dissimilarities2

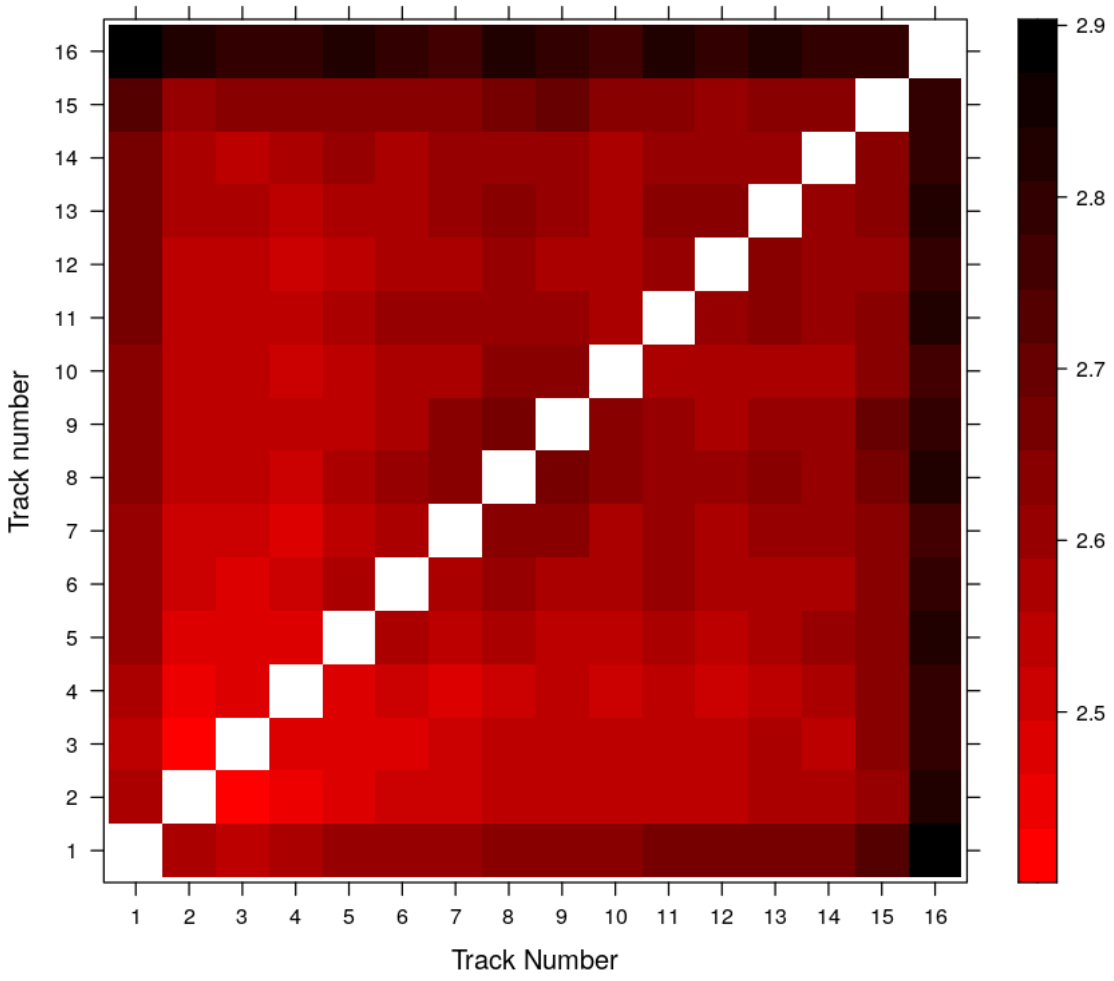
December 14, 2020

1 Nearest-neighbor interpolation

I read about different interpolation techniques that we can use in order to upsample album lengths. Here are some notes that I took to myself, but which might help you interpret the results.

- We are only upsampling the number of tracks for each album (all albums have 4 features per track, and they stay that way);
- It's like stretching an image only in the y axis.
- In our case, bilinear and bicubic interpolation - which interpolates between both x and y axis - become simple linear and cubic interpolations (value of the x axis doesn't affect the interpolated value on the y axis), so we don't need to worry about leakage of information between features.
- Still, using linear/cubic interpolation between two tracks in an album will cause the interpolated value to be a function of both tracks i and $i + 1$.
- I don't know if this is acceptable in our case, since we are mostly interested in the contrast between neighbor tracks/sections. Something to think about;
- Given this consideration, I chose to go with the simplest interpolation technique: nearest neighbors.
- This is close to - but much better than - my intuitive upsampling implementation, since interpolated cells only repeat the value that is closest on the matrix.

1.1 Results



1.2 Interpretation

Here we see a kind of cluster in the beginning of the albums - coordinates $(7, 7)$. Most interestingly is the fact that dissimilarity scores are higher for tracks 1 and 16. Next I'm implementing the permutation test.

Compare with original dissimilarity matrices, which we calculated without upsampling, but averaging accross albums.

2 Permutation test

2.1 Procedures:

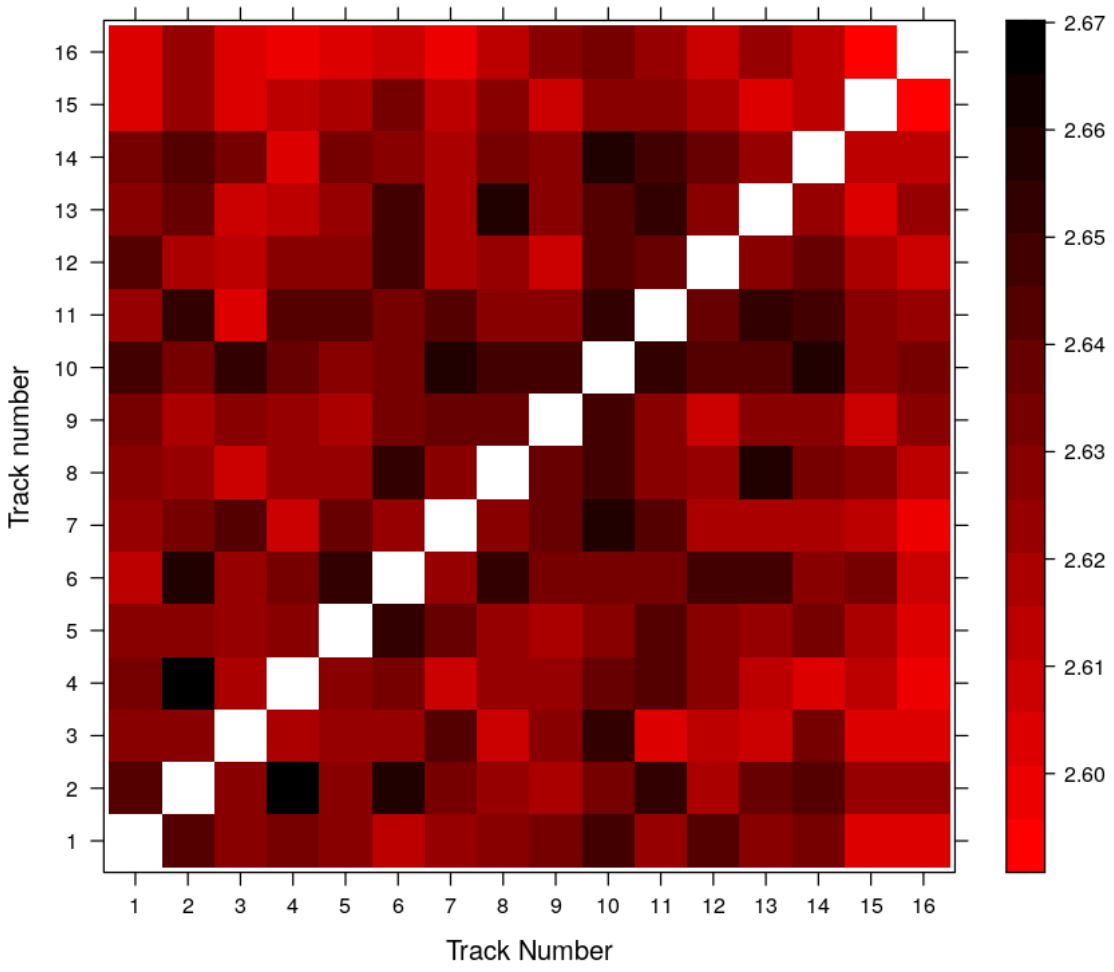
- Nearest-neighbor interpolation of each album - generates 16x16 matrices;
- Shuffle all upsampled albums within itself;
- Compute dissimilarity matrix for each album;
- Compute the average dissimilarity for each cell of each matrix;
- Record and repeat the process 10k times;
- Compute mean and sd for each cell of the dissimilarity matrix generated throughout the 10k runs. (this gives a total of 10k*number of albums in our sample).
- Subtract the difference between our original upsampled dissimilarity matrix and divide it with the sd.

2.2 Results

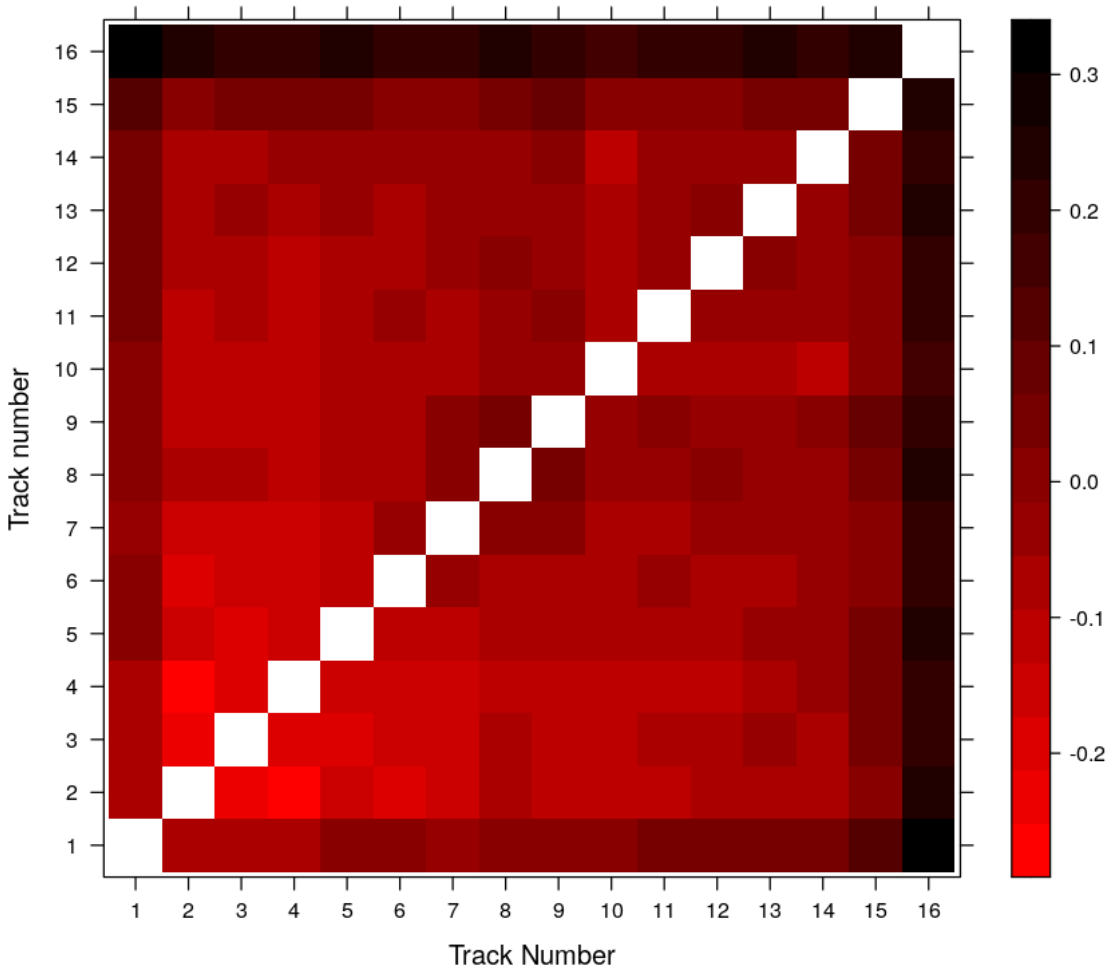
2.2.1 Permutated dissimilarity means

Actual permutation happened 4254 times. Even after trying to optimize the code, each dissimilarity matrix took 16 seconds to process. This yiealded a 52 hours-long processing time.

I was able to let it run for about half of that. This means that I permutated 17249970 albums ($4254 * number\ of\ albums$). This permutation gave us the matrix below, which shows the mean dissimilarity for each pair of tracks.


Track number
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Track Number
2.67
2.66
2.65
2.64
2.63
2.62
2.61
2.60

2.2.2 Normalized matrices



2.2.3 Interpretation

There seems to be two clusters arround $(7, 7)$ and $(10, 10)$. These clusters are formed arround the edges of the albums. Perhaps this explains the imbalence in class accuracy that we had in Models 3 and 4.

Most interestingly, in my opinion, is the fact that opening and closing tracks (i.e. 1 and 16) are the most dissimilar ones.

Note: We can't interpret track numbers so directly, because albums are upsampled to 16. Edges, however, are still edges (tracks 1 and 16 are the first and last tracks of each album).