Nonlinear Clustering for Music with t-SNE

Sara Summerton
10075097

December 3, 2018

Music can be said to be more than a collection of sounds. Perhaps we can say that one song sounds similar to another because of a melody or collection of instruments used, but describing exactly *how* different say, Joni Mitchell's "Blue" is from David Bowie's "Starman" can be quite difficult. In this report I try a technique called t-distributed Stochastic Neighbor Embedding (t-SNE) to try to visualise the similarities of songs from my music library. Such a technique could be useful and interesting as a starting point for e.g. how services such as YouTube suggest similar songs. It was found that the algorithm was very sensitive to initialisation parameters, and while results could perhaps be improved by higher resolution reduction of the song data, the generated plots did display some expected clustering of songs.

Theory

t-SNE is a dimensionality reduction technique that maps each point in n-dimensional space to a point in 2-dimensional space. It is a nonlinear clustering algorithm that attempts to both preserve local variation as well as reveal global structural patterns. Specifically, for a data set $\mathbf{X} = \{x_1, x_2, \dots, x_n\}$, cost function parameter *perplexity*, and optimisation parameters iterations T, learning rate η, and momentum $\alpha(t)$, the algorithm aims to produce a low-dimensional data representation $\mathbf{Y}^{(t)} = \{y_1, y_2, \dots, y_n\}$. The basic steps are as follows:

- Compute pairwise affinities $p_{j|i}$ with given perplexity using Gaussian joint probabilities,

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_j||^2/2\sigma_i^2)}.$$

- Set $p_{ij} = \dfrac{p_{j|i} + p_{i|j}}{2n}$.
- Initialise a solution $\mathbf{Y}^{(0)} = \{y_1, y_2, \dots, y_n\}$.
- Compute affinities in the low-dimensional space using a Cauchy distribution, which has larger tails than a Gaussian, where the joint probabilities q_{ij} are

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}.$$

- Do the previous step for each iteration, using gradient descent to minimise the G^2 or KL divergence of the joint probabilities in the original space and the embedded space.

Data and methodology

This project made use of the python library LibROSA, which is designed to analyse music and audio files. Included artists were chosen for whom the entire discography was available, and no

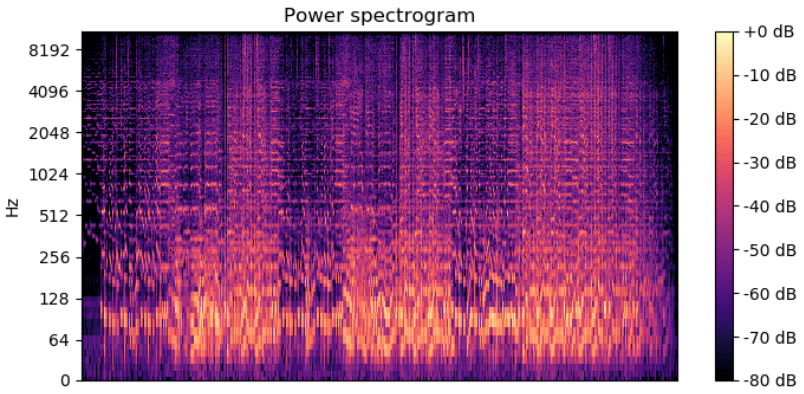


Figure 1: The power spectrogram for "Lucy in the Sky with Diamonds" by The Beatles.

songs were deleted so as not to inadvertently introduce bias. The only exception for this was Mozart, for whom songs from the "100 Best Mozart" collection were used. This did result in large variations between the number of songs for each artist, as seen below:

Artist	**Number of songs**	**Artist**	**Number of songs**
Aphex Twin	224	Autechre	194
The Beatles	157	Beyoncé	62
Charles Mingus	51	David Bowie	90
Joni Mitchell	222	Kanye West	100
Madonna	209	Mastodon	65
Miles Davis	102	Mozart	100
Pink Floyd	116		

The following features (mostly suggested from the librosa documentation) were extracted for each song:

- Mean and standard deviation of the spectral centroid, e.g.

```
centroid=librosa.feature.spectral_centroid(song, sample_rate)
centroid_mean=np.mean(centroid)
centroid_std=np.std(centroid)
```

- Mean and standard deviation of the p^{th}-order spectral bandwidth
- Mean and standard deviation of the spectral contrast [R16]
- Mean and standard deviation of the zero crossing rate
- BPM

Each of the songs was split into harmonic and percussive bands and the power spectrogram was computed.

```
y_harmonic, y_percussive = librosa.effects.hpss(song)
stft_harmonic=librosa.core.stft(y_harmonic)
stft_percussive=librosa.core.stft(y_percussive)
```

The following features (again from the librosa documentation) were then extracted from the bands:

- Root-mean-square and standard deviation energy for harmonic and percussive spectrogram

- Mean and standard deviation of coefficients of fitting an nth-order polynomial to the columns of the harmonic spectrogram
- Mean and standard deviation of tonal centroid features (tonnetz) for harmonic bands
- The onset of the percussive band
- The skew and kurtosis of the harmonic and percussive spectrograms

Additionally, each spectrogram was split into 5 frequency bands between 30 Hz to 10KHz on a log scale:

```
band_resolution=[5]

for no_bands in band_resolution:
  bands=np.logspace(1.3,4,num_bands)/10
  bands_int=bands.astype(int)

  for i in range(0,len(bands_int)-1):
    _h=librosa.feature.rmse(S=(splitF(bands[i],bands[i+1],stft_harmonic)))
    _p=librosa.feature.rmse(S=(splitF(bands[i],bands[i+1],stft_percussive)))

    #Calculate statistics for harmonic and percussive over time series, e.g.
    rms_h=np.mean(np.abs(_h))
```

Note that by construction, bands/10 = frequency. For each of these bands the root mean square, standard deviation, skew, and kurtosis were calculated.

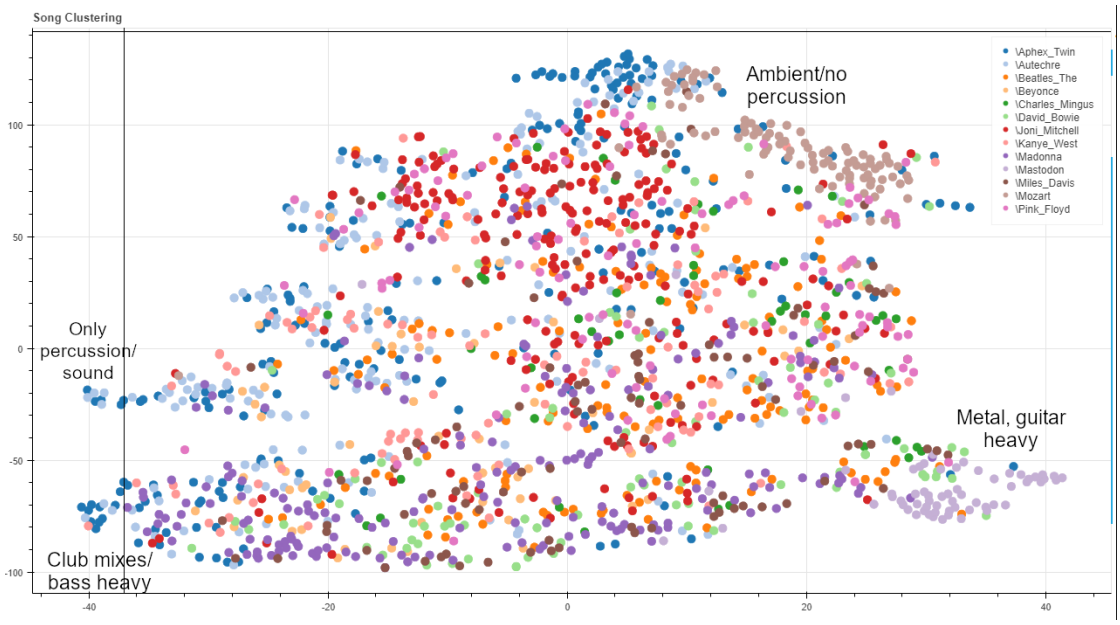
In total, 43 features were extracted and stored in an ordered dictionary for each song, which was then added to a higher dictionary under the name of the artist and stored in a `.pkl` file. From there, the data was trimmed and the t-SNE function of scikit-learn was applied. The results were plotted using the python plot tool bokeh.

```
for artist in all_data:
  data=load_obj(artist.replace('.pkl',''))

  for song in data:
    songfeat.append(data[song])
    songname.append(song.replace('.mp3',''))

  features=[]
  feature_names=[]
  feature_names.append(list(songfeat[0].keys()))
  for i in range(len(songfeat)):
    features.append(list(songfeat[i].values()))

  all_features+=features

reduced_data = TSNE(n_components=2, n_iter=5000, learning_rate=250,
                    perplexity=30).fit_transform(all_features)
```

The iterations, learning rate η, and perplexity were mostly found by starting at $T = 1000$ and $\eta = 250$ and slowly increasing both until satisfying distributions were achieved. This algorithm is highly sensitive to these initialisation parameters, and "How to Use t-SNE Effectively" by Wattenberg et. al. was found to be very useful.

Results and discussion

The following plot was generated, where I have annotated a few perceived qualities of some outlying clusters.



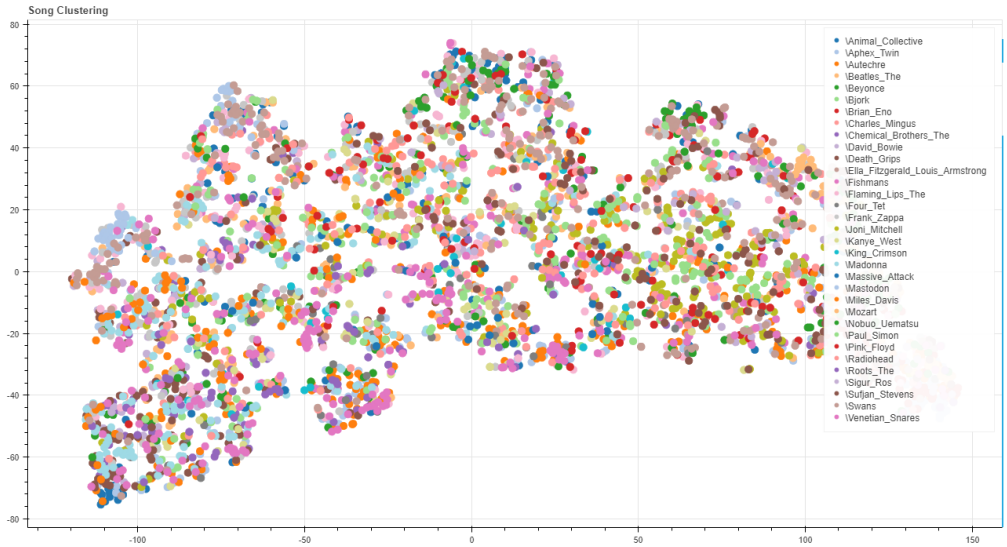
It should be noted that, as t-SNE is reducing 43 dimensions into two, the axes on generated plots mean very little, and numbers are arbitrary. Furthermore, the cost function is not convex, and multiple restarts will end up in local minima and generate a different plot each time.

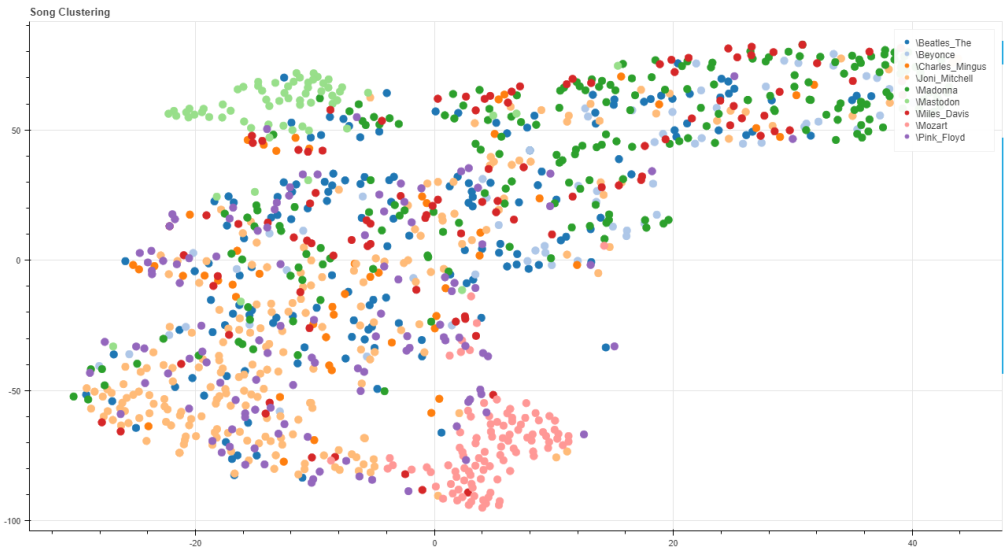
Exploring this plot is quite interesting. Jazz - Charles Mingus and Miles Davis - is broadly dispersed, as is Pink Floyd. Metal (Mastadon) is clustered quite tightly together in the bottom right, and interestingly David Bowie seems to have quite a few songs there as well. Club mixes have migrated over to the bottom left, while more ambient music and tracks with no percussive element are near the top. The electronic artists, Aphex Twin and Autechre, are spread out but tend to hang around the margins, with little or no songs in the bottom right, reflecting that often their music is oriented around a beat (or lack thereof). Ambient and classical are near each other, and a distinct cluster can be seen for piano tracks. The Beatles and Joni Mitchell are spread out across the middle, showing a wide variety but few extremes.

Five frequency bands is a crude split for music data, and better results would almost certainly be obtained with higher resolution. Unfortunately in this case time and processing power restricted the number of bands from which features could be extracted. Furthermore, in many cases (skew, kurtosis) feature values were calculated from the *time average* of each band over the whole song, which may give similar values for songs that do not have obvious differences between verses and the refrain and those that do.

Finally, the clear downside of this technique is its lack of quantitative information: there is no way to tell which features influenced the clustering and how heavily. Nevertheless, the clustering reflects qualities that we may use to describe similar songs, making this an effective visualisation technique.

Bonus plots





Interactive versions of these plots can be viewed at

Acknowledgements

This project was heavily inspired by Alex Clarke (github: informationcake), whose code was helpful. Additionally, I found distill.pub/2016/misread-tsne/ incredibly useful for explaining the parameters of t-SNE. Finally, the original 2008 paper *Visualizing data using t-SNE* by van der Maaten and Hinton helped with the technical explaination.