Program #3
February 11, 2018

CS5720: Data Visualization
R. M. Parry

Music Visualization

For this assignment you will produce a two-dimensional visualization of major and minor chords that places similar chords near each other and dissimilar chords far away from each other.

1 Theory

The Spotify track analysis API provides pitch information for each short segment of audio within a song. The segments are chosen to roughly represent a note or chord, and each beat is made up of one or more of these segments. The pitch information is provided as a 12-vector invariant to octave. If we wanted to listen to these pitch vectors, it might sound like a combination of Shepard tones. [Check out this auditory illusion of constantly increasing pitch using Shepard tones.] A Shepard tone is generated by combining sine waves with octave spaced frequencies. For example, middle A is defined as 440 Hz. Its Shepard tone might contain a weighted combination of sine waves at 27.5, 55, 110, 220, 440, 880, 1760, 3520, and 7040 Hz with frequencies in the middle of that range given more weight.

Instead of listening to them, we want to visualize their similarity. It makes sense that we would want similar pitches to appear close to each other and that dissimilar pitches to be farther part but this is hard to see in the 12-dimensional space. Research suggests that for major and minor chords, this 12-vector can be efficiently represented in two-dimensions. We will use a self-organizing map to do the projection and visualize the result.

Music theory already gives us some clues about what keys are similar. The circle of fifths places the major and minor keys in a specific order with similar keys as neighbors around a circle. Here are the notes in the order they appear on a piano:

Table 1: 12-tone Chromatic Scale

C	C#	D	Eb	E	F	F#	G	Ab	A	Bb	B

The next note in the series wraps back around to the C one octave above, then C# one octave above and so on. The most similar keys are those that are separated by five steps. For example, C's neighbors are five steps up (F) and five steps down (G). Stepping through the circle five steps at a time produces the circle of fifths:

Table 2: Circle of Fifths

Major:	C	G	D	A	E	B	F#	C#	Ab	Eb	Bb	F
Minor:	A	E	B	F#	C#	Ab	Eb	Bb	F	C	G	D

2 Simple representation (S-patterns)

Marc Leman describes an *S-representation* that indicates which Shepard tones are present in a chord:

Table 3: S-patterns

Abbrev.	Name	C	C#	D	Eb	E	F	F#	G	Ab	A	Bb	B
C	Single note	1	0	0	0	0	0	0	0	0	0	0	0
CM	Major triad	1	0	0	0	1	0	0	1	0	0	0	0
Cm	Minor triad	1	0	0	1	0	0	0	1	0	0	0	0

3 Residue representations (R-patterns)

Each note has subharmonics at 1/2, 1/3, 1/4, 1/5, ... of the frequency. For example, the first eight subharmonics of middle A (A_4) are the following: 220 (A_3, octave), 147 (D_3, fifth), 110 (A_2, octave), 88 (F_2, major third), 73 (D_2, fifth), 63 (B_1, minor seventh), 55 (A_1, octave), 49 Hz (G_1, major second). Notice that A shows up three times, D shows up two times, with F, and B also represented. Leman suggests weighting these according to their count and strength by the following:

Table 4: Subharmonic Weights

Abbrev.	Name	Octave (C)	Fifth (F)	Major third (Ab)	Minor seventh (D)	Major second (Bb)	Minor third (A)
C	Single note	1.00	0.50	0.33	0.25	0.20	0.10

The various weights combine to form a 12-vector capturing the similarity based on subharmonics. For example, the subharmonic sum of C-Major:

Table 5: Subharmonic Sums

S-image	1	0	0	0	1	0	0	1	0	0	0	0
octave(1.00)	C	-	-	-	E	-	-	G	-	-	-	-
fifth(0.50)	G	-	-	-	-	C	-	-	-	E	-	-
maj.third(0.33)	E	-	-	G	-	-	-	-	C	-	-	-
min.sev.(0.25)	-	-	C	-	-	-	E	-	-	G	-	-
maj.sec.(0.20)	-	-	E	-	-	G	-	-	-	-	C	-
min.third(0.10)	-	E	-	-	G	-	-	-	-	C	-	-
R-image	1.83	0.10	0.45	0.33	1.10	0.70	0.25	1.00	0.33	0.85	0.20	0.00
Notes:	C	C#	D	Eb	E	F	F#	G	Ab	A	Bb	B

The resultant 12-vector R-pattern captures the musical similarity between pitches composed of Shepard tones. Other R-patterns can be assembled by rotation:

4 Normalized representation (O-patterns)

Spotify pitch vectors have been normalized to have a maximum value of 1.0. In order to compare pitch vectors that could have a large magnitude difference, Leman normalized each vector using the following equation:

$$O_i = \frac{R_i}{R_max} \times \frac{1}{\sqrt{\sum_{i=1}^{12} \frac{R_i}{R_max}}} \tag{1}$$

This results in the following O-patterns:

Table 6: R-patterns

Abbrev.	Name	C	C#	D	Eb	E	F	F#	G	Ab	A	Bb	B
C	Single note	1.00	0.00	0.25	0.00	0.00	0.50	0.00	0.00	0.33	0.1	0.2	0.00
CM	Major triad	1.83	0.10	0.45	0.33	1.10	0.70	0.25	1.00	0.33	0.85	0.20	0.00
Cm	Minor triad	1.60	0.20	0.25	1.33	0.10	0.95	0.00	1.00	0.83	0.35	0.20	0.33

Table 7: O-patterns

Abbrev.	Name	C	C#	D	Eb	E	F	F#	G	Ab	A	Bb	B
C	Single note	0.65	0.00	0.16	0.00	0.00	0.32	0.00	0.00	0.21	0.06	0.13	0.00
CM	Major triad	0.51	0.03	0.12	0.09	0.30	0.19	0.07	0.28	0.09	0.24	0.06	0.00
Cm	Minor triad	0.47	0.06	0.07	0.39	0.03	0.28	0.00	0.30	0.25	0.10	0.06	0.10

5 Data

You will visualize the major triad and minor triad for every pitch (24 total O-patterns). You can form the patterns for pitches other than C by circularly shifting the values. You can download the CSV file here: `http://cs.appstate.edu/~rmp/cs5720/chords.csv`

6 Self-organizing Maps (SOM)

An overview of self-organizing maps can be found here:
`http://en.wikipedia.org/wiki/Self-organizing_map`

For this assignment, you will implement the following pseudocode:

for all $i \in [1, p]$, $j \in [1, p]$, $k \in [1, m]$ **do**
  $w_{ijk} \leftarrow U(0, 1)$
end for
for all $s \in [1, \lambda]$ **do**
  for all $t \in [1, n]$ **do**
    $\mathbf{r}_t \leftarrow$ random input vector
    $i, j \leftarrow$ index of best matching unit that minimizes $\|\mathbf{w}_{ij} - \mathbf{r}_t\|$
    for all $k \in [1, p]$, $l \in [1, p]$ **do**
      $\sigma \leftarrow \frac{1}{3}\left(p - 1 - \frac{s}{c}\right)$ is the size of the neighborhood
      $\theta \leftarrow e^{-\frac{d(i,j,k,l)}{2\sigma^2}}$ is the neighborhood function
      $\mathbf{w}_{kl} \leftarrow \mathbf{w}_{kl} + \theta\alpha(\mathbf{r}_t - \mathbf{w}_{kl})$
    end for
  end for
end for

Because the grid of neurons connects the left edge to the right edge and the top edge to the bottom edge (forming a torus), the distance function d must be computed with that in mind. So, the upper-left corner is a distance of one away from the upper-right; and the upper-left is a distance of $\sqrt{2}$ away from the lower-right.

Table 8: Parameters

Symbol	Description	Value
n	Number of inputs	24
m	Number of attributes	12
p	Size of each side of map	20
$\mathbf{W}$	The map	$20 \times 20 \times 12$ matrix
$\mathbf{w}_{ij}$	Weight vector at index (i, j)	Any of the 400 12-dimensional weight vectors
t	Index of target input vector	$[1, 24]$
$\mathbf{r}_t$	Target input vector	Any of the 24 vectors
s	Current iteration	$[1, \lambda]$
λ	Iteration limit	360
c	Neighborhood radius decrement period	20
i, j, k, l	Indices of the map	[1,20]
$\alpha(s)$	Learning rate	0.02

7 Submission

This time you will submit your code using GitHub. Login to GitHub and browse to the class repository: `https://github.com/rmparry7/cs5720_share`

Fork the repository by clicking on the "Fork" button in the upper-right corner. This creates a copy of the repository in your user account on GitHub that you should make private. You can clone that repository using the normal technique to a computer where you are working on your assignment. Create a directory `cs5720_share/visuals/music_som/<your_asu_username>/`. Put your source code there and write a bash script to run it. The script should take four arguments from the command line: the input CSV file and the name of the output PNG files.

As before, create a bash script to run your program. For example, if you write a Python program the bash script should look exactly like this:

```
python music_som.py "$@"
```

The `"$@"` passes all the command line arguments to your Python file. If you decide to use R, it should look exactly like this:

```
Rscript music_som.R "$@"
```

The program should produce 3 images with names taken from the command line. The first argument (`music_som.png`) should label every node with its closest match from the input using the labels `CM` or `Cm` or other chords as appropriate. The second argument (`music_som_CM.png`) shows a heatmap of the activations for CM across the map. That is, a heatmap of the dot-product between the O-pattern for CM and the weights at each node. Finally, the third input argument (`music_som_Cm.png`) provides the activation heatmap for C-minor.

When you have committed all your changes to your forked repository, test it by changing to the `cs5720_share` directory and running:

```
$ bash run_visuals.sh
```

If it runs successfully, it should create the three PNG files in your directory:
`cs5720_share/visuals/music_som/<your_asu_username>/` When it works and your changes are committed and pushed to your forked repository, login to GitHub and click "Create pull request". If you make later changes, just cancel the initial pull request and create a new one.