification

MP3, WAV, FLAC, ...	Constant Q transform	Length, dynamic range, tempo, timbre, chroma	Gaussian model

Plan

Constant-Q-Transform: Intuition



Constant Q transform: Definition

Let $x(n)$ be a discrete time-domain signal, f_k the center frequency of bin k and B the frequency bin count per octave. N_k is inversely proportional to f_k. f_s is the sampling rate and $w : \mathbb{R} \to \mathbb{R}, t \mapsto w(t)$ a continous window function with $t = 0$ for $t \notin [0, 1]$.

$$a_k(n) = \frac{1}{N_k} w\left(\frac{n}{N_k}\right) \exp\left(-\mathrm{j}2\pi n \frac{f_k}{f_s}\right) \tag{1}$$

are complex basis functions, time-frequency-atoms or temporal kernels. Then

$$X^{\mathrm{CQ}}(k, n) = \sum_{l=n-\lfloor \frac{N_k}{2} \rfloor}^{n+\lfloor \frac{N_k}{2} \rfloor} x(l) a_k^*\left(l - n + \frac{N_k}{2}\right) \tag{2}$$

is the constant Q transform of $x(n)$. The center frequencies f_k are defined as

$$f_k = f_1 2^{\frac{k-1}{B}}. \tag{3}$$

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Mel-frequency cepstrum)
- Key-invariant chroma (map all notes to one octave)

Note:

- Timbre and chroma are multi-dimensional features (vectors) not scalar values.
- Timbre and chroma are calculated every $10 - 20$ms (thousands of calculated ones per recording).
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Mel-frequency cepstrum)
- Key-invariant chroma (map an octave to one value, key)

Note:

- Timbre and chroma are multi-dimensional features (vectors) not scalar values
- Timbre and chroma are calculated every $10 - 20$ms (thousands calculated once per recording)
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Mel-frequency cepstrum)
- Key-invariant chroma (map all notes to one octave)

Note:

- Timbre and chroma are multi-dimensional features (matrix of scalar values)
- Timbre and chroma are calculated every $10 - 20$ms (thousands calculated once per recording)
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map an octave to one value each)

Note:

- Timbre and chroma are multi-dimensional features (→ 20 values per scalar value)
- Timbre and chroma are calculated every $10 - 20$ms (→ 6000 values calculated once per recording)
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value (→ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (how an octave is split into semitone steps)

Note:

- Timbre and chroma are multi-dimensional features (vectors) for single scalar values.
- Timbre and chroma are calculated every $10 - 20$ms (thousands of values calculated once per recording).
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma

Note:

- Timbre and chroma are multi-dimensional features
- Timbre and chroma are calculated every $10 - 20$ms
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features (matrix of scalar values).
- Timbre and chroma are calculated every $10 - 20$ms (hundreds per recording).
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features (vectors) not scalar values.
- Timbre and chroma are calculated every $10 - 20$ms (hundreds per recording).
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features (vectors) per scalar values.
- Timbre and chroma are calculated every $10 - 20$ms (thousands per recording).
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features (the others are scalar values).
- Timbre and chroma are calculated every $10 - 20$ms (the others are calculated once per recording).
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features, the others are scalar values.
- Timbre and chroma are calculated every $10 - 20$ms (→ thousands calculated ones per recording).
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value (→ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features, the others are scalar values.
- Timbre and chroma are calculated every $10 - 20$ms, the others are calculated once per recording.
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features, the others are scalar values.
- Timbre and chroma are calculated every $10 - 20$ms, the others are calculated once per recording.
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features, the others are scalar values.
- Timbre and chroma are calculated every $10 - 20$ms, the others are calculated once per recording.
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

Note:

- Timbre and chroma are multi-dimensional features, the others are scalar values.
- Timbre and chroma are calculated every $10 - 20$ms, the others are calculated once per recording.
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Feature Extraction

Different features are extracted:

- Length of the piece
- Dynamic range (how relate loud parts to quieter ones)
- Tempo in BPM (not used for classification)
- Timbre (via Constant-Q Cepstrum)
- Key-invariant chroma (map all octaves to one, remove key)

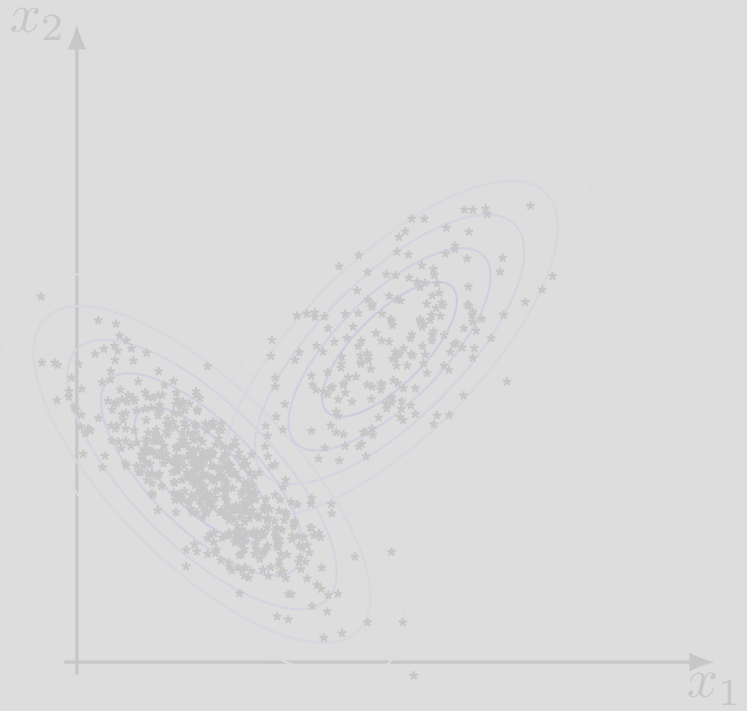
Note:

- Timbre and chroma are multi-dimensional features, the others are scalar values.
- Timbre and chroma are calculated every $10 - 20$ms, the others are calculated once per recording.
- But: Classifiers expect features to be uniform, or at least comparable.
- Solution: Transform many multi-dimensional feature vectors to one scalar value ($\rightarrow$ dimensionality and data count reduction).

Gaussian Mixture Models

Data count reduction works as follows:

- Take all feature vectors of one feature
- Model their probability distribution
- Forget about the original feature vectors
- This step brings the data count reduction: One model (fixed model size) instead of many (arbitrary count) feature vectors

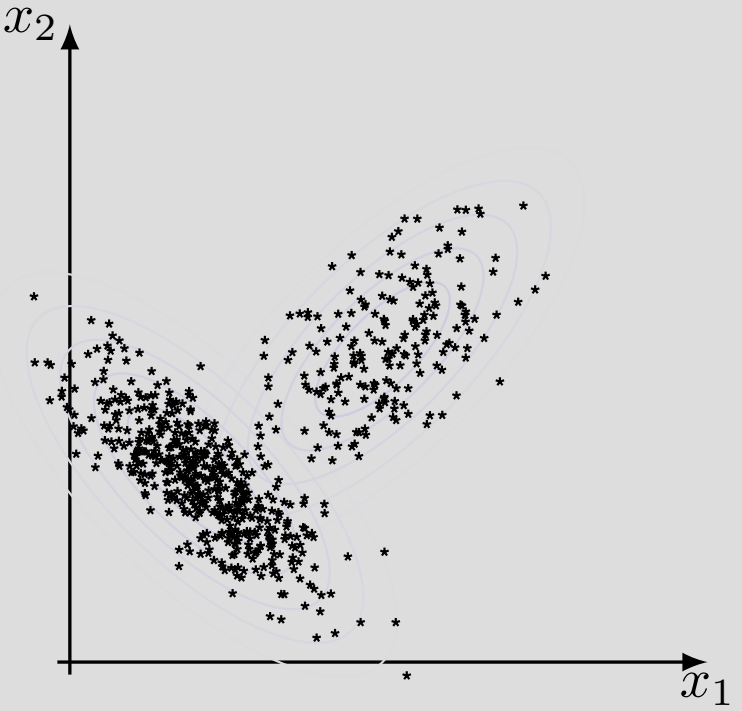


- Do this with feature vectors of one recording: Get a model for the recording
- Do this with feature vectors of all recordings from a category: Get a model for the category!

Gaussian Mixture Models

Data count reduction works as follows:

- Take all feature vectors of one feature
- Model their probability distribution
- Forget about the original feature vectors
- This step brings the data count reduction: One model (fixed model size) instead of many (arbitrary count) feature vectors

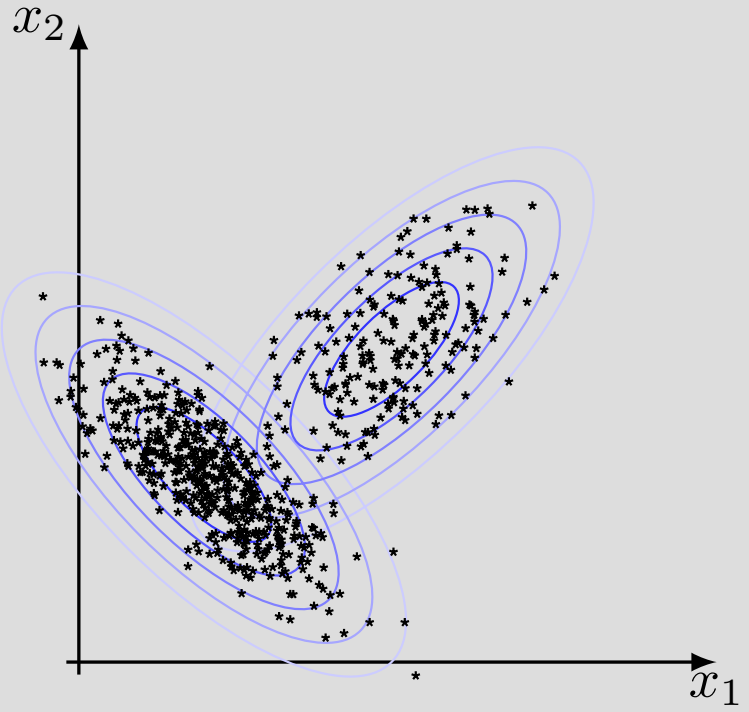


- Do this with feature vectors of one recording: Get a model for the recording
- Do this with feature vectors of all recordings from a category: Get a model for the category!

Gaussian Mixture Models

Data count reduction works as follows:

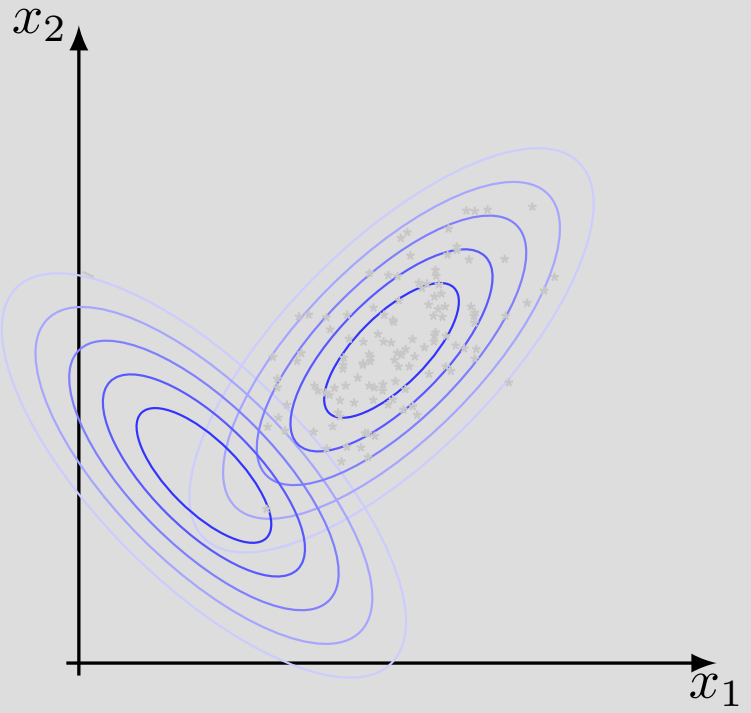
- Take all feature vectors of one feature
- Model their probability distribution
- Forget about the original feature vectors
- This step brings the data count reduction: One model (fixed model size) instead of many (arbitrary count) feature vectors
- Do this with feature vectors of one recording: Get a model for the recording
- Do this with feature vectors of all recordings from a category: Get a model for the category!



Gaussian Mixture Models

Data count reduction works as follows:

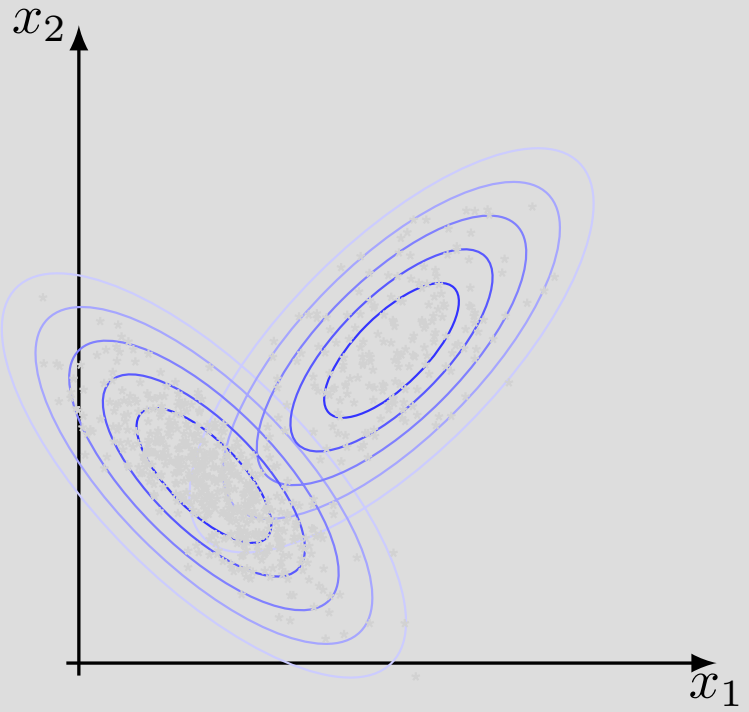
- Take all feature vectors of one feature
- Model their probability distribution
- Forget about the original feature vectors
- This step brings the data count reduction: One model (fixed model size) instead of many (arbitrary count) feature vectors



Gaussian Mixture Models

Data count reduction works as follows:

- Take all feature vectors of one feature
- Model their probability distribution
- Forget about the original feature vectors
- This step brings the data count reduction: One model (fixed model size) instead of many (arbitrary count) feature vectors

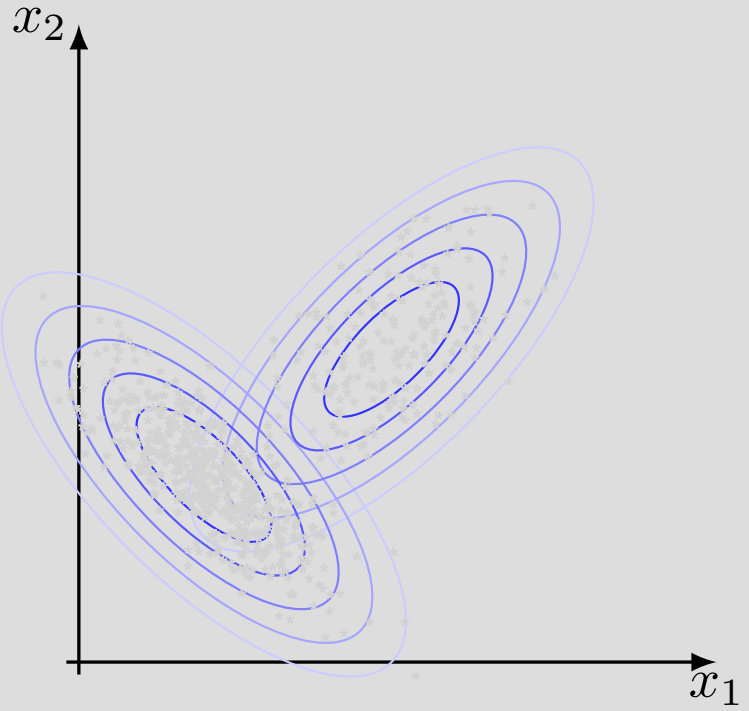


- Do this with feature vectors of one recording: Get a model for the recording
- Do this with feature vectors of all recordings from a category: Get a model for the category!

Gaussian Mixture Models

Data count reduction works as follows:

- Take all feature vectors of one feature
- Model their probability distribution
- Forget about the original feature vectors
- This step brings the data count reduction: One model (fixed model size) instead of many (arbitrary count) feature vectors

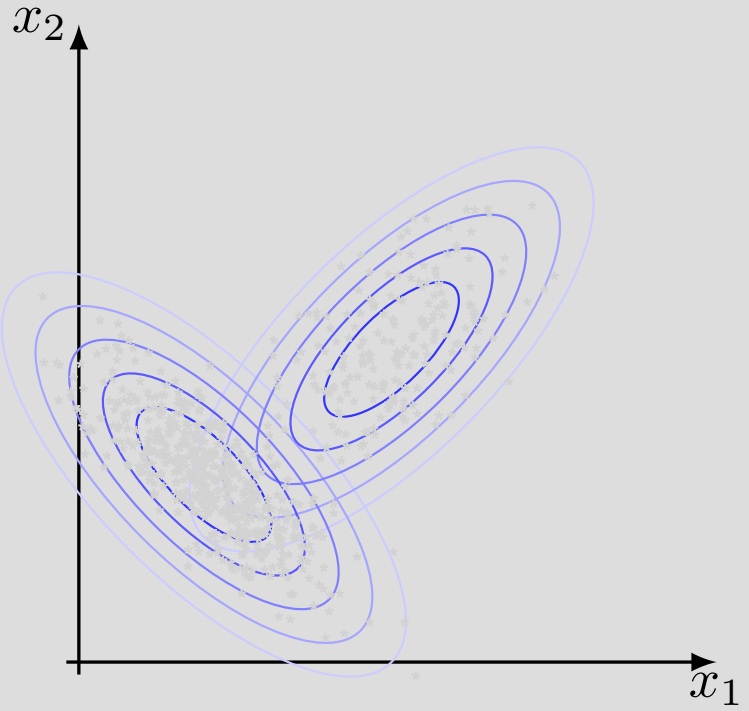


- Do this with feature vectors of one recording: Get a model for the recording
- Do this with feature vectors of all recordings from a category: Get a model for the category!

Gaussian Mixture Models

Data count reduction works as follows:

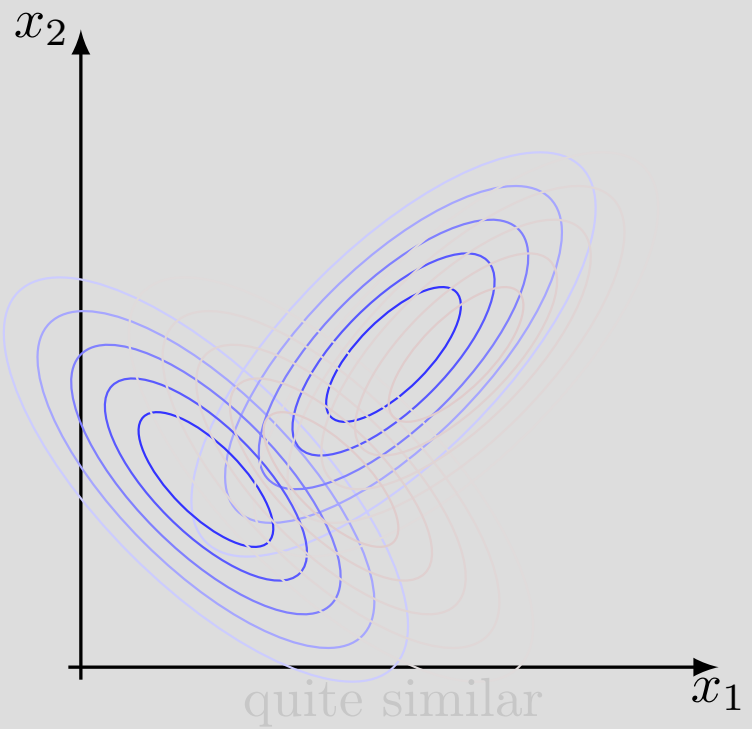
- Take all feature vectors of one feature
- Model their probability distribution
- Forget about the original feature vectors
- This step brings the data count reduction: One model (fixed model size) instead of many (arbitrary count) feature vectors



- Do this with feature vectors of one recording: Get a model for the recording
- Do this with feature vectors of all recordings from a category: Get a model for the category!

GMM: Dimensionality reduction

Dimensionality reduction works through comparision of the models:



quite similar

not that similar

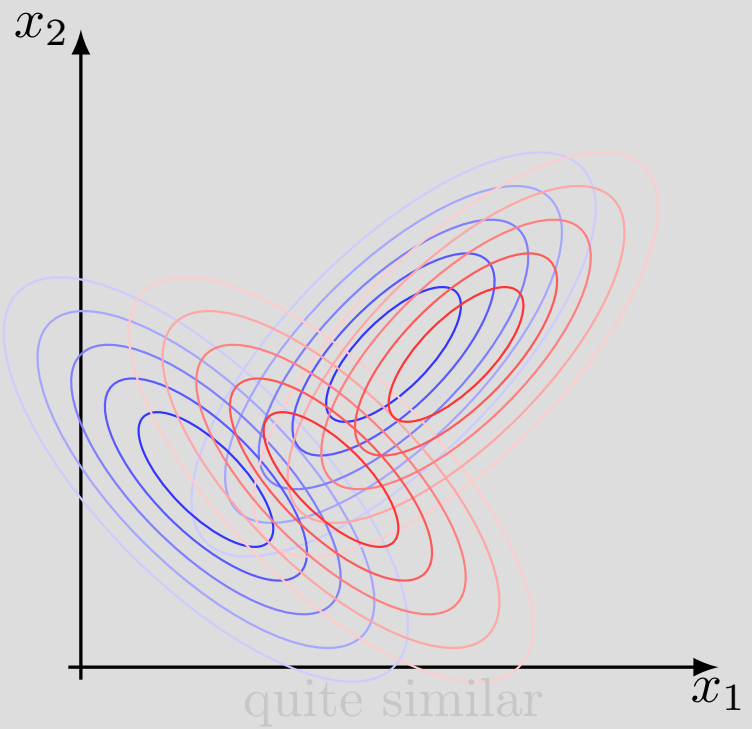
$d(a, b) \approx 0.9$

$d(a, b) \approx 30$

For comparision, the Kullback-Leibler divergence is used (Monte-Carlo integration!).

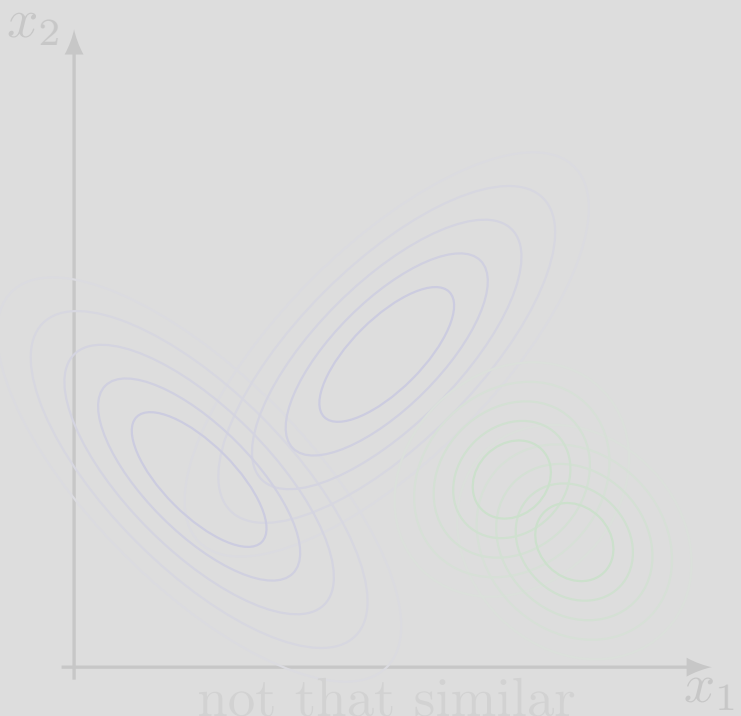
GMM: Dimensionality reduction

Dimensionality reduction works through comparision of the models:



quite similar

$d(a, b) \approx 0.9$



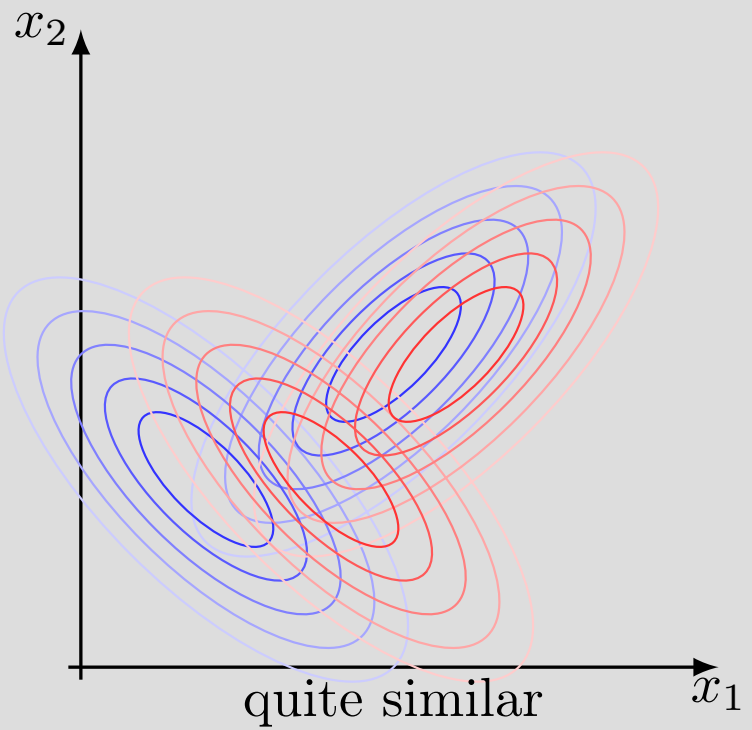
not that similar

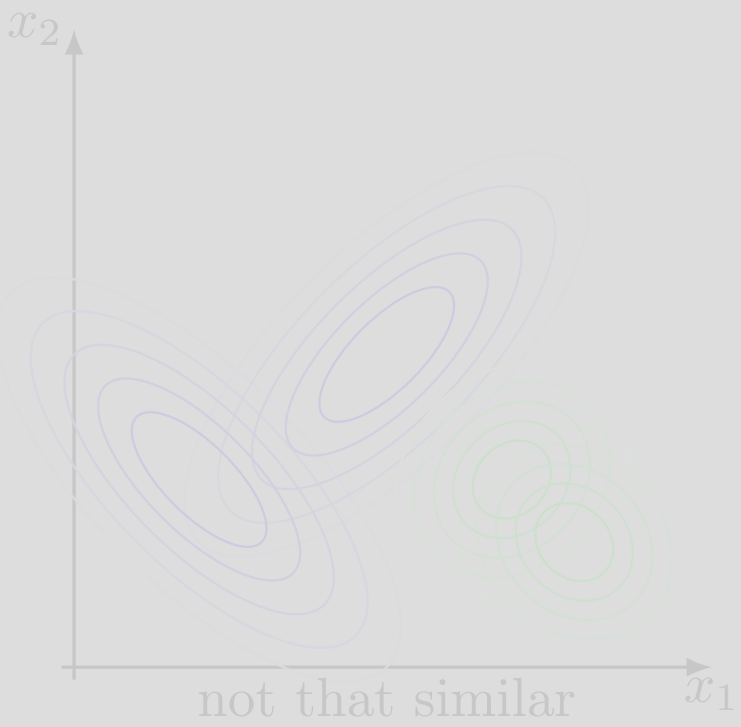
$d(a, b) \approx 30$

For comparision, the Kullback-Leibler divergence is used (Monte-Carlo integration!).

GMM: Dimensionality reduction

Dimensionality reduction works through comparision of the models:





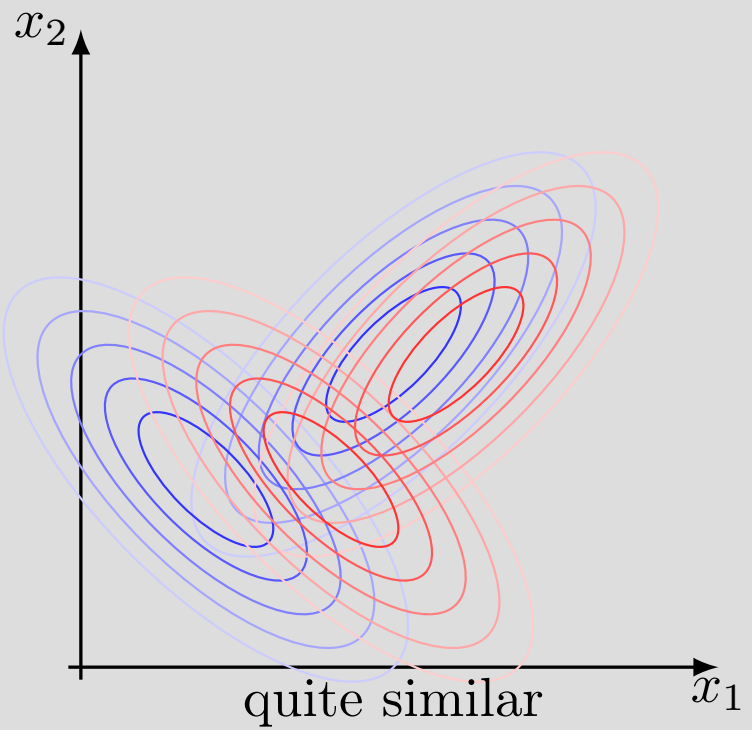
$d(a, b) \approx 0.9$

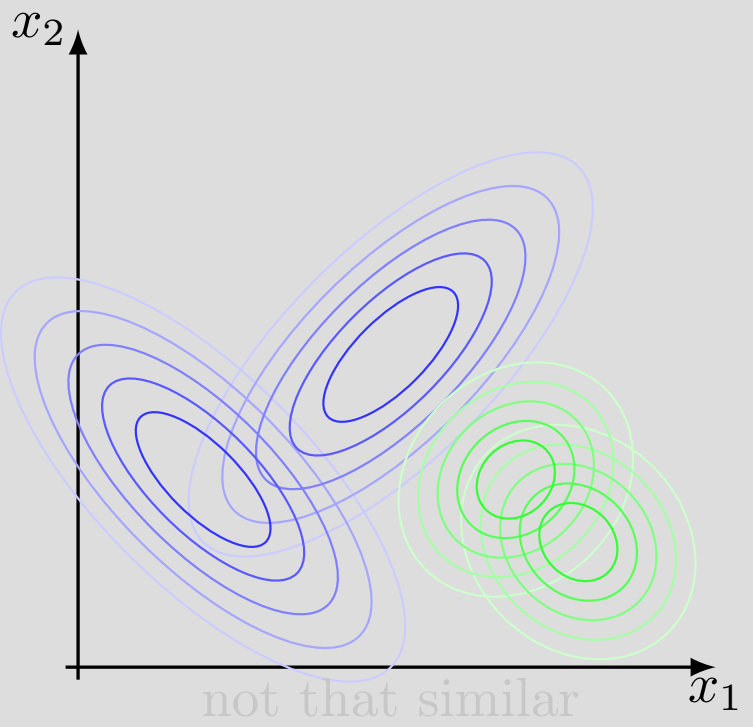
$d(a, b) \approx 30$

For comparision, the Kullback-Leibler divergence is used (Monte-Carlo integration!).

GMM: Dimensionality reduction

Dimensionality reduction works through comparision of the models:



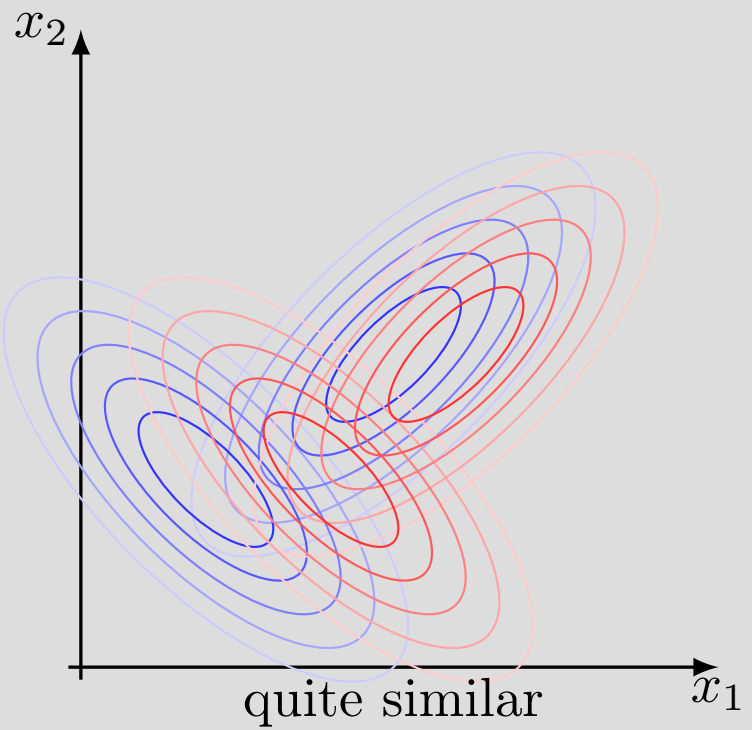


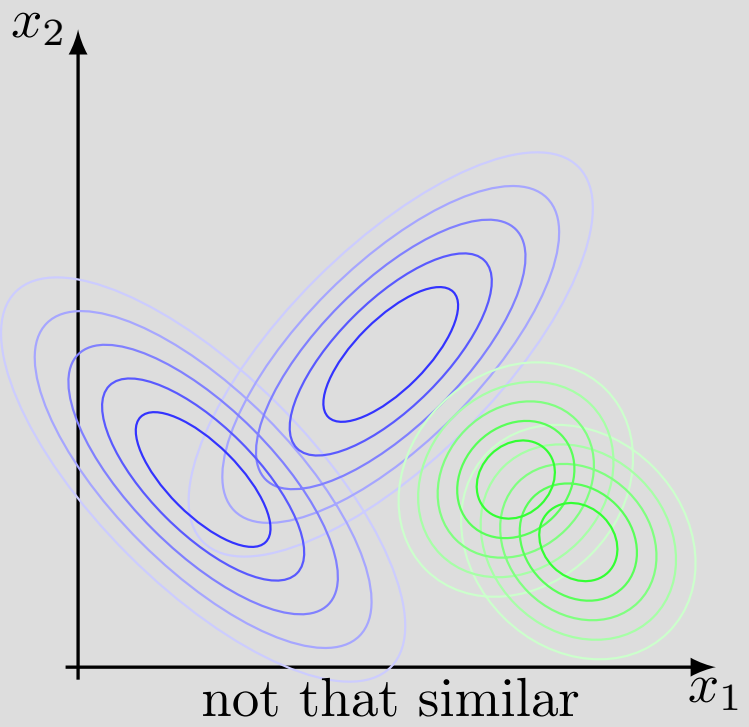
$d(a, b) \approx 0.9$ $d(a, b) \approx 30$

For comparision, the Kullback-Leibler divergence is used (Monte-Carlo integration!).

GMM: Dimensionality reduction

Dimensionality reduction works through comparision of the models:



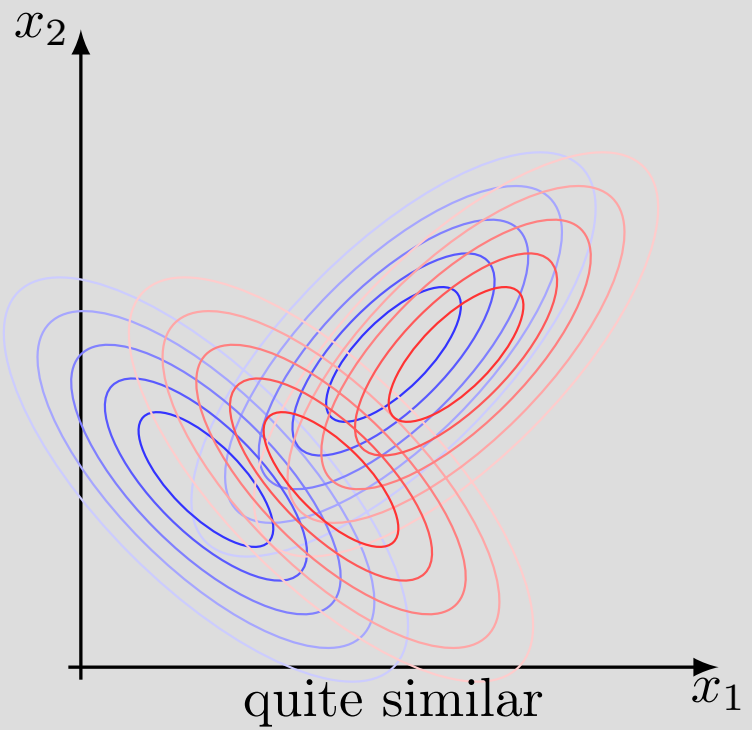


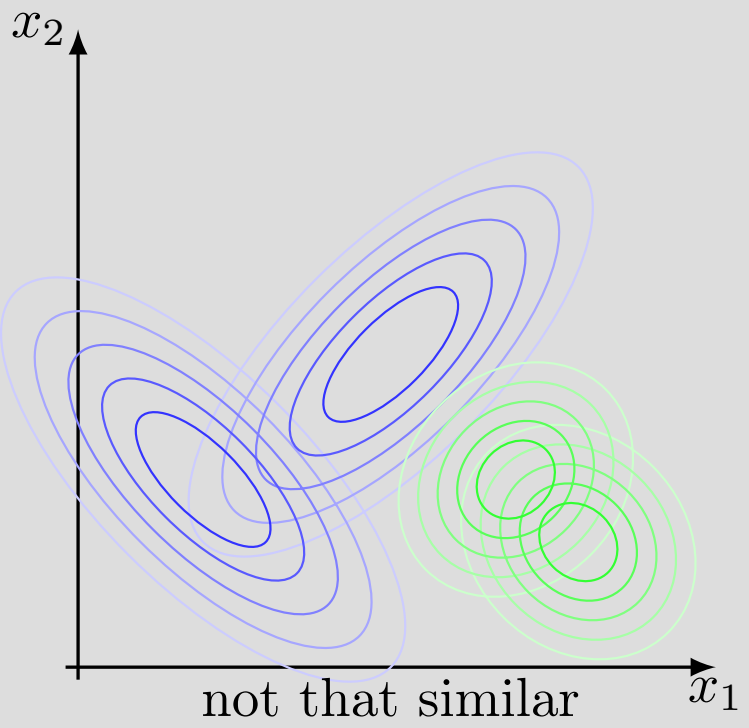
$d(a, b) \approx 0.9$ $d(a, b) \approx 30$

For comparision, the Kullback-Leibler divergence is used (Monte-Carlo integration!).

GMM: Dimensionality reduction

Dimensionality reduction works through comparision of the models:





$d(a, b) \approx 0.9$ $d(a, b) \approx 30$

For comparision, the Kullback-Leibler divergence is used (Monte-Carlo integration!).

GMM: Applied to recordings and categories

How does it work?

- Build a model for every recording
- Build a model for every category
- Compare recording-model to category-model
- Combine resulting scalar values for timbre and chroma with other scalar values to new "all-feature vector":

$$\text{feature vector} = \begin{pmatrix} \text{timbre similarity to category model} \\ \text{chroma similarity to category model} \\ \text{dynamic range} \\ \text{length of the recording} \end{pmatrix}$$

- There is one such feature vector per recording

GMM: Applied to recordings and categories

How does it work?

- Build a model for every recording
- Build a model for every category
- Compare recording-model to category-model
- Combine resulting scalar values for timbre and chroma with other scalar values to new "all-feature vector":

$$\text{feature vector} = \begin{pmatrix} \text{timbre similarity to category model} \\ \text{chroma similarity to category model} \\ \text{dynamic range} \\ \text{length of the recording} \end{pmatrix}$$

- There is one such feature vector per recording

GMM: Applied to recordings and categories

How does it work?

- Build a model for every recording
- Build a model for every category
- Compare recording-model to category-model
- Combine resulting scalar values for timbre and chroma with other scalar values to new "all-feature vector":

$$\text{feature vector} = \begin{pmatrix} \text{timbre similarity to category model} \\ \text{chroma similarity to category model} \\ \text{dynamic range} \\ \text{length of the recording} \end{pmatrix}$$

- There is one such feature vector per recording

GMM: Applied to recordings and categories

How does it work?

- Build a model for every recording
- Build a model for every category
- Compare recording-model to category-model
- Combine resulting scalar values for timbre and chroma with other scalar values to new "all-feature vector":

$$\text{feature vector} = \begin{pmatrix} \text{timbre similarity to category model} \\ \text{chroma similarity to category model} \\ \text{dynamic range} \\ \text{length of the recording} \end{pmatrix}$$

- There is one such feature vector per recording

GMM: Applied to recordings and categories

How does it work?

- Build a model for every recording
- Build a model for every category
- Compare recording-model to category-model
- Combine resulting scalar values for timbre and chroma with other scalar values to new "all-feature vector":

$$\text{feature vector} = \begin{pmatrix} \text{timbre similarity to category model} \\ \text{chroma similarity to category model} \\ \text{dynamic range} \\ \text{length of the recording} \end{pmatrix}$$

- There is one such feature vector per recording

GMM: Applied to recordings and categories

How does it work?

- Build a model for every recording
- Build a model for every category
- Compare recording-model to category-model
- Combine resulting scalar values for timbre and chroma with other scalar values to new "all-feature vector":

$$\text{feature vector} = \begin{pmatrix} \text{timbre similarity to category model} \\ \text{chroma similarity to category model} \\ \text{dynamic range} \\ \text{length of the recording} \end{pmatrix}$$

- There is one such feature vector per recording

Plan

Classification: Classical approaches

- Classical approaches use categories
- Decision: Does a recording belong to a category, or not?
- Score is binary: e.g. -1 or 1
- Positive and negative examples needed for training (ideally many)
- Approaches exist that only need positive examples
- Examples for binary classifiers: LDA, SVM, (ANN)

Classification: Classical approaches

- Classical approaches use categories
- Decision: Does a recording belong to a category, or not?
- Score is binary: e.g. -1 or 1
- Positive and negative examples needed for training (ideally many)
- Approaches exist that only need positive examples
- Examples for binary classifiers: LDA, SVM, (ANN)

Classification: Classical approaches

- Classical approaches use categories
- Decision: Does a recording belong to a category, or not?
- Score is binary: e.g. -1 or 1
- Positive and negative examples needed for training (ideally many)
- Approaches exist that only need positive examples
- Examples for binary classifiers: LDA, SVM, (ANN)

Classification: Classical approaches

- Classical approaches use categories
- Decision: Does a recording belong to a category, or not?
- Score is binary: e.g. -1 or 1
- Positive and negative examples needed for training (ideally many)
- Approaches exist that only need positive examples
- Examples for binary classifiers: LDA, SVM, (ANN)

Classification: Classical approaches

- Classical approaches use categories
- Decision: Does a recording belong to a category, or not?
- Score is binary: e.g. -1 or 1
- Positive and negative examples needed for training (ideally many)
- Approaches exist that only need positive examples
- Examples for binary classifiers: LDA, SVM, (ANN)

Classification: Classical approaches

- Classical approaches use categories
- Decision: Does a recording belong to a category, or not?
- Score is binary: e.g. -1 or 1
- Positive and negative examples needed for training (ideally many)
- Approaches exist that only need positive examples
- Examples for binary classifiers: LDA, SVM, (ANN)

Classification: Approach used

- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)

\+ Better matches are shown first

\+ No need for both positive and negative examples

\+ Flexible approach, fits to users needs

– There is no decision which recordings definitely do not match

Classification: Approach used

- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)

\+ Better matches are shown first

\+ No need for both positive and negative examples

\+ Flexible approach, fits to users needs

– There is no decision which recordings definitely do not match

Classification: Approach used

- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)

\+ Better matches are shown first

\+ No need for both positive and negative examples

\+ Flexible approach, fits to users needs

– There is no decision which recordings definitely do not match

Classification: Approach used

- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)

\+ Better matches are shown first

\+ No need for both positive and negative examples

\+ Flexible approach, fits to users needs

\– There is no decision which recordings definitely do not match

Classification: Approach used

- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)

⊕ Better matches are shown first

\+ No need for both positive and negative examples

\+ Flexible approach, fits to users needs

– There is no decision which recordings definitely do not match

Classification: Approach used

- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)

⊕ Better matches are shown first

⊕ No need for both positive and negative examples

\+ Flexible approach, fits to users needs

− There is no decision which recordings definitely do not match

Classification: Approach used

- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)

⊕ Better matches are shown first

⊕ No need for both positive and negative examples

⊕ Flexible approach, fits to users needs

– There is no decision which recordings definitely do not match

Classification: Approach used

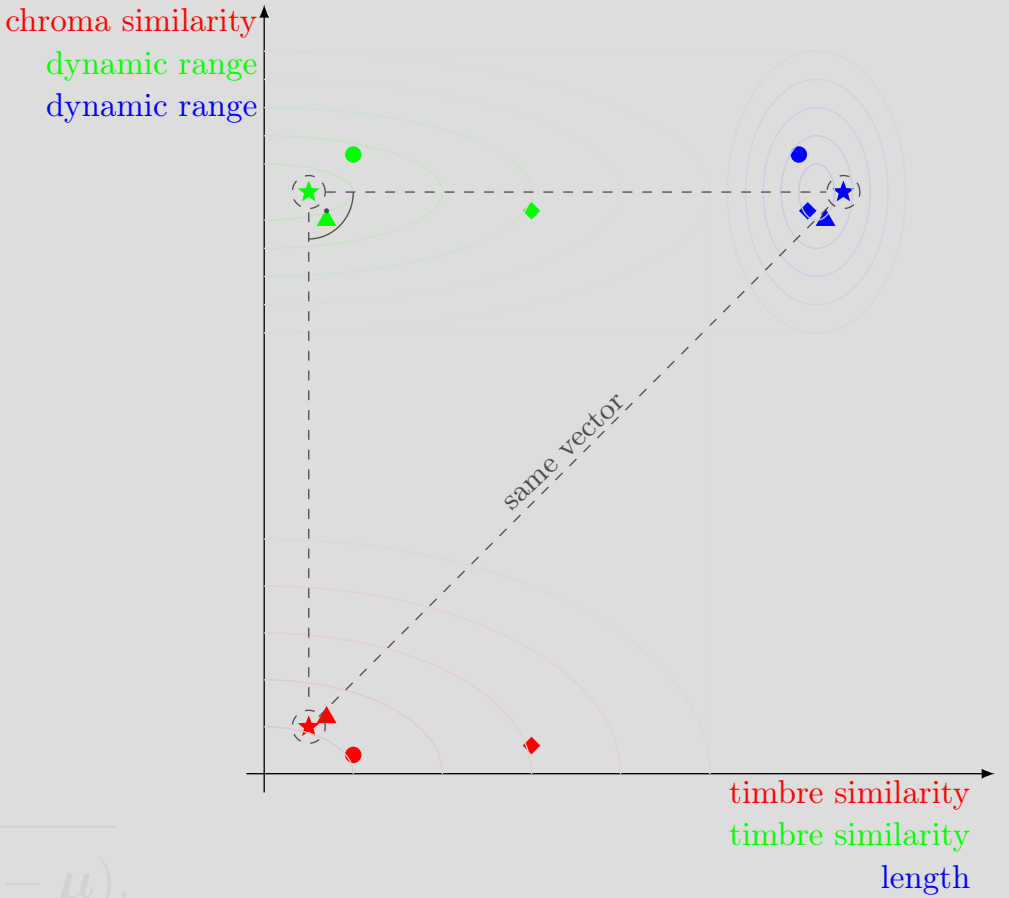
- Different approach here: Recordings get a score from $[-1, 1]$ for a category
- Gives a ranking rather than a classification
- Positive and negative examples can be used, but there is no need for both
- Only a few examples are needed (works from a single feature vector, 5-10 is ideal)
- ⊕ Better matches are shown first
- ⊕ No need for both positive and negative examples
- ⊕ Flexible approach, fits to users needs
- ⊖ There is no decision which recordings definitely do not match

Classification: How does it work? (1/2)

- Four-dimensional recording feature vectors used
- Calculate distribution of vectors ($\rightarrow$ covariance matrix)
- Gaussian Model (no mixture!) of positive example feature vectors
- Calculate Mahalanobis distance of any other feature vector:

$$d_{\Sigma}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}.$$

- This gives a distance value in $[0, \infty[$.



Sectional drawing of feature vectors

GMM: Applied to recordings and categories

How does it work?

- Build a model for every recording
- Build a model for every category
- Compare recording-model to category-model
- Combine resulting scalar values for timbre and chroma with other scalar values to new "all-feature vector":

$$\text{feature vector} = \begin{pmatrix} \text{timbre similarity to category model} \\ \text{chroma similarity to category model} \\ \text{dynamic range} \\ \text{length of the recording} \end{pmatrix}$$

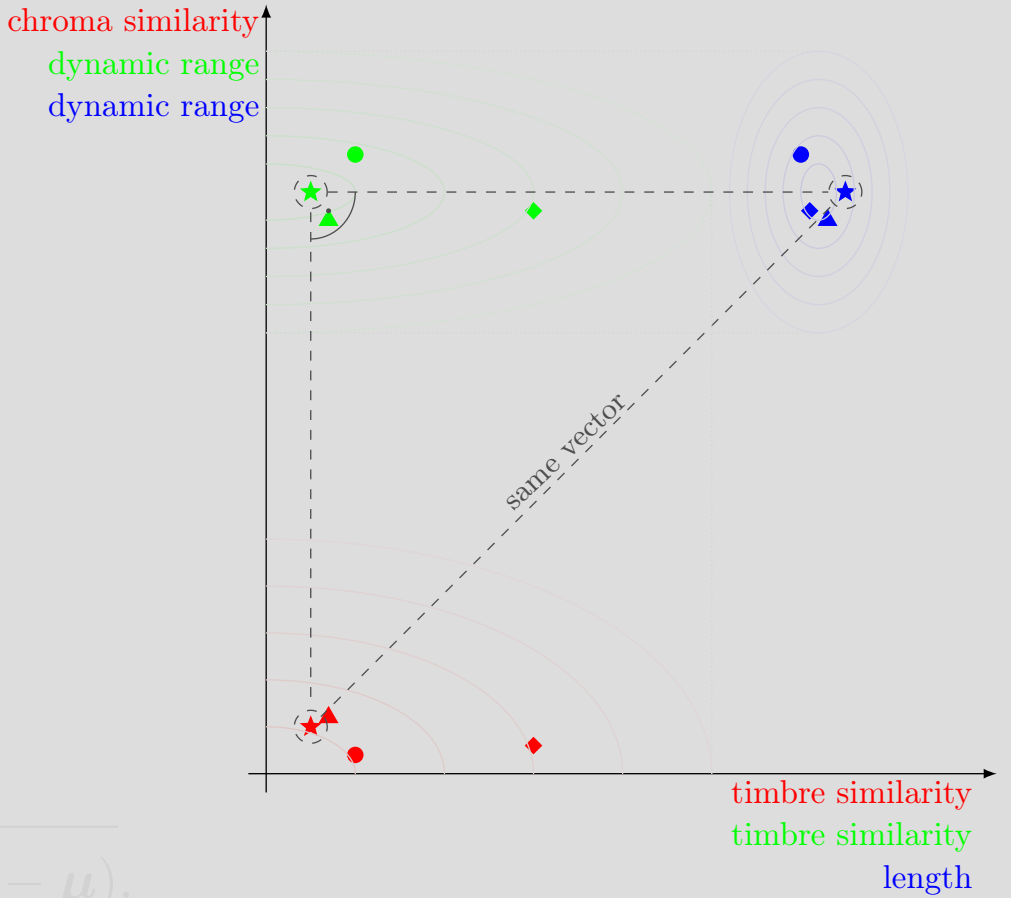
- There is one such feature vector per recording

Classification: How does it work? (1/2)

- Four-dimensional recording feature vectors used
- Calculate distribution of vectors ($\rightarrow$ covariance matrix)
- Gaussian Model (no mixture!) of positive example feature vectors
- Calculate Mahalanobis distance of any other feature vector:

$$d_{\Sigma}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}.$$

- This gives a distance value in $[0, \infty[$.



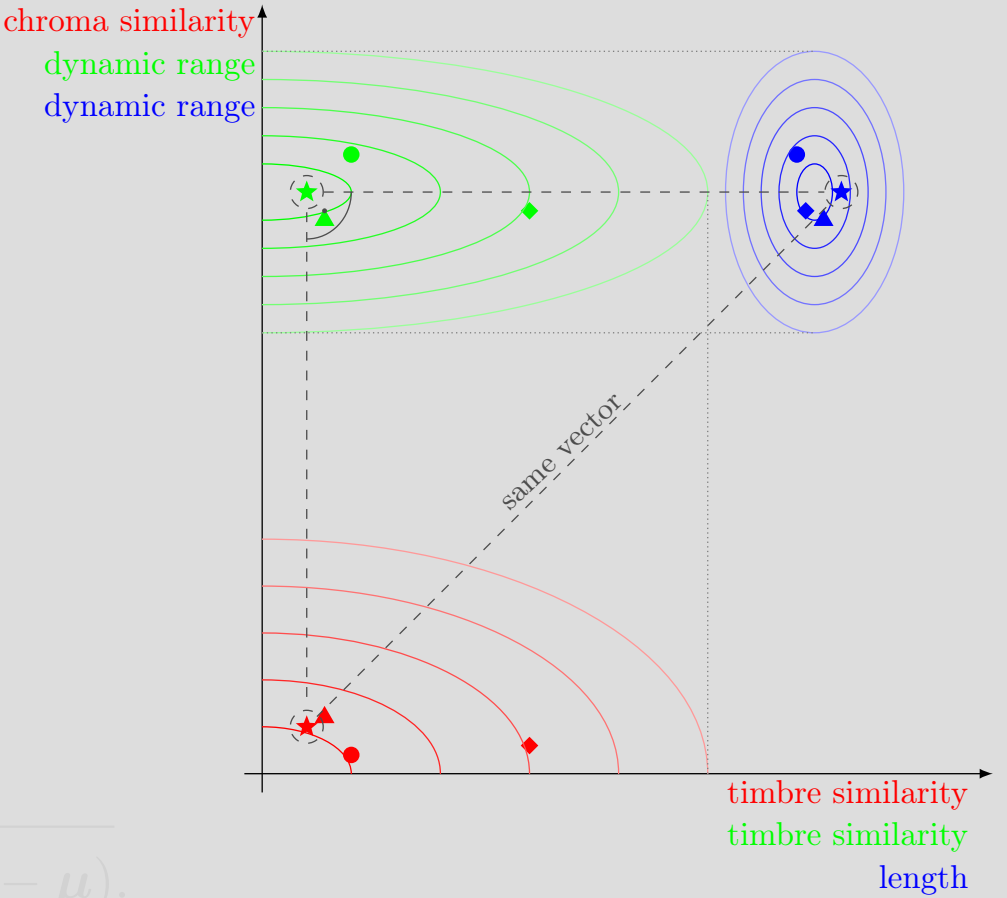
Sectional drawing of feature vectors

Classification: How does it work? (1/2)

- Four-dimensional recording feature vectors used
- Calculate distribution of vectors ($\rightarrow$ covariance matrix)
- Gaussian Model (no mixture!) of positive example feature vectors
- Calculate Mahalanobis distance of any other feature vector:

$$d_{\Sigma}(x, y) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}.$$

- This gives a distance value in $[0, \infty[$.



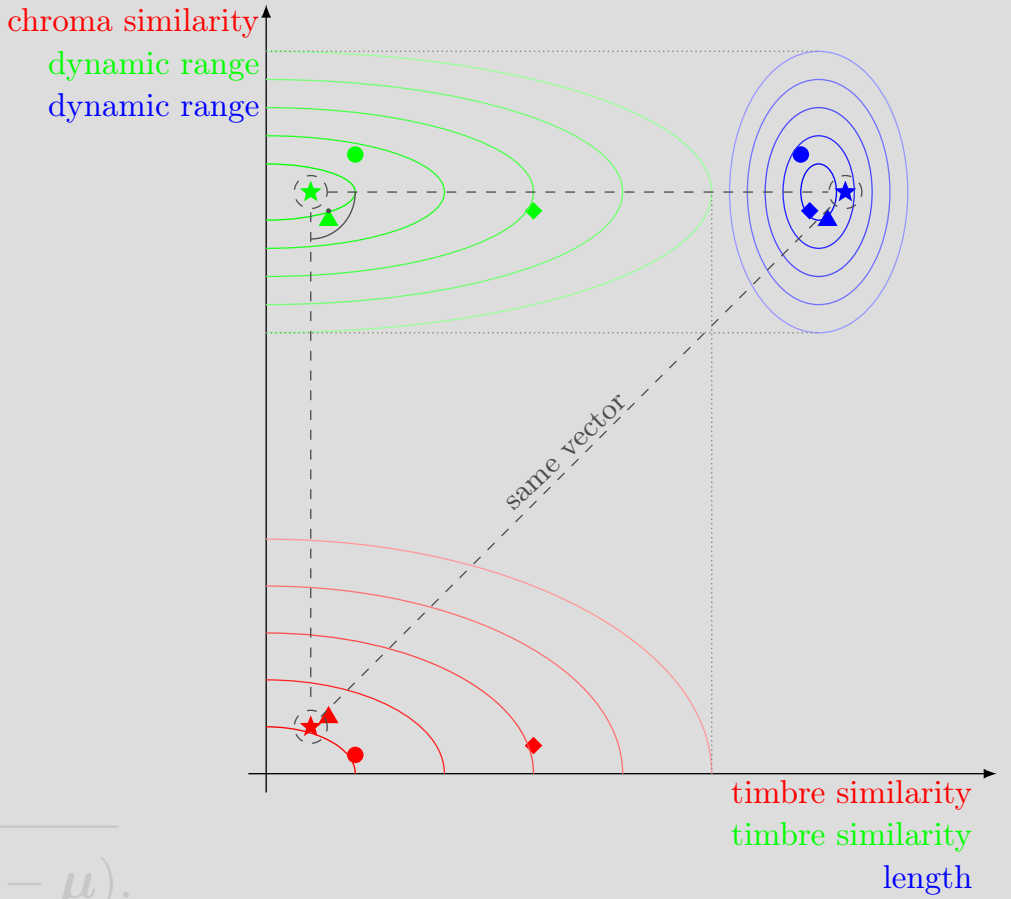
Sectional drawing of feature vectors

Classification: How does it work? (1/2)

- Four-dimensional recording feature vectors used
- Calculate distribution of vectors ($\rightarrow$ covariance matrix)
- Gaussian Model (no mixture!) of positive example feature vectors
- Calculate Mahalanobis distance of any other feature vector:

$$d_{\Sigma}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}.$$

- This gives a distance value in $[0, \infty[$.



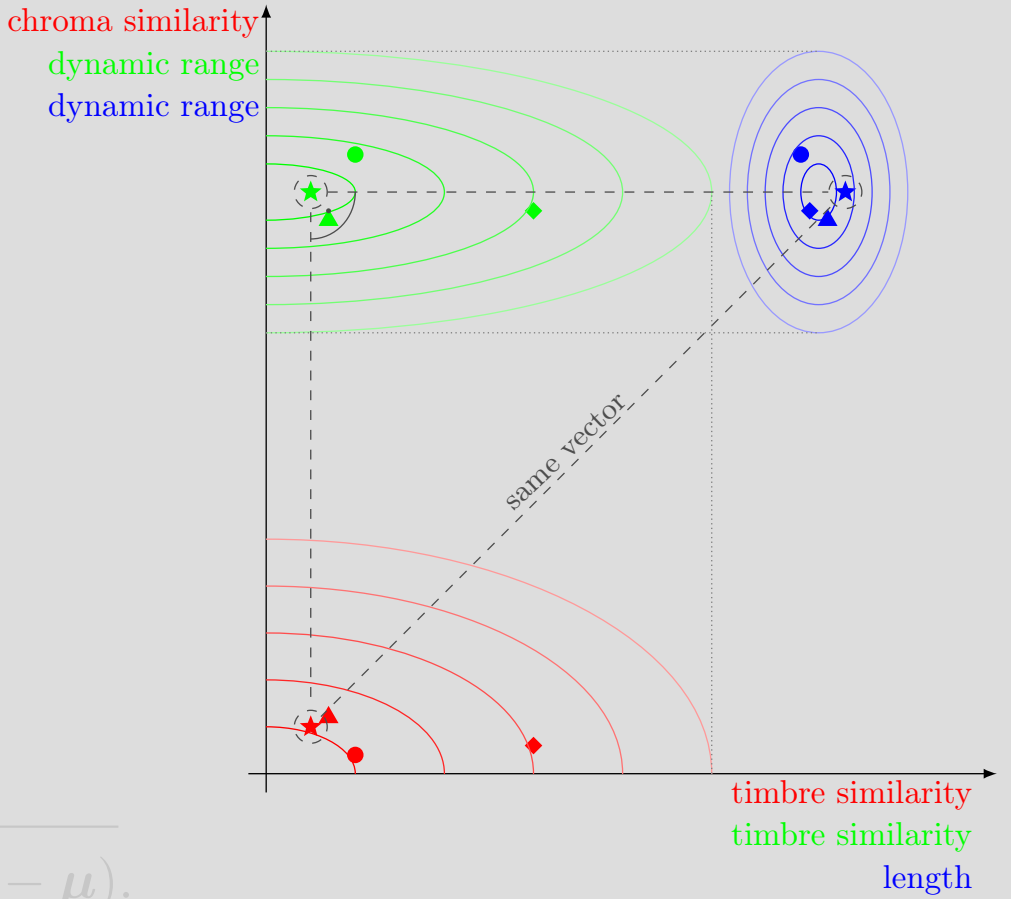
Sectional drawing of feature vectors

Classification: How does it work? (1/2)

- Four-dimensional recording feature vectors used
- Calculate distribution of vectors ($\rightarrow$ covariance matrix)
- Gaussian Model (no mixture!) of positive example feature vectors
- Calculate Mahalanobis distance of any other feature vector:

$$d_{\Sigma}(x, y) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}.$$

- This gives a distance value in $[0, \infty[$.

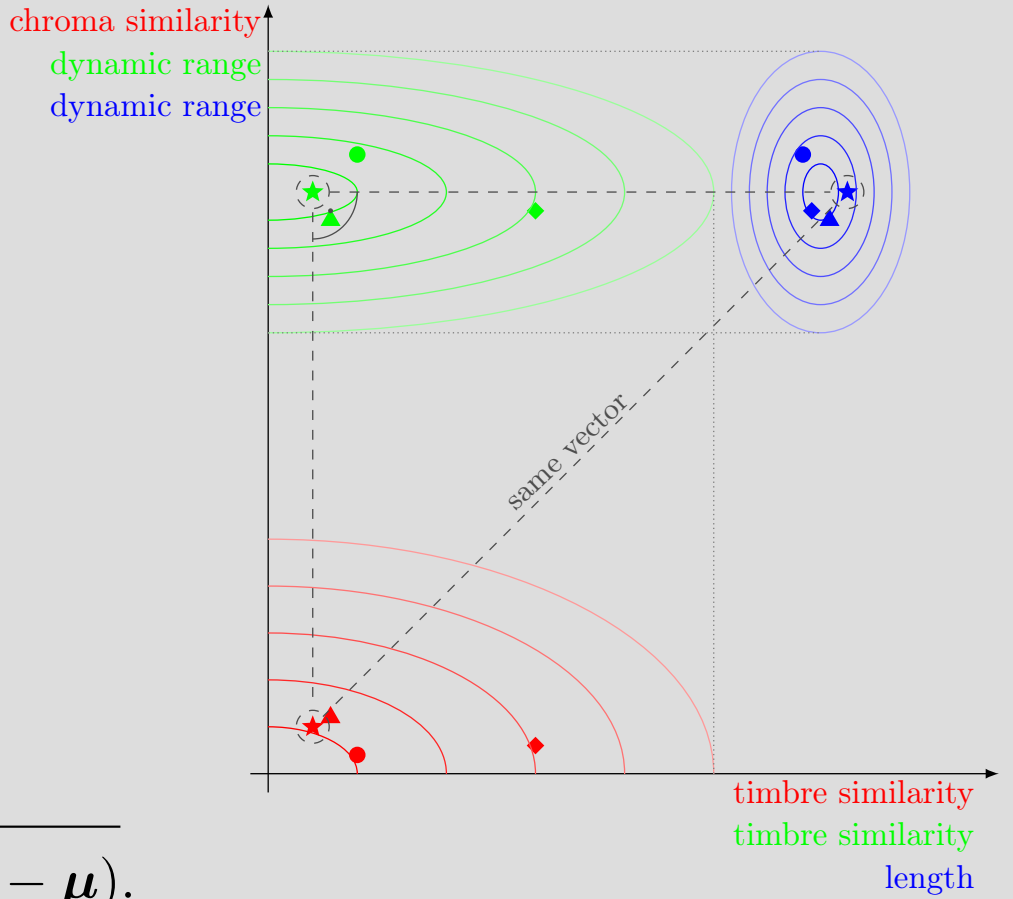


Sectional drawing of feature vectors

Classification: How does it work? (1/2)

- Four-dimensional recording feature vectors used
- Calculate distribution of vectors ($\rightarrow$ covariance matrix)
- Gaussian Model (no mixture!) of positive example feature vectors
- Calculate Mahalanobis distance of any other feature vector:

$$d_{\boldsymbol{\Sigma}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}.$$



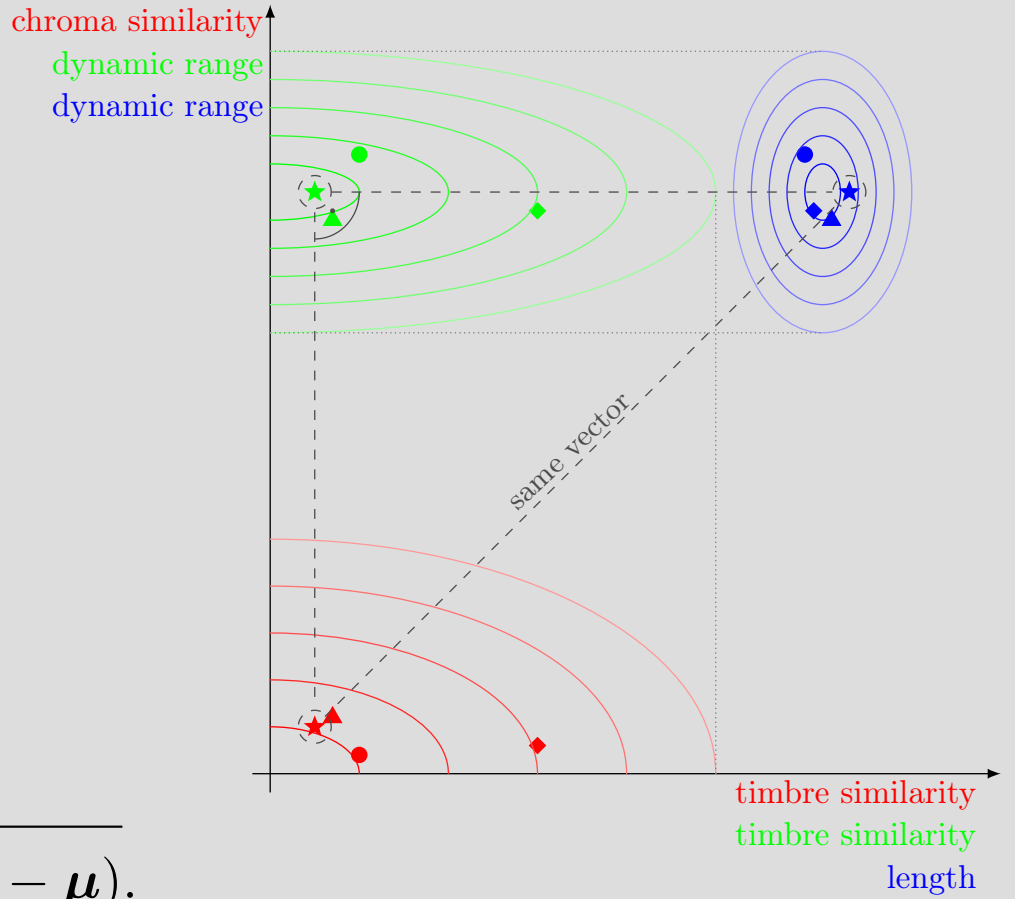
Sectional drawing of feature vectors

- This gives a distance value in $[0, \infty[$.

Classification: How does it work? (1/2)

- Four-dimensional recording feature vectors used
- Calculate distribution of vectors ($\rightarrow$ covariance matrix)
- Gaussian Model (no mixture!) of positive example feature vectors
- Calculate Mahalanobis distance of any other feature vector:

$$d_{\boldsymbol{\Sigma}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}.$$

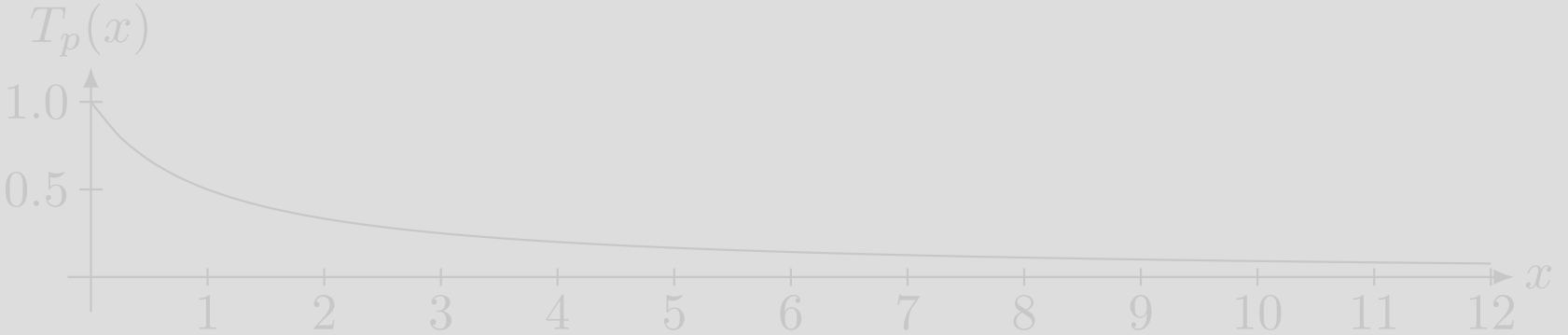


Sectional drawing of feature vectors

- This gives a distance value in $[0, \infty[$.

Classification: How does it work? (2/2)

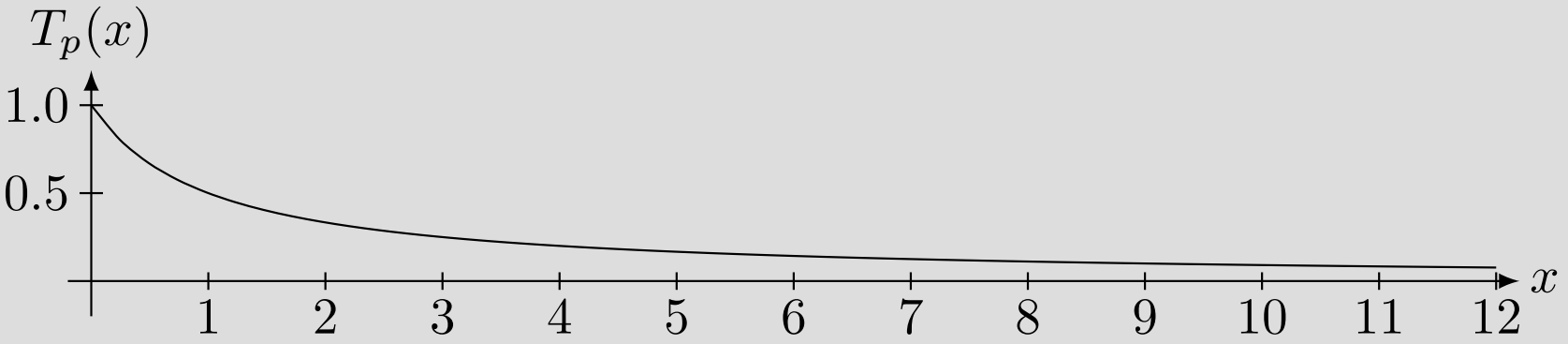
- Transform from $[0, \infty[$ to $[0, 1]$ through $T_p(x) = \frac{1}{1+x}$



- Up to now: Positive model
- Negative model: Second model, mapped to $[-1, 0]$ via $T_n(x) = \frac{-1}{1+x}$
- Sum both intervals: Values from $[-1, 1]$

Classification: How does it work? (2/2)

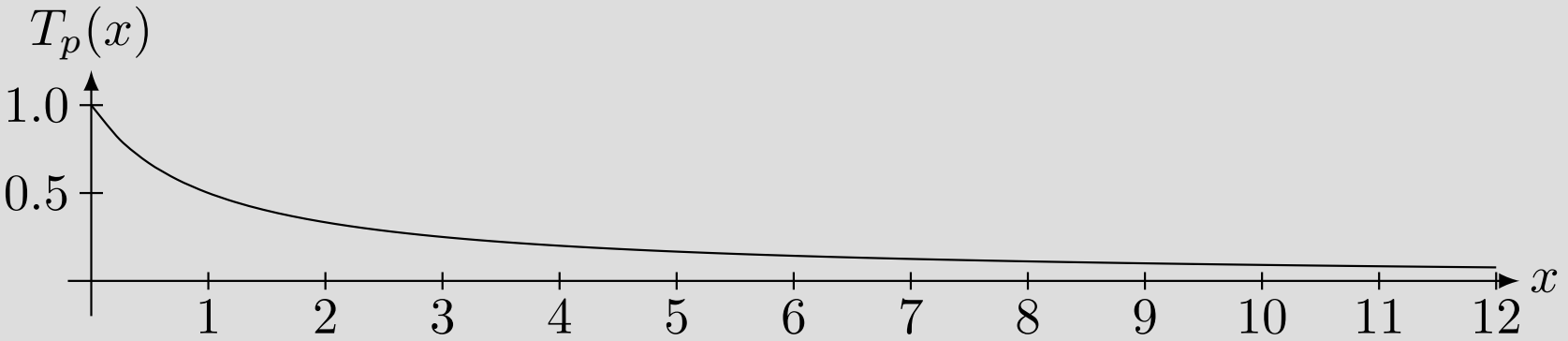
- Transform from $[0, \infty[$ to $[0, 1]$ through $T_p(x) = \frac{1}{1+x}$



- Up to now: Positive model
- Negative model: Second model, mapped to $[-1, 0]$ via $T_n(x) = \frac{-1}{1+x}$
- Sum both intervals: Values from $[-1, 1]$

Classification: How does it work? (2/2)

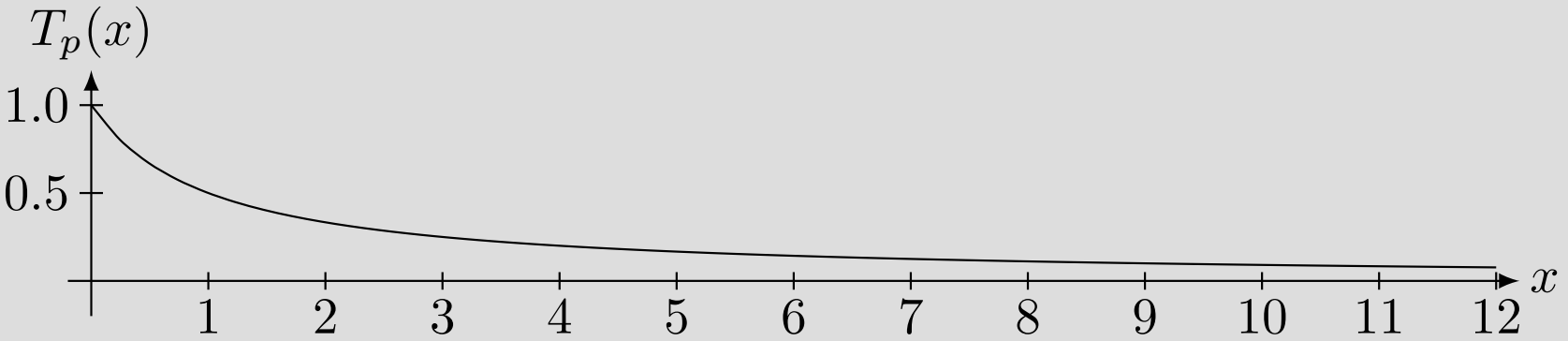
- Transform from $[0, \infty[$ to $[0, 1]$ through $T_p(x) = \frac{1}{1+x}$



- Up to now: Positive model
- Negative model: Second model, mapped to $[-1, 0]$ via $T_n(x) = \frac{-1}{1+x}$
- Sum both intervals: Values from $[-1, 1]$

Classification: How does it work? (2/2)

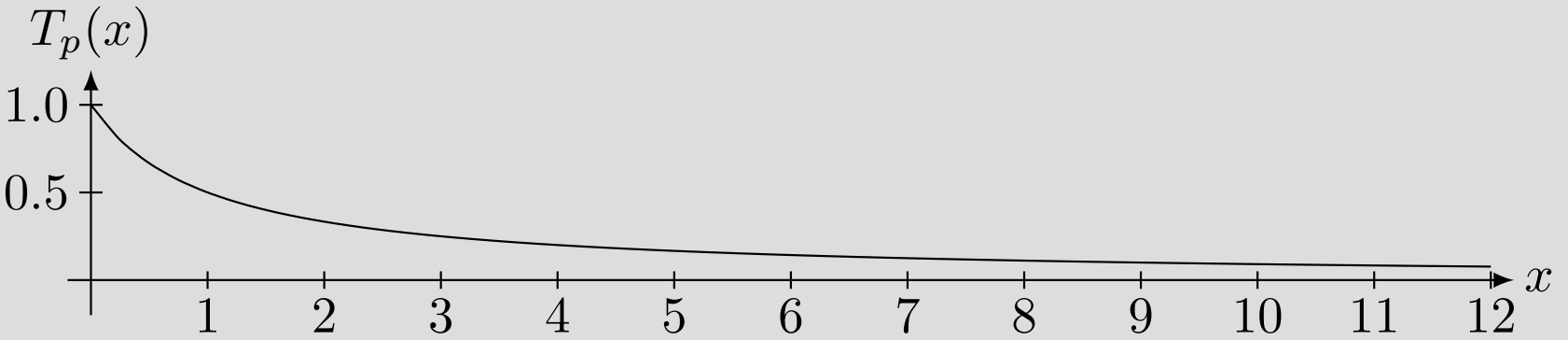
- Transform from $[0, \infty[$ to $[0, 1]$ through $T_p(x) = \frac{1}{1+x}$



- Up to now: Positive model
- Negative model: Second model, mapped to $[-1, 0]$ via $T_n(x) = \frac{-1}{1+x}$
- Sum both intervals: Values from $[-1, 1]$

Classification: How does it work? (2/2)

- Transform from $[0, \infty[$ to $[0, 1]$ through $T_p(x) = \frac{1}{1+x}$



- Up to now: Positive model
- Negative model: Second model, mapped to $[-1, 0]$ via $T_n(x) = \frac{-1}{1+x}$
- Sum both intervals: Values from $[-1, 1]$

Plan

Results

Testing procedure: Train classifier with positive and negative examples, take 100 best matches, count same-category matches.

- `Classical`: Three positives, three negatives → 94% matches, first "false-positive" at rank 57
- `Jazz/RnB`: Two positives, two negatives → 89% matches, first "false-positive" at rank 41
- `Pop/Rock`: Two positives, one negative → 87% matches, first "false-positive" at rank 13

Results

Testing procedure: Train classifier with positive and negative examples, take 100 best matches, count same-category matches.

- `Classical`: Three positives, three negatives → 94% matches, first "false-positive" at rank 57
- `Jazz/RnB`: Two positives, two negatives → 89% matches, first "false-positive" at rank 41
- `Pop/Rock`: Two positives, one negative → 87% matches, first "false-positive" at rank 13

Results

Testing procedure: Train classifier with positive and negative examples, take 100 best matches, count same-category matches.

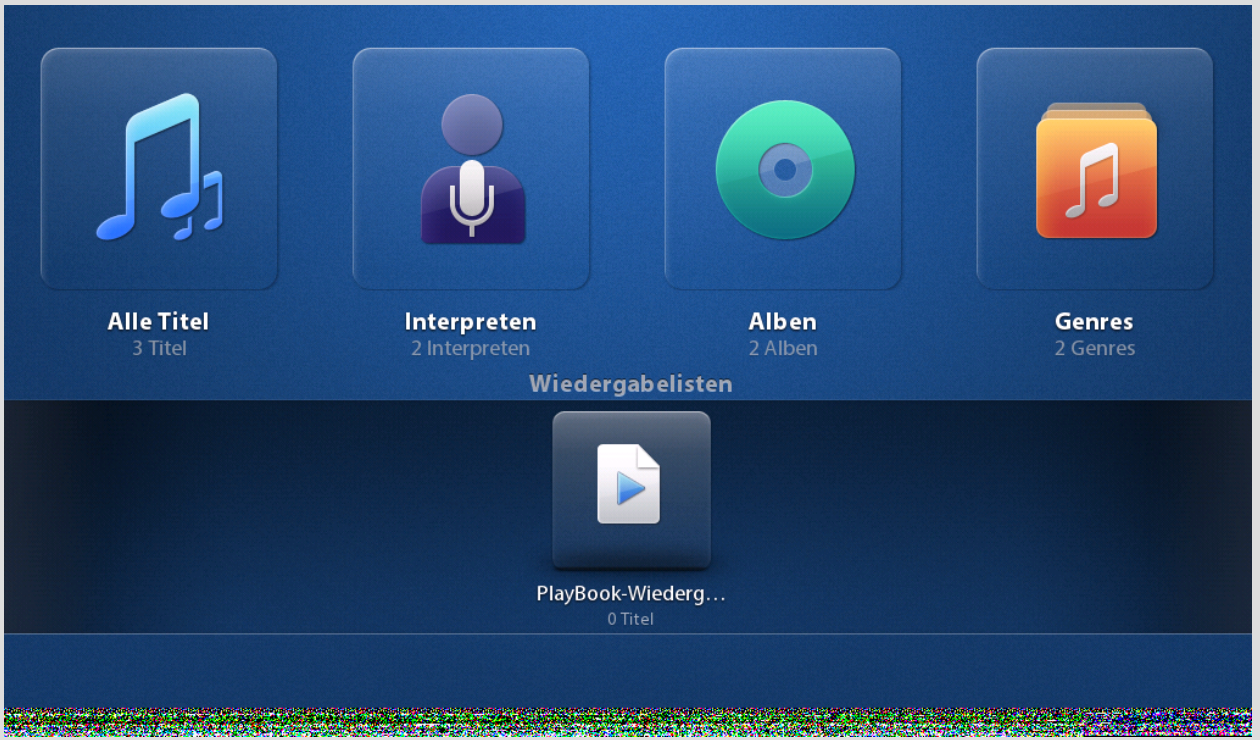
- `Classical`: Three positives, three negatives → 94% matches, first "false-positive" at rank 57
- `Jazz/RnB`: Two positives, two negatives → 89% matches, first "false-positive" at rank 41
- `Pop/Rock`: Two positives, one negative → 87% matches, first "false-positive" at rank 13

Results

Testing procedure: Train classifier with positive and negative examples, take 100 best matches, count same-category matches.

- `Classical`: Three positives, three negatives → 94% matches, first "false-positive" at rank 57
- `Jazz/RnB`: Two positives, two negatives → 89% matches, first "false-positive" at rank 41
- `Pop/Rock`: Two positives, one negative → 87% matches, first "false-positive" at rank 13

Demonstration



Questions?

Any questions left?

▸ Dynamic range ▸ Tempo ▸ Timbre ▸ Chroma

Plan

Dynamic range

- Intuition: We want to define a measure of how loud parts of a musical piece relate to the quieter ones.
- The measure should be small if most of the signal is at one volume. It should increase with the amount of volume changes during the recording.

Within the context of music comparision, we define the dynamic range of an audio signal as the root of the mean energy of the continous input signal $x_\mathrm{c}(t)$, which is

$$\mathrm{dyn}_\mathrm{cRMS} = \sqrt{\frac{1}{T_\mathrm{c}} \int_0^{T_\mathrm{c}} x_\mathrm{c}^2(t)\,\mathrm{d}\,t} \tag{4}$$

with T_c being the last point in time of the signal.

Dynamic range

- Intuition: We want to define a measure of how loud parts of a musical piece relate to the quieter ones.
- The measure should be small if most of the signal is at one volume. It should increase with the amount of volume changes during the recording.

Within the context of music comparision, we define the dynamic range of an audio signal as the root of the mean energy of the continous input signal $x_\mathrm{c}(t)$, which is

$$\mathrm{dyn}_\mathrm{cRMS} = \sqrt{\frac{1}{T_\mathrm{c}} \int_0^{T_\mathrm{c}} x_\mathrm{c}^2(t)\,\mathrm{d}\,t} \tag{4}$$

with T_c being the last point in time of the signal.

Dynamic range

- Intuition: We want to define a measure of how loud parts of a musical piece relate to the quieter ones.
- The measure should be small if most of the signal is at one volume. It should increase with the amount of volume changes during the recording.

Within the context of music comparision, we define the dynamic range of an audio signal as the root of the mean energy of the continous input signal $x_\mathrm{c}(t)$, which is

$$\mathrm{dyn}_\mathrm{cRMS} = \sqrt{\frac{1}{T_\mathrm{c}} \int_0^{T_\mathrm{c}} x_\mathrm{c}^2(t)\,\mathrm{d}\,t} \tag{4}$$

with T_c being the last point in time of the signal.

Dynamic range

This definition will be changed slightly for the implementation:

$$\mathrm{dyn}_{\mathrm{dRMS}} = 1 - \sqrt{\frac{1}{N}\sum_{n=0}^{N} \mathrm{nsum}_{\mathrm{CQ}}{}^{2}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)}. \tag{5}$$

with

$$\mathrm{nsum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \frac{1}{R}\sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{6}$$

and

$$R = \max_{t_n}\left(\sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)|\right). \tag{7}$$

Remark

Here, we are talking of discrete points in time. Every t_n refers to the continous time interval $[t_n, t_{n+1}]$.

Dynamic range

This definition will be changed slightly for the implementation:

$$\mathrm{dyn}_{\mathrm{dRMS}} = 1 - \sqrt{\frac{1}{N}\sum_{n=0}^{N} \mathrm{nsum}_{\mathrm{CQ}}{}^{2}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)}. \tag{5}$$

with

$$\mathrm{nsum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \frac{1}{R}\sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{6}$$

and

$$R = \max_{t_n}\left(\sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)|\right). \tag{7}$$

Remark

Here, we are talking of discrete points in time. Every t_n refers to the continous time interval $[t_n, t_{n+1}]$.

Dynamic range

This definition will be changed slightly for the implementation:

$$\mathrm{dyn}_{\mathrm{dRMS}} = 1 - \sqrt{\frac{1}{N}\sum_{n=0}^{N} \mathrm{nsum}_{\mathrm{CQ}}{}^{2}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)}. \tag{5}$$

with

$$\mathrm{nsum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \frac{1}{R}\sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{6}$$

and

$$R = \max_{t_n}\left(\sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)|\right). \tag{7}$$

Remark

Here, we are talking of discrete points in time. Every t_n refers to the continous time interval $[t_n, t_{n+1}]$.

Tempo in bpm (beats per minute)

- Intuitive: The speed at which humans tap when listening to a song
- Problem: That speed is not well-defined. Some persons tap at quarters, some at halves, ...

1. Take the sum of the constant-Q bins $\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
2. Calculate the difference vector $\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
3. Calculate the autocorrelation of the difference vector
4. Find recurring peaks in the autocorrelation function

$$\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{8}$$

$$\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \mathrm{sum}_{\mathrm{CQ}}(t_n) - \mathrm{sum}_{\mathrm{CQ}}(t_{n+1}) \tag{9}$$

$$\mathrm{a}_{\mathrm{CQ}}(\mathrm{d}_{\mathrm{CQ}}(t_n), \tau) = \sum_{\tau=0}^{\tau_{\max}} \mathrm{d}_{\mathrm{CQ}}(t_n) * \mathrm{d}_{\mathrm{CQ}}(t_n - \tau) \tag{10}$$

Tempo in bpm (beats per minute)

- Intuitive: The speed at which humans tap when listening to a song
- Problem: That speed is not well-defined. Some persons tap at quarters, some at halves, ...

1. Take the sum of the constant-Q bins $\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
2. Calculate the difference vector $\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
3. Calculate the autocorrelation of the difference vector
4. Find recurring peaks in the autocorrelation function

$$\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{8}$$

$$\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \mathrm{sum}_{\mathrm{CQ}}(t_n) - \mathrm{sum}_{\mathrm{CQ}}(t_{n+1}) \tag{9}$$

$$\mathrm{a}_{\mathrm{CQ}}(\mathrm{d}_{\mathrm{CQ}}(t_n), \tau) = \sum_{\tau=0}^{\tau_{\max}} \mathrm{d}_{\mathrm{CQ}}(t_n) * \mathrm{d}_{\mathrm{CQ}}(t_n - \tau) \tag{10}$$

Tempo in bpm (beats per minute)

- Intuitive: The speed at which humans tap when listening to a song
- Problem: That speed is not well-defined. Some persons tap at quarters, some at halves, ...

1. Take the sum of the constant-Q bins $\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
2. Calculate the difference vector $\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
3. Calculate the autocorrelation of the difference vector
4. Find recurring peaks in the autocorrelation function

$$\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{8}$$

$$\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \mathrm{sum}_{\mathrm{CQ}}(t_n) - \mathrm{sum}_{\mathrm{CQ}}(t_{n+1}) \tag{9}$$

$$\mathrm{a}_{\mathrm{CQ}}(\mathrm{d}_{\mathrm{CQ}}(t_n), \tau) = \sum_{\tau=0}^{\tau_{\max}} \mathrm{d}_{\mathrm{CQ}}(t_n) * \mathrm{d}_{\mathrm{CQ}}(t_n - \tau) \tag{10}$$

Tempo in bpm (beats per minute)

- Intuitive: The speed at which humans tap when listening to a song
- Problem: That speed is not well-defined. Some persons tap at quarters, some at halves, ...

1. Take the sum of the constant-Q bins $\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
2. Calculate the difference vector $\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
3. Calculate the autocorrelation of the difference vector
4. Find recurring peaks in the autocorrelation function

$$\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{8}$$

$$\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \mathrm{sum}_{\mathrm{CQ}}(t_n) - \mathrm{sum}_{\mathrm{CQ}}(t_{n+1}) \tag{9}$$

$$\mathrm{a}_{\mathrm{CQ}}(\mathrm{d}_{\mathrm{CQ}}(t_n), \tau) = \sum_{\tau=0}^{\tau_{\max}} \mathrm{d}_{\mathrm{CQ}}(t_n) * \mathrm{d}_{\mathrm{CQ}}(t_n - \tau) \tag{10}$$

Tempo in bpm (beats per minute)

- Intuitive: The speed at which humans tap when listening to a song
- Problem: That speed is not well-defined. Some persons tap at quarters, some at halves, ...

1. Take the sum of the constant-Q bins $\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
2. Calculate the difference vector $\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
3. Calculate the autocorrelation of the difference vector
4. Find recurring peaks in the autocorrelation function

$$\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{8}$$

$$\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \mathrm{sum}_{\mathrm{CQ}}(t_n) - \mathrm{sum}_{\mathrm{CQ}}(t_{n+1}) \tag{9}$$

$$\mathrm{a}_{\mathrm{CQ}}(\mathrm{d}_{\mathrm{CQ}}(t_n), \tau) = \sum_{\tau=0}^{\tau_{\max}} \mathrm{d}_{\mathrm{CQ}}(t_n) * \mathrm{d}_{\mathrm{CQ}}(t_n - \tau) \tag{10}$$

Tempo in bpm (beats per minute)

- Intuitive: The speed at which humans tap when listening to a song
- Problem: That speed is not well-defined. Some persons tap at quarters, some at halves, ...

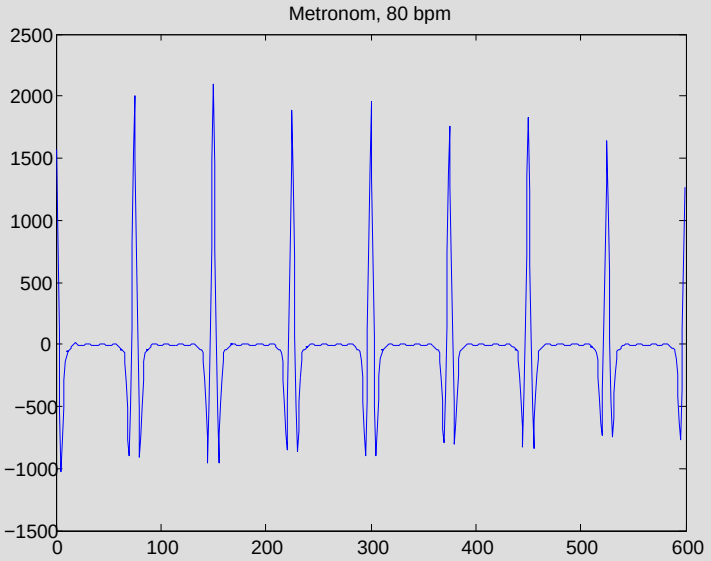
1. Take the sum of the constant-Q bins $\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
2. Calculate the difference vector $\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n)$
3. Calculate the autocorrelation of the difference vector
4. Find recurring peaks in the autocorrelation function

$$\mathrm{sum}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \sum_{b=0}^{B} |\boldsymbol{X}^{\mathrm{CQ}}(b, t_n)| \tag{8}$$

$$\mathrm{d}_{\mathrm{CQ}}(\boldsymbol{X}^{\mathrm{CQ}}, t_n) = \mathrm{sum}_{\mathrm{CQ}}(t_n) - \mathrm{sum}_{\mathrm{CQ}}(t_{n+1}) \tag{9}$$

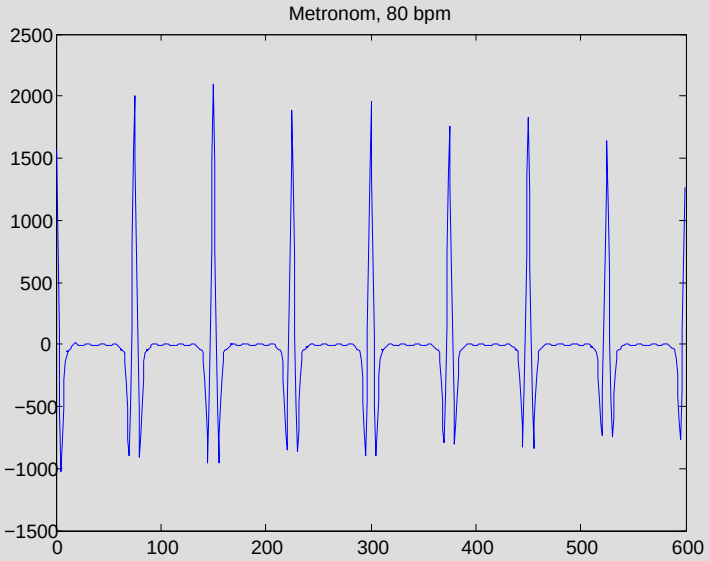
$$\mathrm{a}_{\mathrm{CQ}}(\mathrm{d}_{\mathrm{CQ}}(t_n), \tau) = \sum_{\tau=0}^{\tau_{\max}} \mathrm{d}_{\mathrm{CQ}}(t_n) * \mathrm{d}_{\mathrm{CQ}}(t_n - \tau) \tag{10}$$

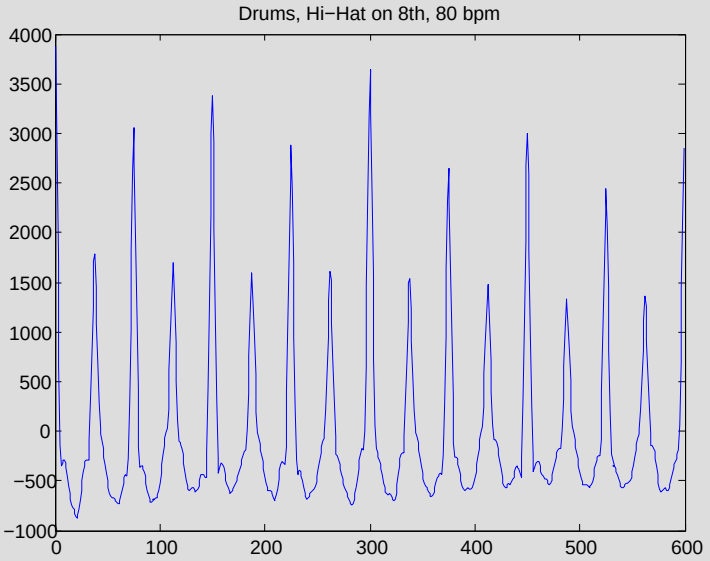
Tempo: Find recurring peaks



The unit of the absissica is $10\mu s$, the ordinate has no unit.

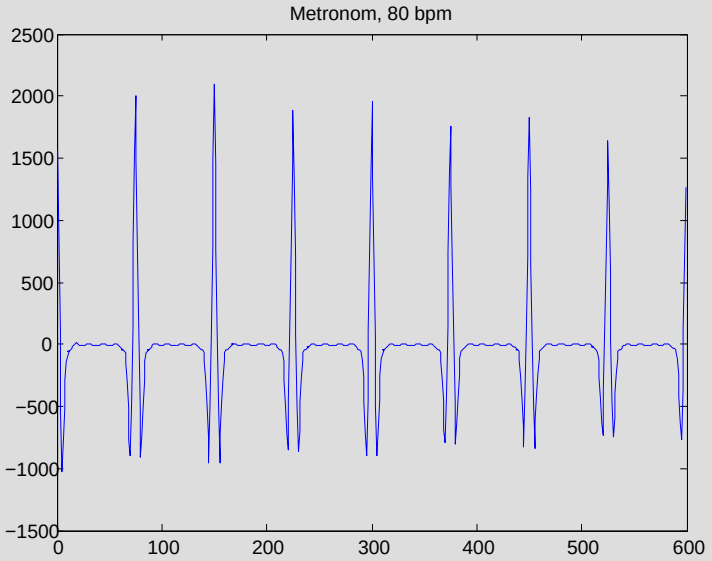
Tempo: Find recurring peaks

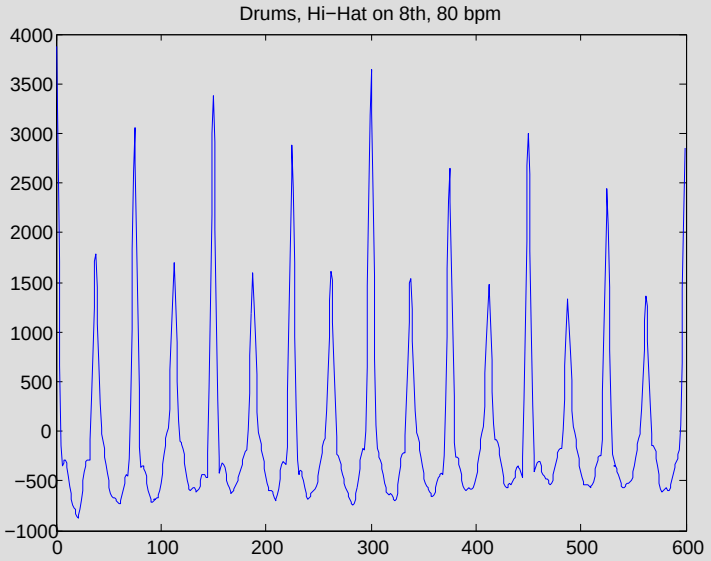


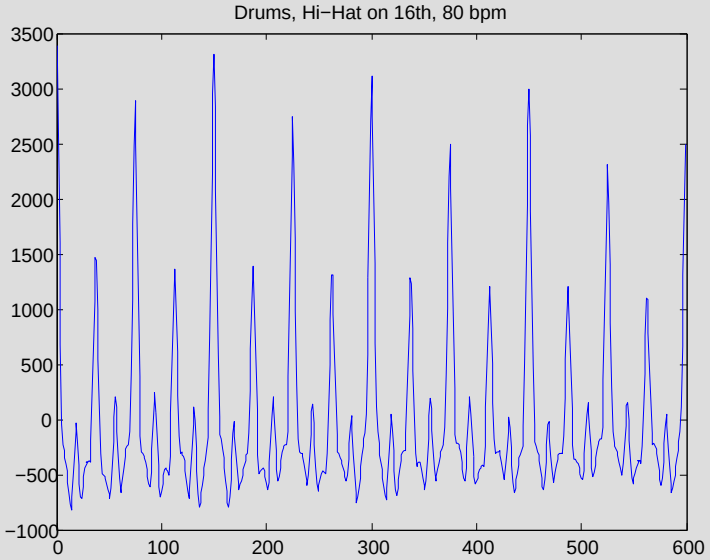


The unit of the absissica is $10\mu s$, the ordinate has no unit.

Tempo: Find recurring peaks

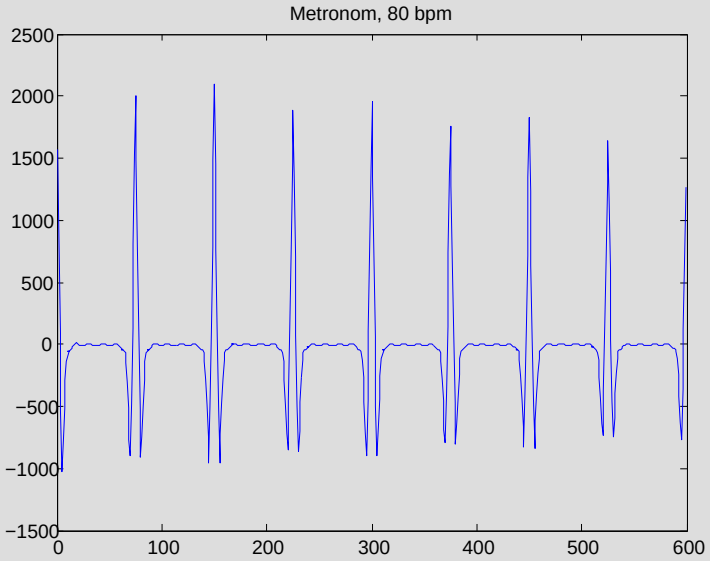


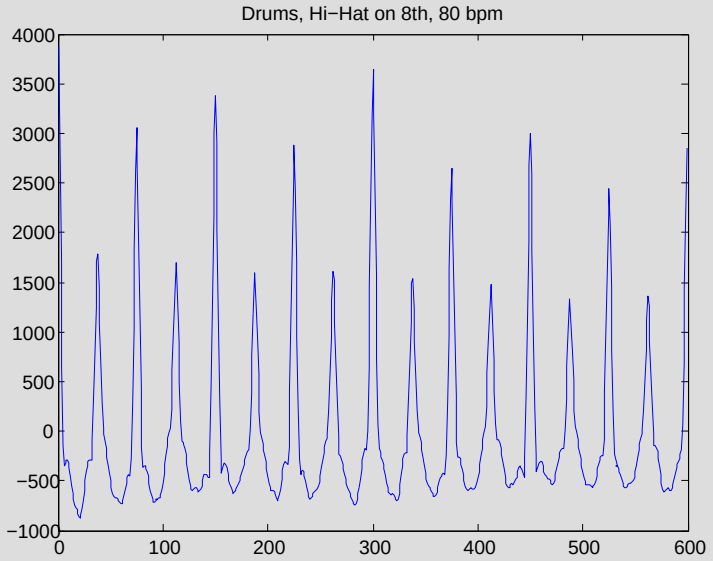


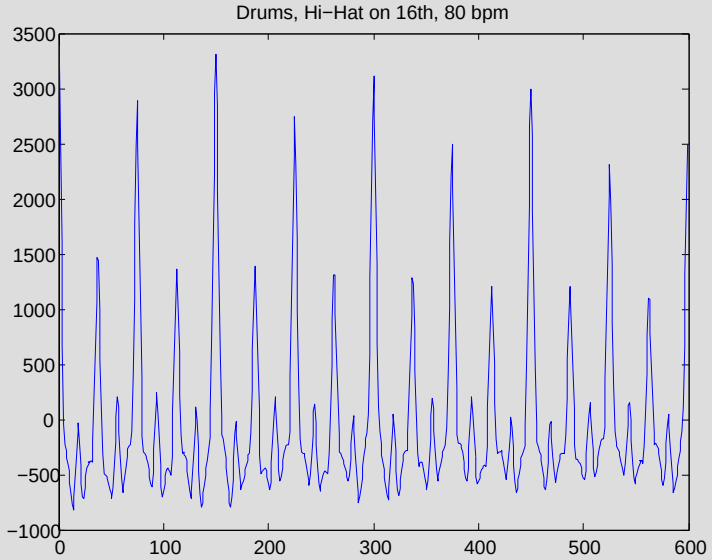


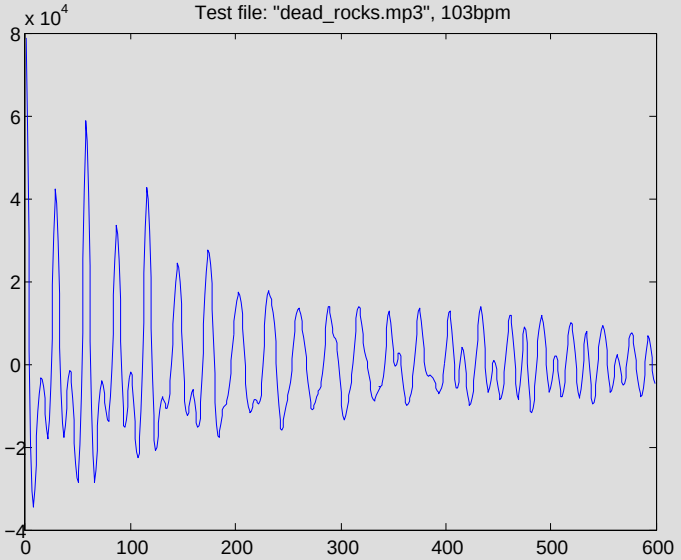
The unit of the absissica is $10\mu s$, the ordinate has no unit.

Tempo: Find recurring peaks









The unit of the absissica is $10\mu s$, the ordinate has no unit.

Timbre

- The timbre of a signal is “the way it sounds”
- It is a multi-dimensional feature
- In many publications, the Mel Frequency Cepstrum (MFC) is used
- The lower (e.g. 8-16) coefficients describe the timbre
- Short-time feature: typically one vector every 10-50ms
- The MFC is not based on the Constant-Q transform
- Similar features can be derived from the Constant-Q transform (see [11])

Timbre

- The timbre of a signal is “the way it sounds”
- It is a multi-dimensional feature
- In many publications, the Mel Frequency Cepstrum (MFC) is used
- The lower (e.g. 8-16) coefficients describe the timbre
- Short-time feature: typically one vector every 10-50ms
- The MFC is not based on the Constant-Q transform
- Similar features can be derived from the Constant-Q transform (see [11])

Timbre

- The timbre of a signal is “the way it sounds”
- It is a multi-dimensional feature
- In many publications, the Mel Frequency Cepstrum (MFC) is used
- The lower (e.g. 8-16) coefficients describe the timbre
- Short-time feature: typically one vector every 10-50ms
- The MFC is not based on the Constant-Q transform
- Similar features can be derived from the Constant-Q transform (see [11])

Timbre

- The timbre of a signal is "the way it sounds"
- It is a multi-dimensional feature
- In many publications, the Mel Frequency Cepstrum (MFC) is used
- The lower (e.g. 8-16) coefficients describe the timbre
- Short-time feature: typically one vector every 10-50ms
- The MFC is not based on the Constant-Q transform
- Similar features can be derived from the Constant-Q transform (see [11])

Timbre

- The timbre of a signal is "the way it sounds"
- It is a multi-dimensional feature
- In many publications, the Mel Frequency Cepstrum (MFC) is used
- The lower (e.g. 8-16) coefficients describe the timbre
- Short-time feature: typically one vector every 10-50ms
- The MFC is not based on the Constant-Q transform, but
- Similar features can be derived from the Constant-Q transform (see [11])

Timbre

- The timbre of a signal is "the way it sounds"
- It is a multi-dimensional feature
- In many publications, the Mel Frequency Cepstrum (MFC) is used
- The lower (e.g. 8-16) coefficients describe the timbre
- Short-time feature: typically one vector every 10-50ms
- The MFC is not based on the Constant-Q transform, but:
- Similar features can be derived from the Constant-Q transform (see [11])

Timbre

- The timbre of a signal is “the way it sounds”
- It is a multi-dimensional feature
- In many publications, the Mel Frequency Cepstrum (MFC) is used
- The lower (e.g. 8-16) coefficients describe the timbre
- Short-time feature: typically one vector every 10-50ms
- The MFC is not based on the Constant-Q transform, but:
- Similar features can be derived from the Constant-Q transform (see [11])

Timbre: Calculation of Constant-Q Cepstrum

Windowed Input signal	$w(t)x_C(t)$
↓	
Constant-Q Transform	$X^{\mathrm{CQ}}(k, t_n)$
↓	
Logarithm of absolute values	$\log(\lvert\odot\rvert)$
↓	
Discrete Cosine Transform	$\sum_{n=0}^{N-1} \odot \cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right)$
↓	
Constant-Q Cepstrum	take only lower coefficients

Timbre: Calculation of Constant-Q Cepstrum

Windowed Input signal	$w(t)x_C(t)$
↓	
Constant-Q Transform	$X^{\mathrm{CQ}}(k, t_n)$
↓	
Logarithm of absolute values	$\log(\lvert\odot\rvert)$
↓	
Discrete Cosine Transform	$\sum_{n=0}^{N-1} \odot \cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right)$
↓	
Constant-Q Cepstrum	take only lower coefficients

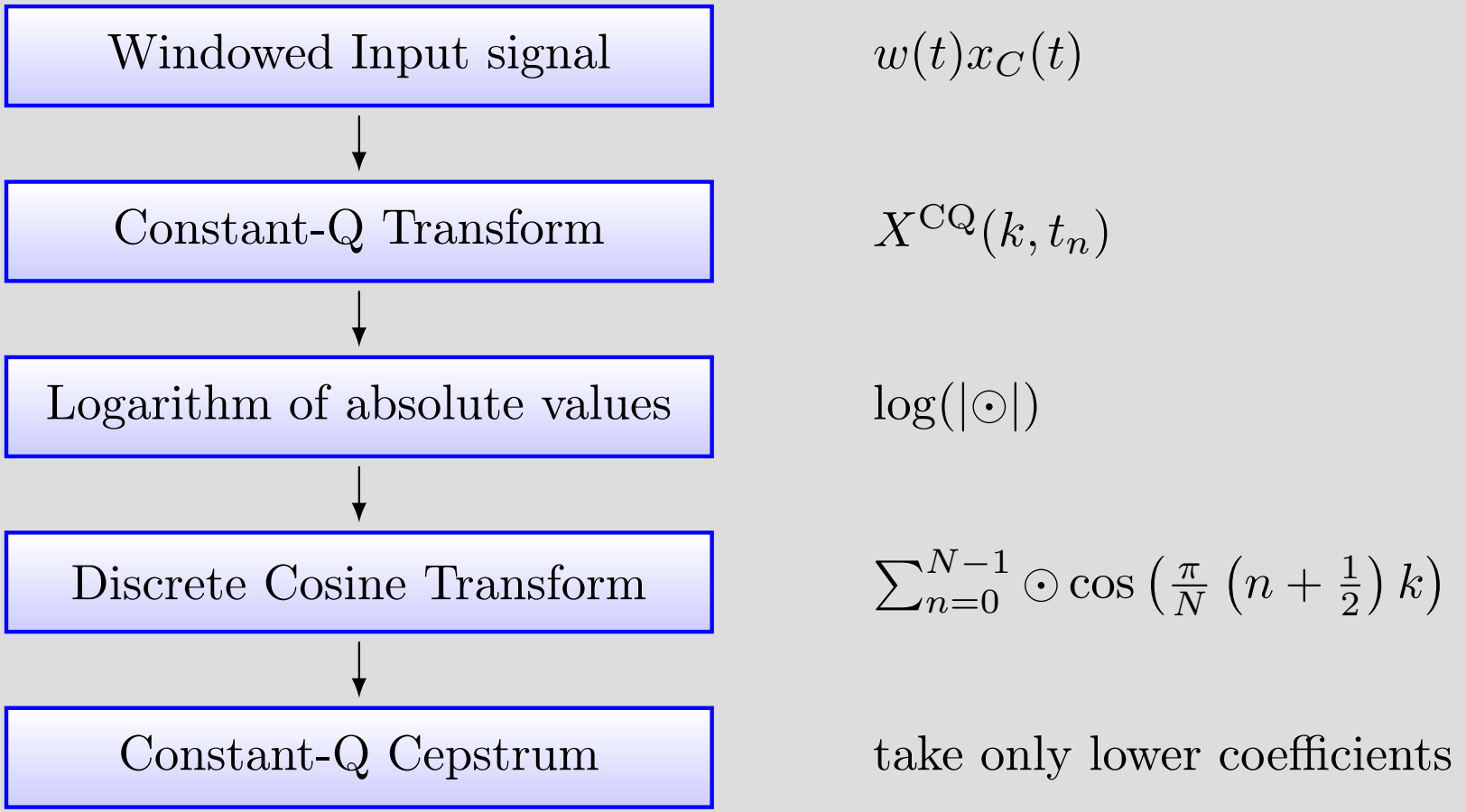
Timbre: Calculation of Constant-Q Cepstrum

Windowed Input signal	$w(t)x_C(t)$
↓	
Constant-Q Transform	$X^{\mathrm{CQ}}(k, t_n)$
↓	
Logarithm of absolute values	$\log(\lvert\odot\rvert)$
↓	
Discrete Cosine Transform	$\sum_{n=0}^{N-1} \odot \cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right)$
↓	
Constant-Q Cepstrum	take only lower coefficients

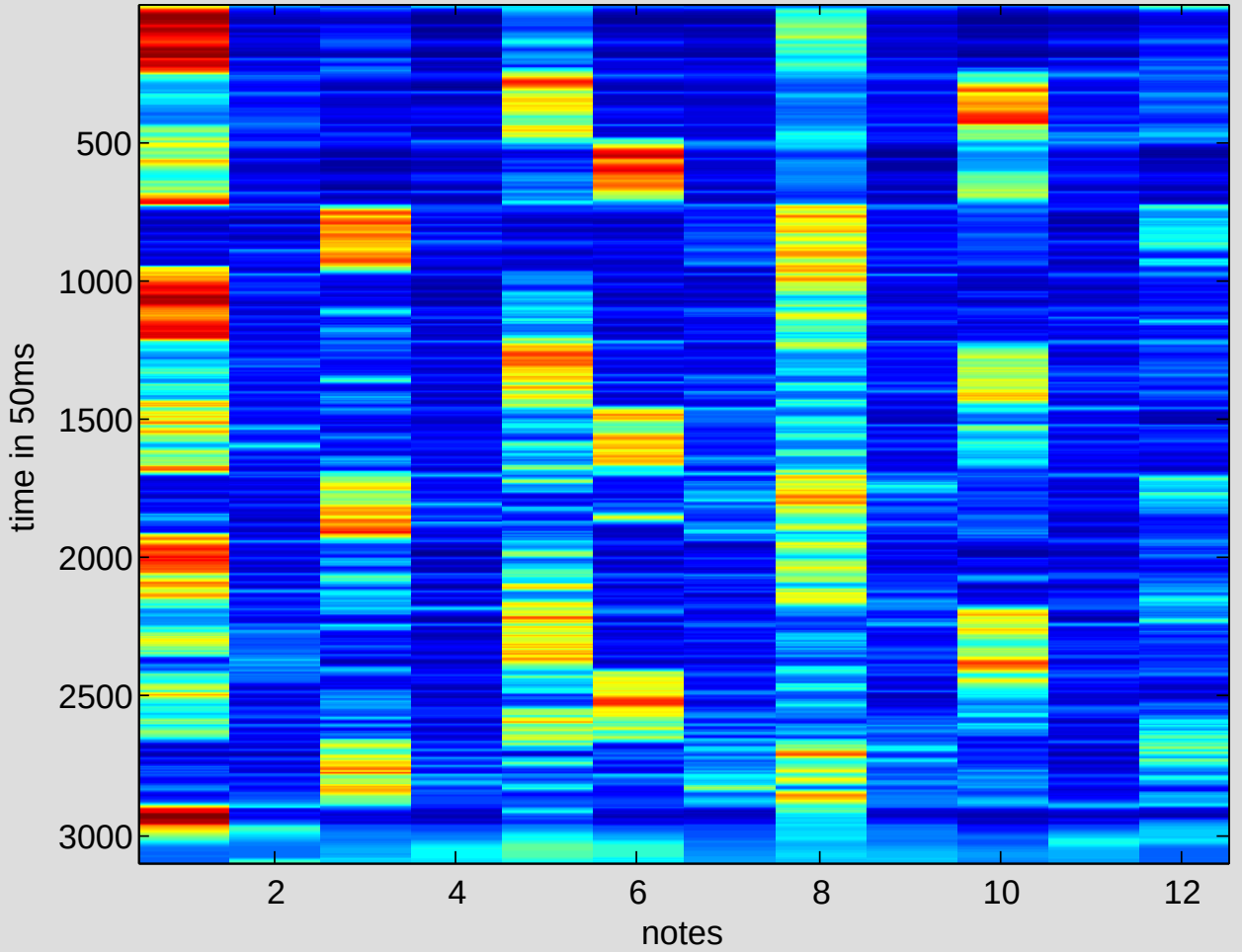
Timbre: Calculation of Constant-Q Cepstrum

Windowed Input signal	$w(t)x_C(t)$
↓	
Constant-Q Transform	$X^{\mathrm{CQ}}(k, t_n)$
↓	
Logarithm of absolute values	$\log(\lvert\odot\rvert)$
↓	
Discrete Cosine Transform	$\sum_{n=0}^{N-1} \odot \cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right)$
↓	
Constant-Q Cepstrum	take only lower coefficients

Timbre: Calculation of Constant-Q Cepstrum



Key-invariant chroma: Intuition



Key-invariant chroma: Definition of chroma

The chroma bin is defined as

$$c(b,t) = \sum_{p=0}^{P-1} |\boldsymbol{X}^{\mathrm{CQ}}(b+12p,t)| \tag{11}$$

where P is the number of octaves in the constant Q transform, b is one bin in an octave, and t is a point in time. The chroma vector is the vector of chroma bins

$$\boldsymbol{c}(t) = \begin{pmatrix} c(1,t) \\ c(2,t) \\ \vdots \\ c(12,t) \end{pmatrix}. \tag{12}$$

Key-invariant chroma: Key-invariance

fehlt noch

Plan

Bibliography I

Fabrizio Argenti, Paolo Nesi, and Gianni Pantaleo.
Automatic Transcription of Polyphonic Music based on the Constant-Q Bispectral Analysis.
IEEE Transactions on Audio, Speech and Language Processing, 19(6):1610–1630, August 2011.

American Standards Association.
Acoustical Terminology.
1960.

J.J. Aucouturier and F. Pachet.
Music similarity measures: What's the use.
In Proceedings of the 3rd International Symposium on Music Information Retrieval, page 157–163, 2002.

Bibliography II


Ehrhard Behrends.
Analysis, volume 2.
Vieweg, April 2004.


Juan P. Bello and Jeremy Pickens.
A Robust Mid-level Representation for Harmonic Content in Music Signals.
Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR-05), pages 304–311, September 2005.


G. Beylkin, R. Coifman, and V. Rohlkin.
Fast Wavelet Transforms and Numerical Algorithms I.
Communications on Pure and Applied Mathematics, (44):141–183, 1991.


Christopher M. Bishop.
Clarendon Press, Oxford, 1995.

Bibliography III

Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer, 2006.

Richard Brent and Paul Zimmermann.
Modern Computer Arithmetic.
Cambridge University Press, 2010.

Judith C. Brown.
Calculation of a constant Q spectral transformation.
J. Acoust Soc. Am., 89(1):425–434, January 1991.

Judith C. Brown.
Computer identification of musical instruments using pattern recognition with cepstral coefficients as features.
Acoustical Society of America, 105(3):1933–1941, March 1999.

Bibliography IV

Judith C. Brown and Miller S. Puckette.
An efficient algorithm for the calculation of a constant Q transform.
J. Acoust Soc. Am., 92(5):2698–2701, 1992.

Thomas H. Cormen, Charles E. Leierson, Ronald Rivest, and Clifford Stein.
Algorithmen - Eine Einführung.
Oldenbourg Verlag, München, 3. edition, 2010.

G. Cybenko.
Approximation by superpositions of a sigmoidal function.
Mathematics of Control, Signals, and Systems (MCSS), 2(4):303–314, 1989.

Zhouyu Fu, Kai Ming Ting, and Dengsheng Zhang.
A Survey of Audio-Based Music Classification and Annotation.
IEEE Transactions on Multimedia, 13(2):303–319, April 2011.

Bibliography V


James E. Gentle.
Random Number Generation and Monte Carlo Methods.
Springer, 2. edition, 2003.


Masataka Goto.
An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds.
Journal of New Music Research, 30(2):159–171, 2001.


Frederic J. Harris.
On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform.
Proceedings of the IEEE, 66:51–83, January 1978.

Bibliography VI


Jesper H. Jensen, Daniel P.W. Ellis, Mads G. Christensen, and Søren H. Jensen.
Evaluation of Distance Measures between Gaussian Mixture Models of MFCCs.
In ISMIR 2007: Proceedings of the 8th International Conference on Music Information Retrieval, pages 107–108, Vienna, September 2007.


Kristoffer Jensen.
Timbre Models of Musical Sounds.
PhD thesis, University of Copenhagen, 1999.


M. Karjalainen, V. Välimäki, and Z. Jánosy.
Towards high-quality sound synthesis of the guitar and string instruments.
In Proc. ICMC, pages 56–63, 1993.

Bibliography VII


T. Kinnunen, T. Kilpeläinen, and P. Fränti.
Comparison of clustering algorithms in speaker identification.
Proceedings IASTED Int. Conf. Signal Processing and Communications, 1:222–227, 2000.


Ulrich Krengel.
Einführung in die Wahrscheinlichkeitstheorie und Statistik.
vieweg studium, Wiesbaden, 8. edition, 2005.


Edward A. Lee.
The Problem with Threads.
Technical Report UCB/EECS-2006-1, EECS Department, University of California, Berkeley, Jan 2006.
The published version of this paper is in IEEE Computer 39(5):33-42, May 2006.

Bibliography VIII


N. Lesh and M. Mitzenmacher.
BubbleSearch: A simple heuristic for improving priority-based greedy algorithms.
Information Processing Letters, 97(4):161–169, 2006.


Beth Logan.
Mel Frequency Cepstral Coefficients for Music Modeling.
In International Symposium on Music Information Retrieval, volume 28, page 5, 2000.


K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf.
An introduction to kernel-based learning algorithms.
IEEE transactions on neural networks, 12(2):181–201, 2001.

Bibliography IX


Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf.
An Introduction to Kernel-Based Learning Algorithms.
IEEE Transactions on Neural Networks, 12(2):181–201, March 2001.


K. Noland and M. Sandler.
Key estimation using a hidden Markov model.
In Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR), page 121–126, 2006.


Alan V. Oppenheim and Ronald W. Schafer.
Zeitdiskrete Signalverarbeitung.
R. Oldenbourg Verlag, München, 2. edition, 1995.

Bibliography X


F. Pachet and J.J. Aucouturier.
"The way it Sounds": timbre models for analysis and retrieval of music signals.
IEEE Transactions on Multimedia, 7(6):1028–1035, 2005.


G. Peeters.
Chroma-based estimation of musical key from audio-signal analysis.
In Proc. of the 7th International Conference on Music Information Retrieval (ISMIR), page 115–120, 2006.


William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery.
Numerical Recipes.
2007.

Bibliography XI


Y. Rubner, C. Tomasi, and L.J. Guibas.
A metric for distributions with applications to image databases.
In Computer Vision, 1998. Sixth International Conference on, page 59–66. IEEE, 1998.


Ingo Schnitt.
Ähnlichkeitssuche in Multimedia-Datenbanken: Retrieval, Suchalgorithmen und Anfragebehandlung.
Oldenbourg Wissenschaftsverlag, 2005.


Dominik Schnitzer.
Indexing Content-Based Music Similarity Models for Fast Retrieval in Massive Databases.
PhD thesis, Johannes Kepler Universität Linz, October 2011.

Bibliography XII


Christian Schörkhuber and Anssi Klapuri.
Constant-Q transform toolbox for music processing.
In 7th Sound and Music Computing Conference, Barcelona, Spain, 2010.


Josef Stoer.
Numerische Mathematik 1.
Springer, Berlin, Heidelberg, New York, 9 edition, 2005.


Josef Stoer and Roland Burlisch.
Numerische Mathematik 2.
Springer, Berlin, Heidelberg, New York, 5 edition, 2005.


Wolfgang Theimer, Igor Vatolkin, and Antti Eronen.
Definitions of Audio Features for Music Content Description.
Technical report, February 2008.

Bibliography XIII


Wolfgang Theimer, Igor Vatolkin, Rainer Martin, Christian Igel, Holger Blume, Bernd Bischl, Martin Botteck, Günther Roetter, Günther Rudolph, and Claus Weihs.
Huge Music Archives on Mobile Devices.
IEEE Signal Processing Magazine, page 24–39, July 2011.


George Tzanetakis and Perry Cook.
Musical Genre Classification of Audio Signals.
IEEE Transactions on Speech and Audio Processing, 10(5):293–302, July 2002.


R. W. Young.
Terminology for Logarithmic Frequency Units.
Acoustical Society of America Journal, 11:134, 1939.

Bibliography XIV


S. Ystad, P. Guillemain, and R. Kronland-Martinet.
Estimation of parameters corresponding to a propagative synthesis model through the analysis of real sounds.
In Proc. ICMC, 1996.


Wieland Ziegenrücker.
ABC Musik.
Breitkopf & Härtel, Wiesbaden, 3. edition, 2000.