jSymbolic: Demonstration and Tutorial

Cory McKay
Marianopolis College and CIRMMT
Topics

- Introduction to “features”
- Introduction to jSymbolic
- jSymbolic demo and tutorial
What are “features”?

- Pieces of information that can characterize something (e.g. a piece of music) in a simple way
- Usually numerical values
  - A feature can be a single value, or it can be a set of related values (e.g. a histogram)
- Can be extracted from pieces as a whole, or from segments of pieces
Example: Two basic features

- **Range (1-D):** Difference in semitones between the highest and lowest pitches.
- **Pitch Class Histogram (12-D):** Each of its 12 values represents the fraction of notes of a particular pitch class. The first value corresponds to the most common pitch class, and each following value to a pitch class a semitone higher than the previous.

Range = G - C = 7 semitones

Pitch Class Histogram: see graph ->
- Note counts: C: 3, D: 10, E: 11, G: 2
- Most common note: E (11/26 notes)
  - Corresponding to 0.423 of the notes
- E is thus pitch class 1, G is pitch class 4, C is pitch class 9, D is pitch class 11
Josquin’s Ave Maria... Virgo serena

- **Range:** 34
- **Repeated notes:** 0.181
- **Vertical perfect 4ths:** 0.070
- **Rhythmic variability:** 0.032
- **Parallel motion:** 0.039
Ockeghem’s *Missa Mi-mi* (Kyrie)

- **Range:** 26
- **Repeated notes:** 0.084
- **Vertical perfect 4ths:** 0.109
- **Rhythmic variability:** 0.042
- **Parallel motion:** 0.076

**Misa Mi-mi: PC Histogram**
## Feature value comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Ave Maria</th>
<th>Misa Mi-mi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>34</td>
<td>26</td>
</tr>
<tr>
<td>Repeated notes</td>
<td>0.181</td>
<td>0.084</td>
</tr>
<tr>
<td>Vertical perfect 4\textsuperscript{ths}</td>
<td>0.070</td>
<td>0.109</td>
</tr>
<tr>
<td>Rhythmic variability</td>
<td>0.032</td>
<td>0.042</td>
</tr>
<tr>
<td>Parallel motion</td>
<td>0.039</td>
<td>0.076</td>
</tr>
</tbody>
</table>

### Ave Maria: PC Histogram
![Ave Maria: PC Histogram](image1)

### Misa Mi-mi: PC Histogram
![Misa Mi-mi: PC Histogram](image2)
Feature visualization: Histograms (1/4)

- **Histograms** are one good way to visualize how the values of a feature are distributed across a corpus as a whole
  - As opposed to focusing on individual pieces
- The **x-axis** corresponds to a series of bins, with each corresponding to a **range of values** for a given feature
  - e.g. the first bin could correspond to Parallel Motion feature values between 0 and 0.1, the next bin to Parallel Motion values between 0.1 and 0.2, etc.
- The **y-axis** indicates the **fraction of all pieces** that have a feature value within the range of each given bin
  - e.g. if 30% of pieces in the corpus have Parallel Motion values between 0.1 and 0.2, then this bin (0.1 to 0.2) will have a y-coordinate of 30% (or, equivalently, 0.3)
Feature visualization: Histograms (2/4)

- In other words:
  - Each bar on a histogram represents the fraction of pieces in a corpus with a feature value falling in that bar’s range of feature values

- Clarification: I am speaking here about a way to visualize a 1-dimensional feature as it is distributed across a corpus of interest
  - This is distinct from the multi-dimensional histogram features discussed earlier
    - e.g. Pitch Class Histograms
  - Although both are equally histograms, of course
These histograms show that Ockeghem tends to have more vertical 6ths (between all pairs of voices) than Josquin:
- Ockeghem peaks in the 0.16 to 0.17 bin
- Josquin peaks in the 0.13 to 0.14 bin

Of course, there are also clearly many exceptions:
- This feature is helpful, but is limited if only considered alone
The histograms for both composers can also be superimposed onto a single chart:
Feature visualization: Scatter plots (1/6)

- **Scatter plots** are another good way to visualize feature data
  - The *x-axis* represents one feature
  - The *y-axis* represents some other feature
  - Each **point** represents the values of these two features for a single piece

- Scatter plots let you see pieces **individually**, rather than aggregating them into bins like histograms
  - Scatter plots also let you see more clearly how the two features **divide** the different composers

- To make them easier to read, scatter plots typically have just **2 dimensions**
  - Computer classifiers, in contrast, work with **n-dimensional** scatterplots (one dimension per feature)
Josquin pieces tend to be left and low on this graph.
Feature visualization: Scatter plots (3/6)

- Simply drawing a single 1-D dividing line ("discriminant") results in a not entirely terrible classifier based only on Vertical Sixths!
  - But many pieces would still be misclassified
  - Get 62% classification accuracy using an SVM and just this one feature
Feature visualization: Scatter plots (4/6)

Could alternatively draw a 1-D discriminant dividing the pieces based only on the **Average Length of Melodic Arcs**
- Get 57% classification accuracy using an SVM and just this one feature
- Not as good as the **Vertical Sixths** discriminant (62%)
Feature visualization: Scatter plots (5/6)

- Drawing a curve (another kind of discriminant) divides the composers still better than either of the previous discriminants
  - Get 80% accuracy using an SVM and just these 2 features!
- More than 2 features are clearly needed to improve performance
Feature visualization: Scatter plots (6/6)

- In fact, many (but not all) types of machine learning in effect simply learn where to place these kinds of discriminants as they train.
- But typically with many more than just two features, of course.
jSymbolic: Introduction

- **jSymbolic** is a software platform I have implemented for extracting features from symbolic music
  - Part of our much larger jMIR package
- Compatible with *Macs, PCs and Linux* computers
- Free and open-source
What does jSymbolic do?

- Extracts **246 unique features**
- Some of these are **multi-dimensional histograms**, including:
  - Pitch and pitch class histograms
  - Melodic interval histograms
  - Vertical interval histograms
  - Chord types histograms
  - Rhythmic value histograms
  - Beat histograms
  - Instrument histograms
- In all, extracts a total of **1497 separate values**
jSymbolic: Extensibility

- jSymbolic is specifically designed such that music scholars can design their own features and work with programmers to then very easily add these features to the jSymbolic infrastructure
  - Fully open source
  - Modular plug-in feature design
  - Automatically handles feature dependencies and scheduling
  - Very well-documented code
jSymbolic demo

- Web site
- Manual
- Tutorial
- GUI
  - API and command line interface
- Configuration files
- Manually examining features
- Analyzing features with Weka
- Looking at the code
  - Adding new features
Thanks for your attention!

- **jSymbolic**: http://jmir.sourceforge.net
- **E-mail**: cory.mckay@mail.mcgill.ca