Self Organizing Maps on Million Song Dataset

Big Data Analytics Fall 2016

Chanakya Kumar, Aidan McConnell, Bhupesh Shetty, Sarah Gerard

Problem Statement

- Use Self-organizing maps to visualize Million song Data set.
- Use cluster and GPU to run experiments

Million Song Dataset

- The MSD contains audio features and metadata for a million contemporary popular music tracks. It contains:
- • 280 GB of data
  • 1,000,000 songs
  • 44,745 unique artists
  • 7, 643 unique terms (Echo Nest tags)
  • 2, 321 unique music brainz tags
  • 43, 943 artists with at least one term
  • 515, 576 dated tracks starting from 1922

Issues with dataset

- Nested directories containing h5 files.
- Each H5 files represents a song.
- A total of million songs i.e. 1 million H5 files.
- Extracting features from H5 files.
- Getting the data in a usable text file format.

Preprocessing H5 files

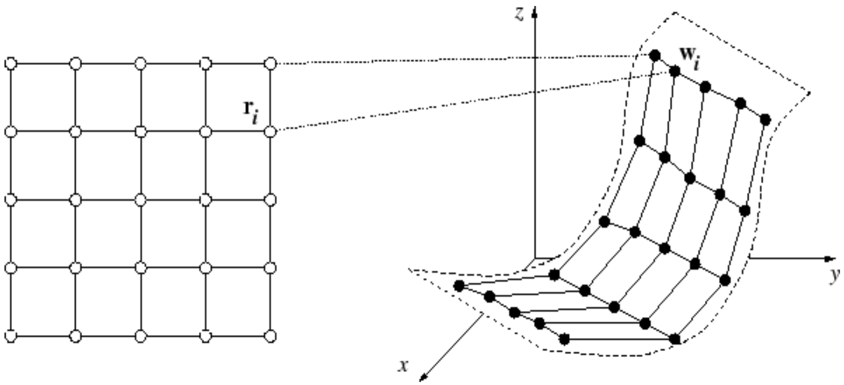
- Extract song features from H5 files.
- 1 D arrays. – Average and norms.
- 2 D arrays. – Covariance, average. (segment timbre, segment pitches).
- Conversion to CSV.
- Finally 220 features.

Features

analysis_sample_rate	artist_7digitalid
artist_familiarity	artist_hotttnesss
artist_id	artist_latitude
artist_location	artist_longitude
artist_mbid	artist_mbtags
artist_mbtags_count	artist_name
artist_playmeid	artist_terms
artist_terms_freq	artist_terms_weight
audio_md5	bars_confidence
bars_start	beats_confidence
beats_start	danceability
duration	end_of_fade_in
energy	key
key_confidence	loudness
mode	mode_confidence
num_songs	release
release_7digitalid	sections_confidence
sections_start	segments_confidence
segments_loudness_max	segments_loudness_max_time
segments_loudness_start	segments_pitches
segments_start	segments_timbre
similar_artists	song_hotttnesss
song_id	start_of_fade_out
tatums_confidence	tatums_start
tempo	time_signature
time_signature_confidence	title
track_7digitalid	track_id
year	

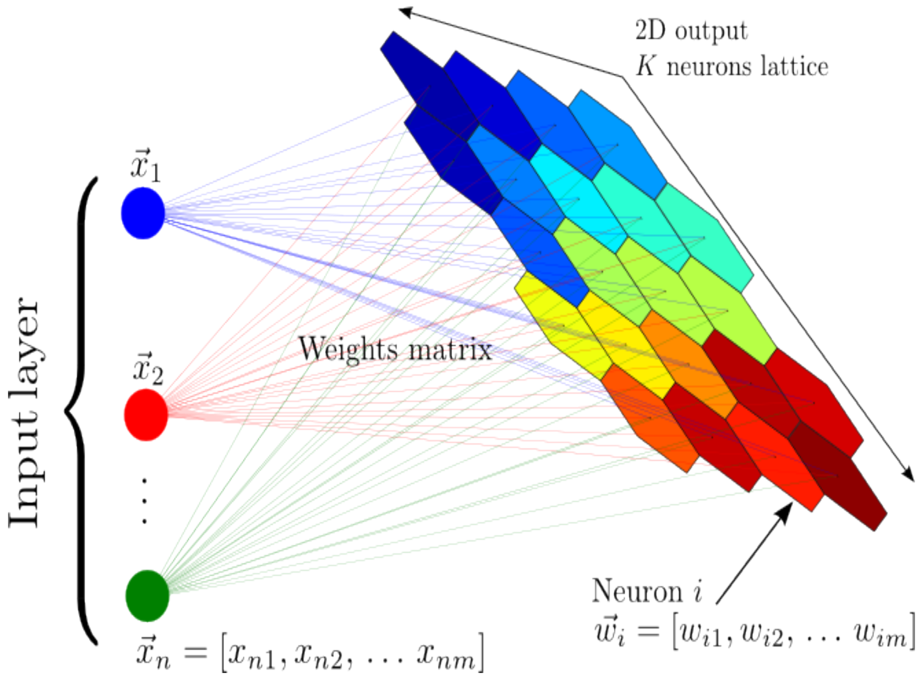
Self Organizing Map

- Useful for visualizing high dimensional data in a low dimensional space
- Uses unsupervised learning to classify data
- SOM is a neural network
- The input data is linked and classified to a node in the network for visualization



Training

- Builds the map using input data
- Each neuron has weights associated with it
- Each data point is matched with a neuron based on the BMU
- Every time this process occurs, the weights are updated



Neighborhood Function

- Neighborhood around BMU starts large
- Neighborhood shrinks iteratively
- Neuron weights adjust to become closer to input data
- The neurons closest to the BMU adjust more than those further away

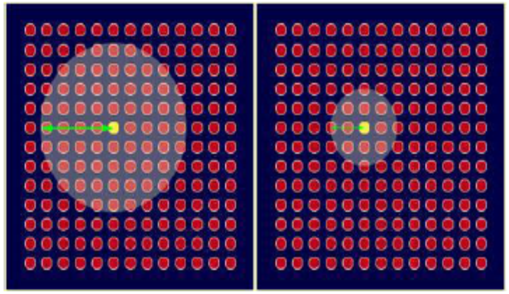
$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right)$$

σ_0 = *the width of lattice at time zero*

t = *the current time step*

λ = *the time constant*

The value of λ depends on σ_0 and the chosen number of iterations for algorithm.

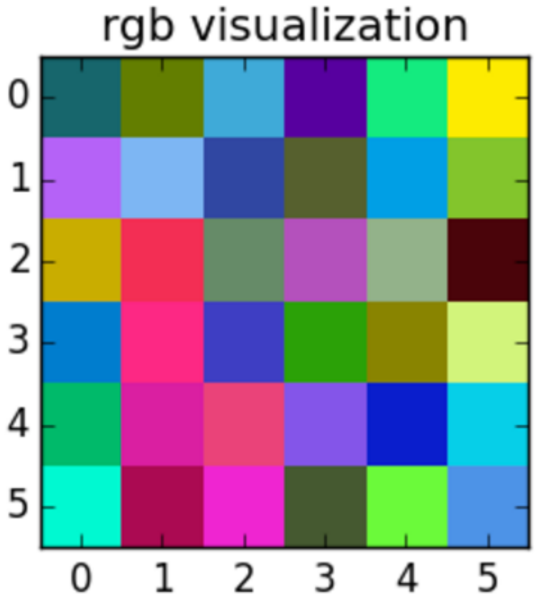


Size of the neighborhood around the BMU shrinks

$$v(t+1) = v(t) + \sigma(t)\alpha(t)(x(t) - v(t))$$

Visualization

- Choose number of neurons wisely
- Visualization of data should be useful and easy to analyze



Approaches

- Waited and cried a little!
- Decrease N/d
- Smart algebra (batch SOM and Gram matrix for distance calculation)
- Used Neon Cluster and GPU (kepler)

Software Platform

Language	C++, Python
Parallel env	OpenMP,MPI,CUDA for GPU
Package	Somoclu

SOM Options

- Naïve implementation (On-line)
- Parallel with MPI
- Parallel with GPU

How is parallel possible (batch!)

Epoch

Data

$$w_j(t_f) = \frac{\sum_{t'=t_0}^{t_f} h_{bj}(t')x(t')}{\sum_{t'=t_0}^{t_f} h_{bj}(t')}$$

Neighborhood function

$$\tilde{d}_k(t) = \|\mathbf{x}(t) - \mathbf{w}_k(t_0)\|^2$$

$$d_c(t) \equiv \min_k \tilde{d}_k(t)$$

BMU

Runtime Comparison

100 epochs and 30*30 MAP
Gaussian Neighborhood
Learning rate 0.1 to 0.01
Full dataset

Methods	Runtime
Naïve (on-line)	53,800 seconds
Parallel without GPU	1,056 seconds
Parallel with GPU	402 seconds

Parameter Estimation

- Extremely difficult!
- Trial and error
- We used quantization error to estimate epochs

$$E_q = \frac{1}{N} \sum_{i=1}^{N} \| x_i - m_c \|$$

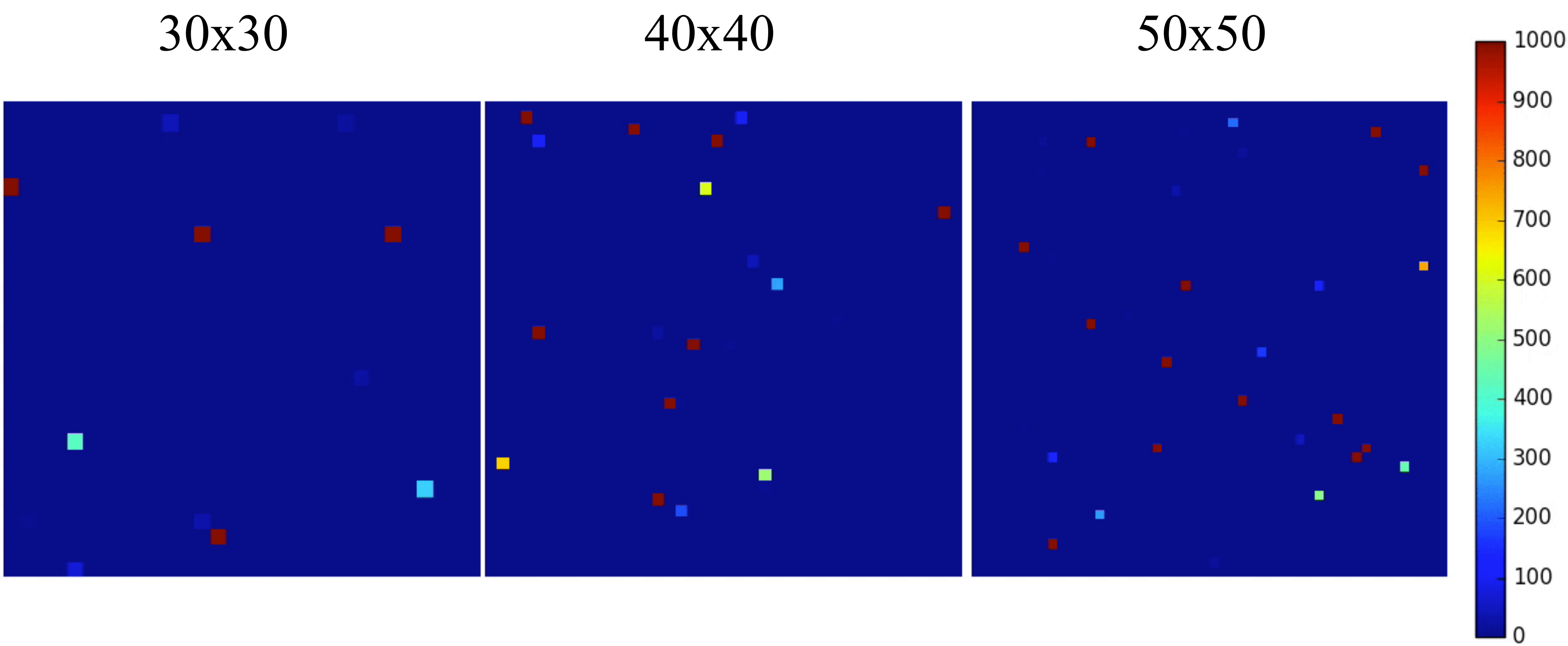
- Lattice dimension and No. of epochs

Experiments

- Selected epochs = 100
- Lattice dimension
  - 25 x 25
  - 30 x 30
  - 40 x 40
  - 50 x 50
  - 75 x 75
  - 100 x 100

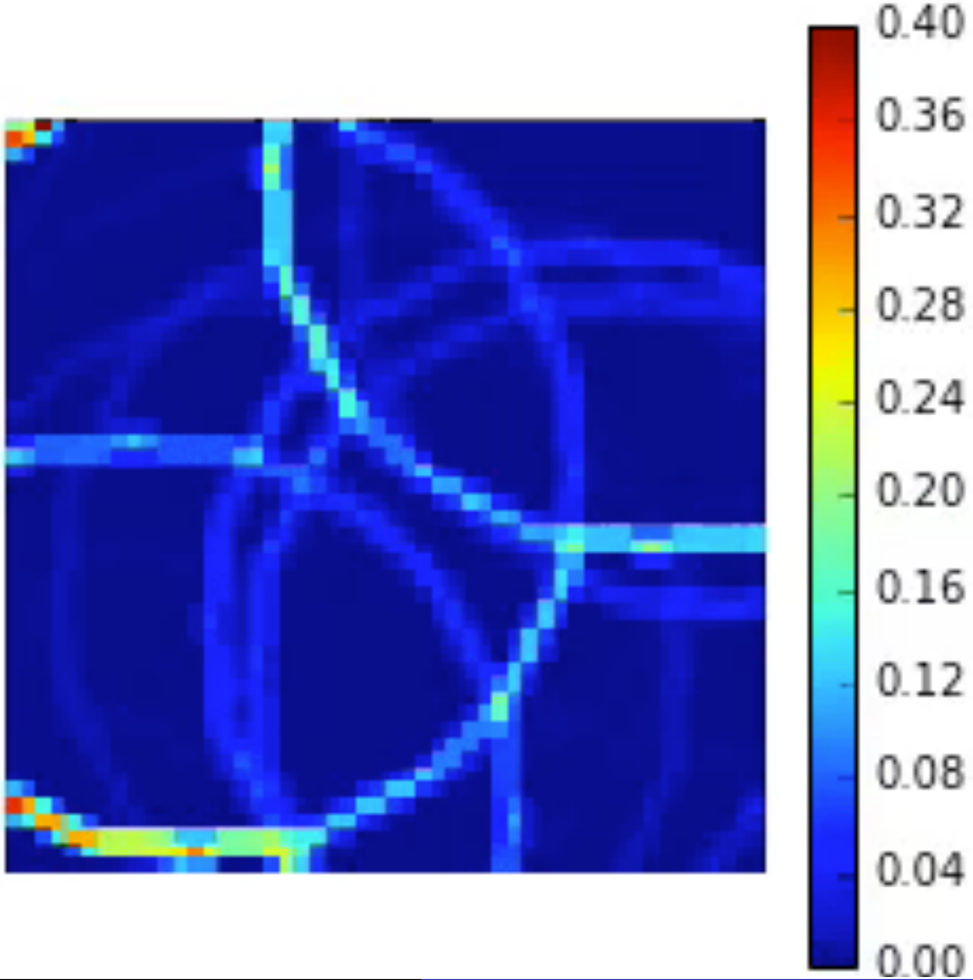
SOM Hitmap

Hitmap: Number of songs with BMU at neuron.

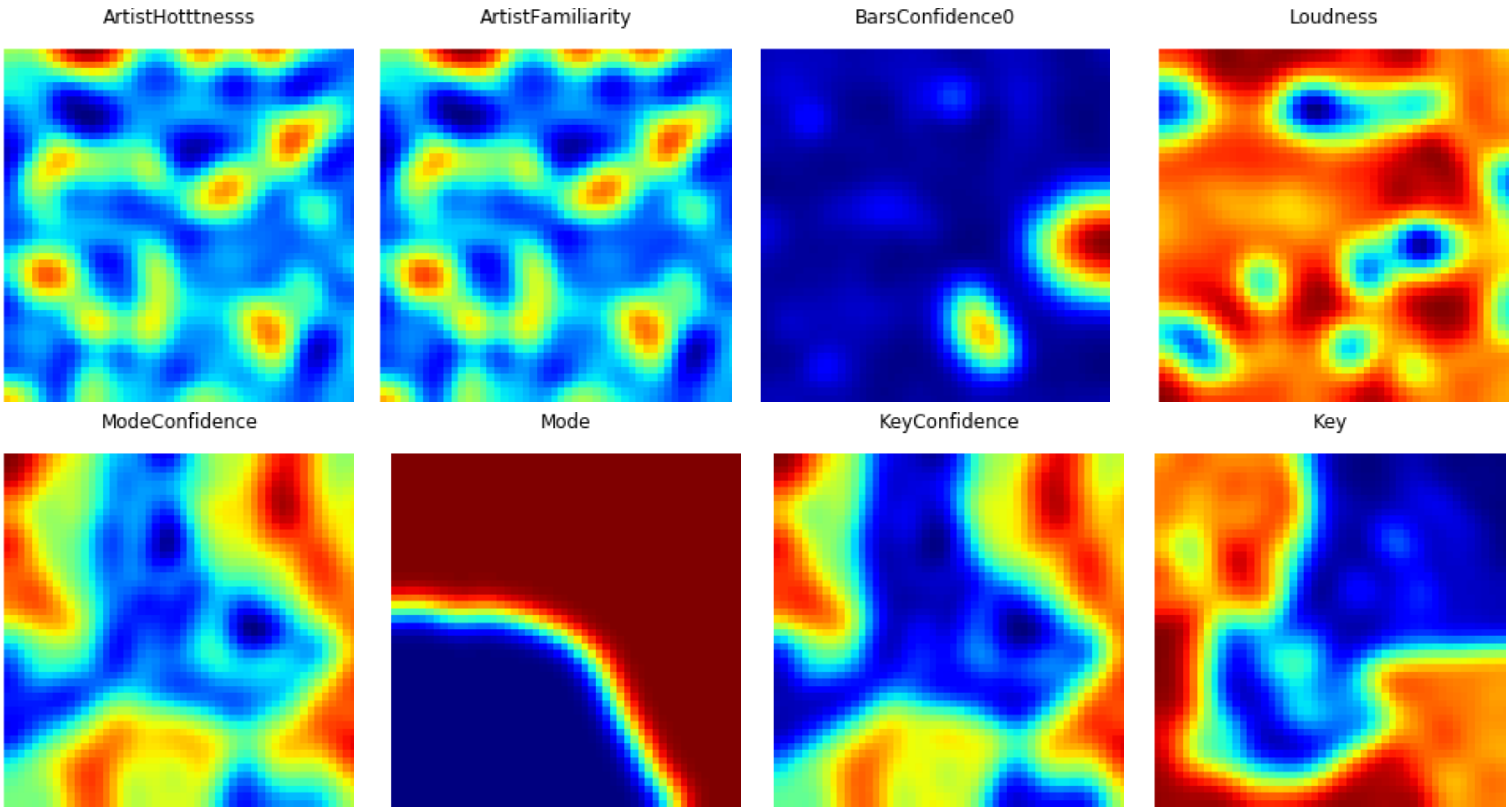


SOM Unified Distance Matrix

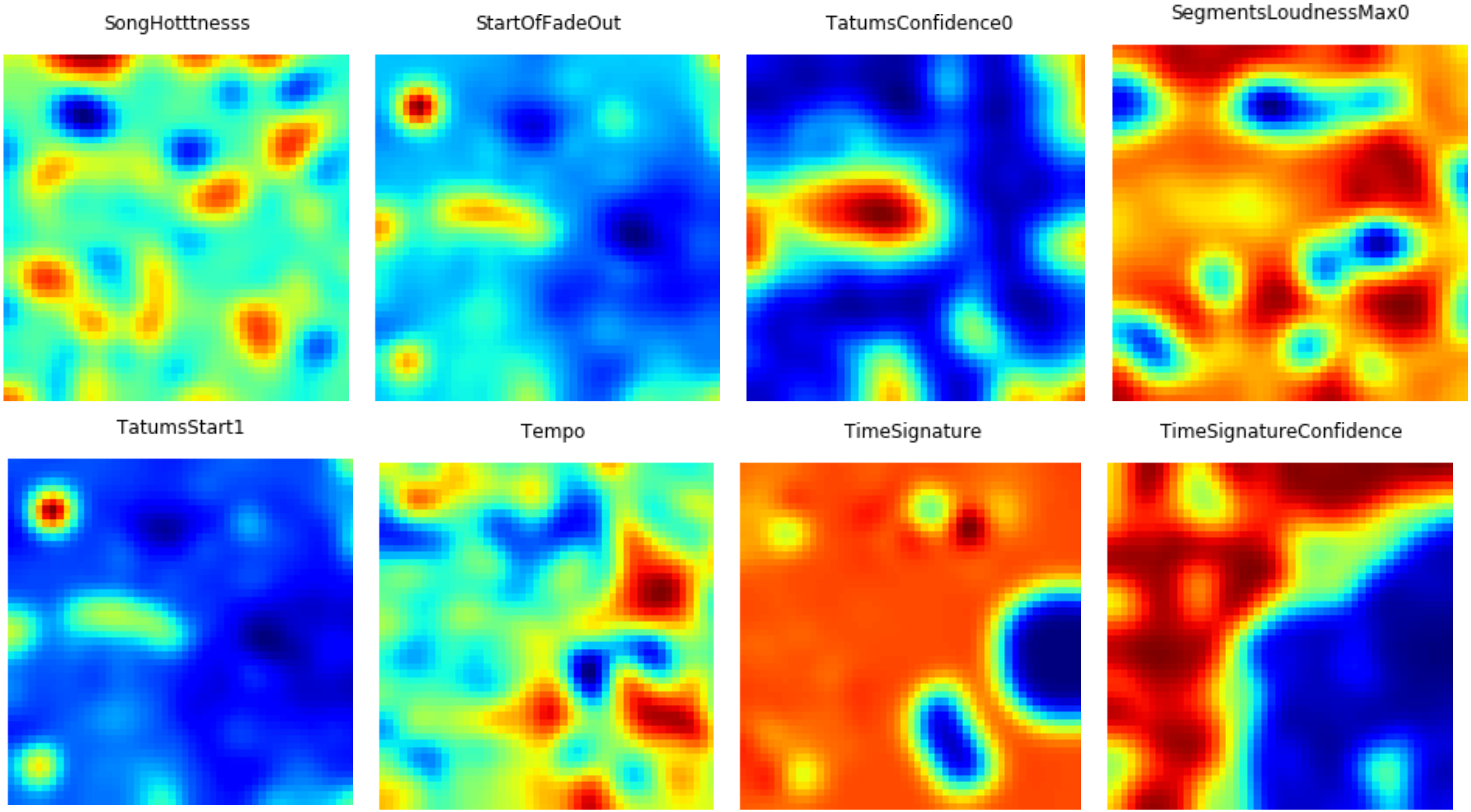
U-Matrix: Average distance from a neuron to it's neighbors.



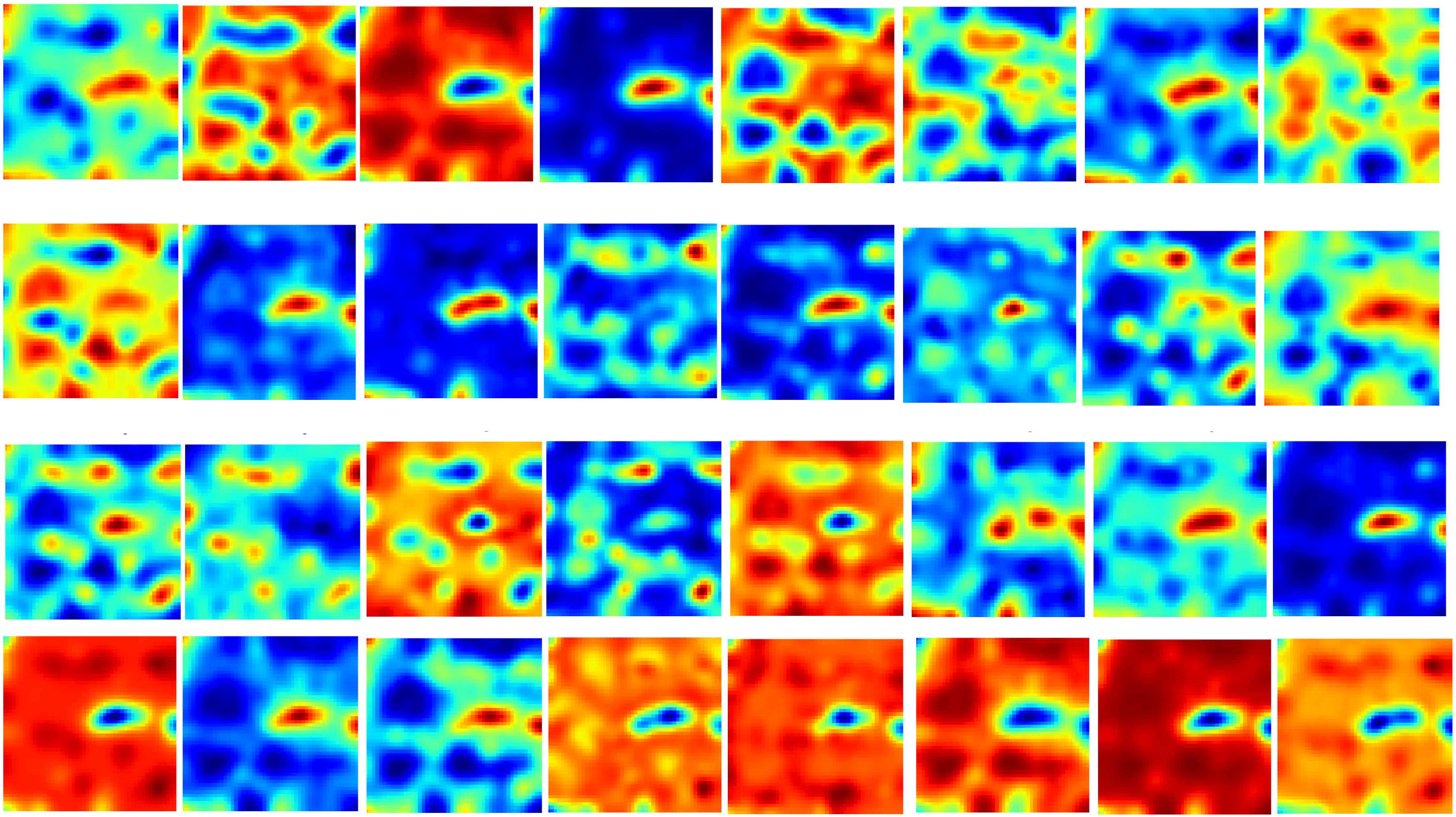
SOM Component Planes



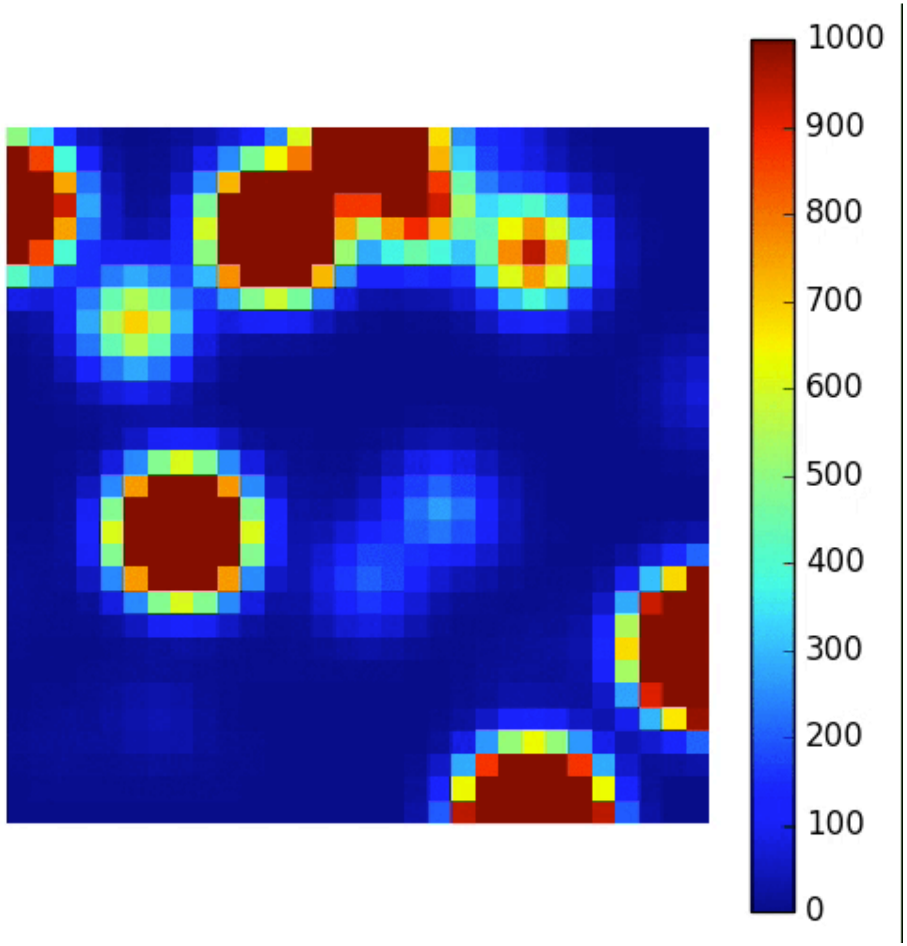
SOM Component Planes

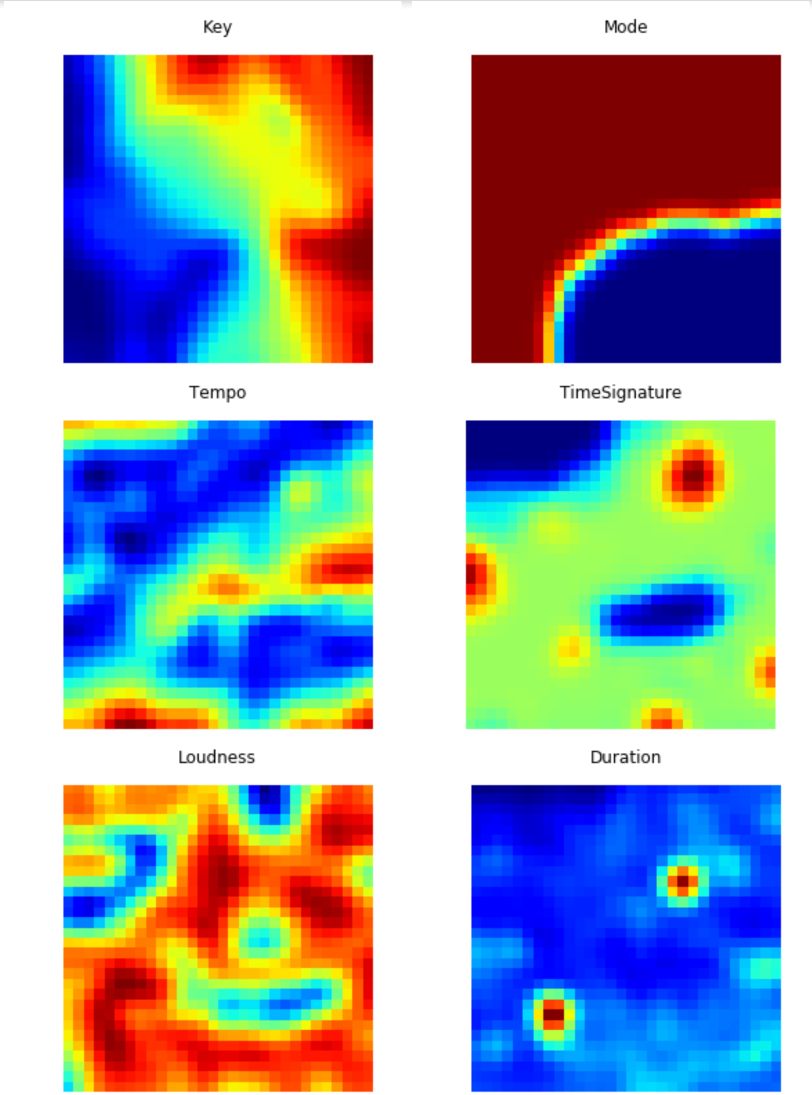


SOM Component Planes

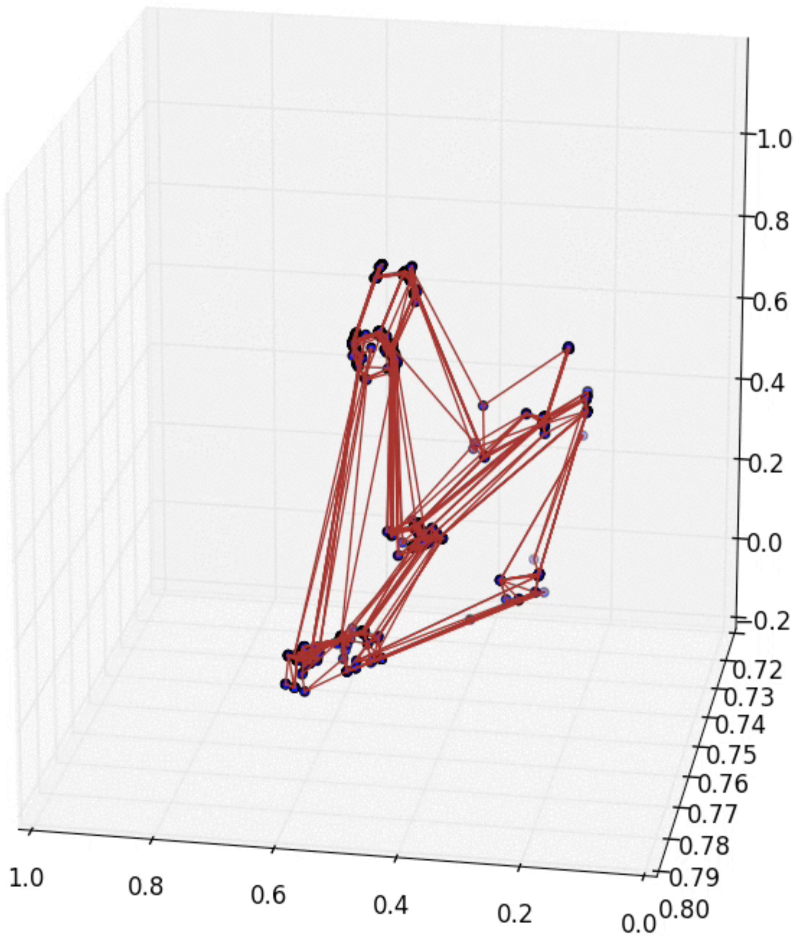


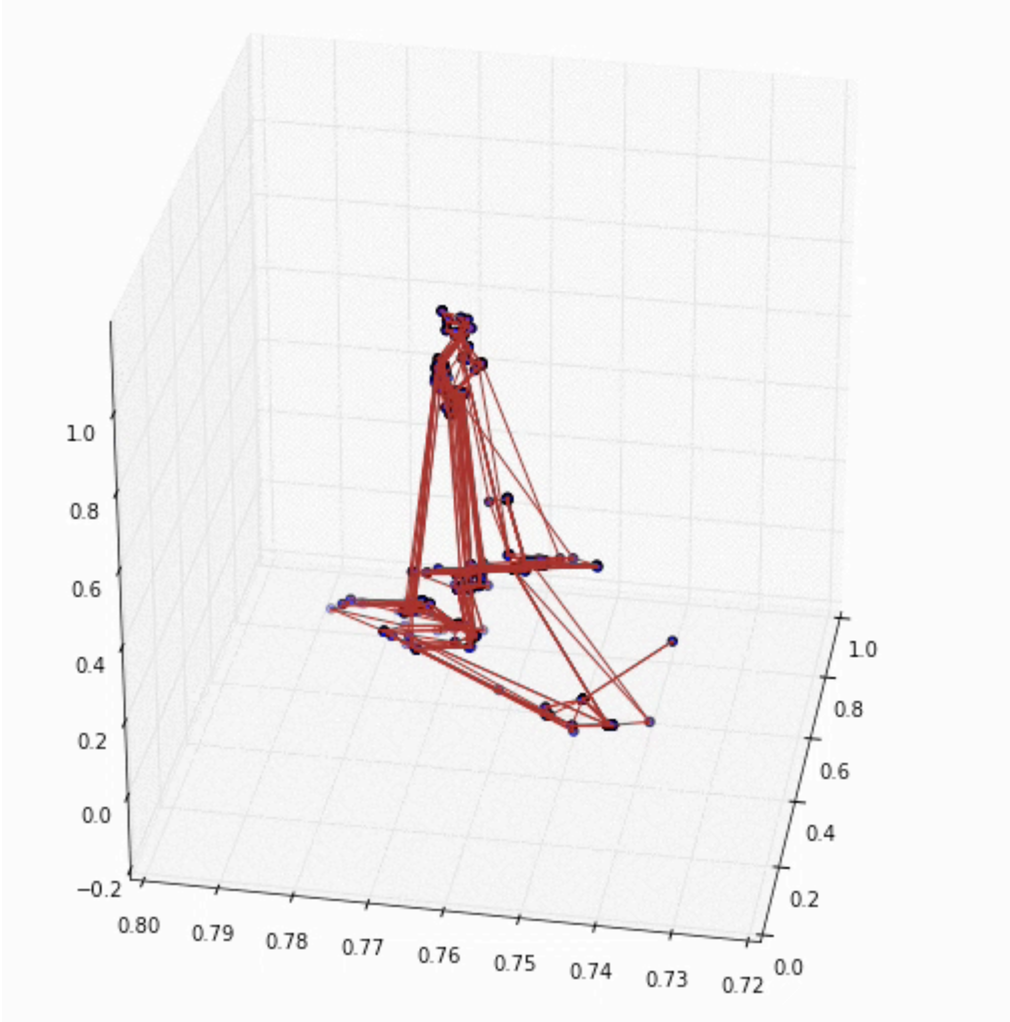
Genre Subset



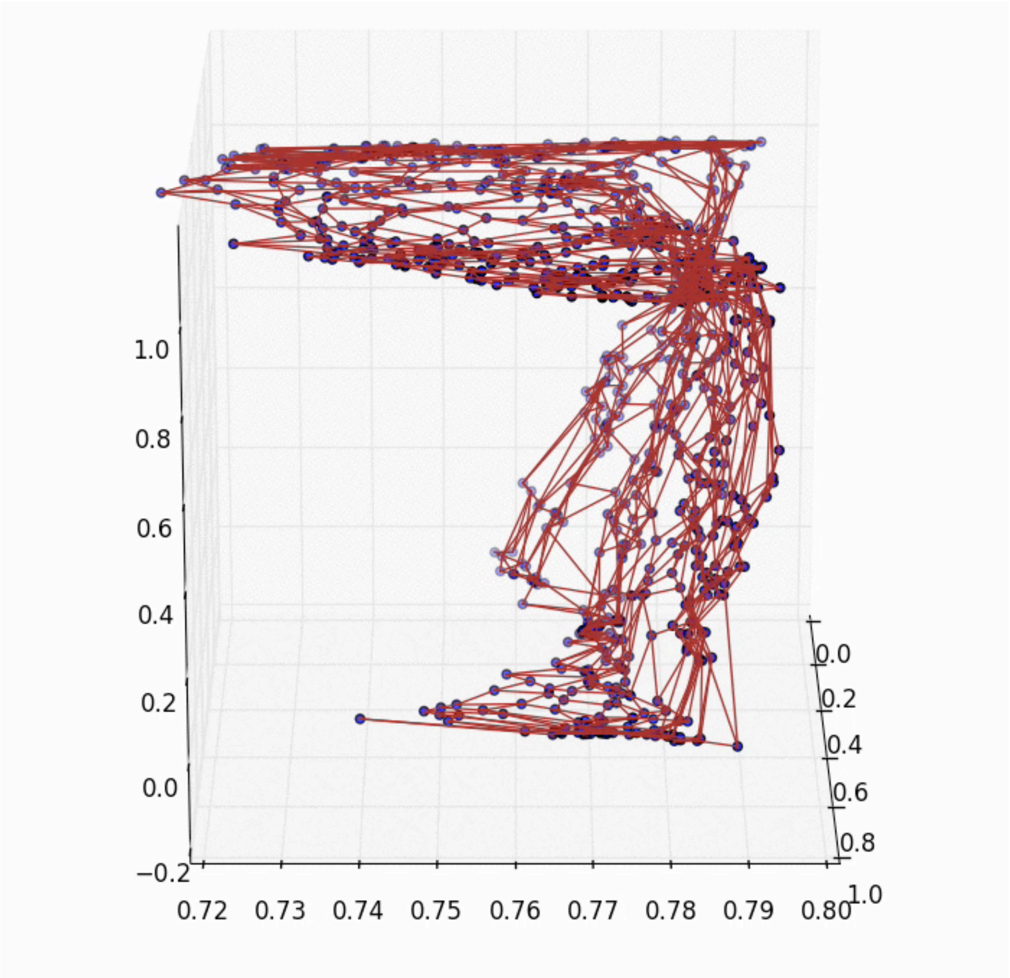


SOM Unfolding





SOM Unfolding

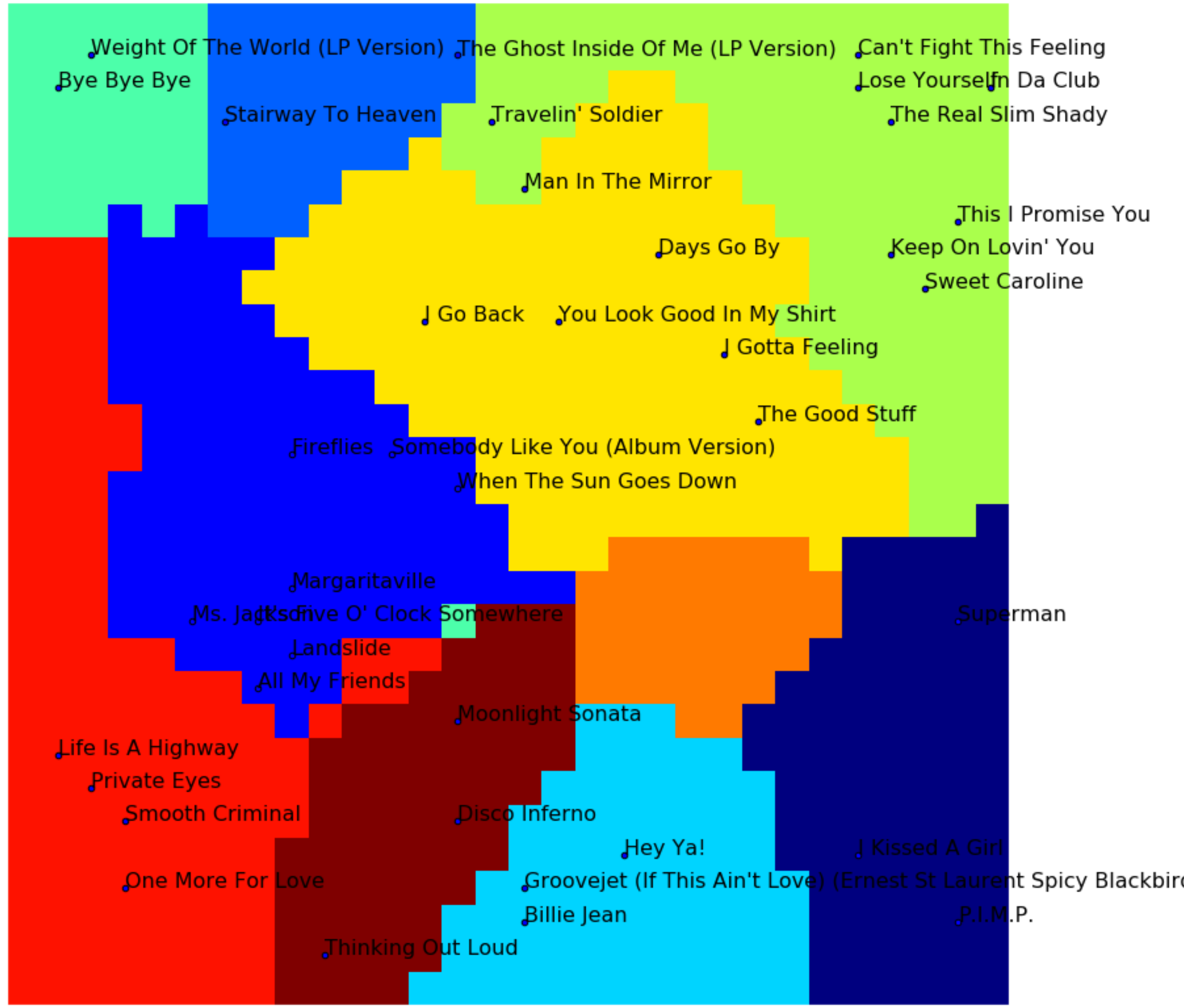


K-Means Clustering

The University of Iowa

Clustered SOM neurons in the data space using final weights as features.

Faithful Projection?



Limitations and Opportunities

- More work on feature selection. Consult a music expert!
- Parameter estimation. We only considered square dimension.
- GPU memory issues.
- Implement another SOM method (Dot-product SOM?)
- Start very very early!!!!

Conclusions

- Implemented both on-line and parallel version of SOM on Million Song Dataset
- Compared the performance in terms of run time.
- GPU is lightning quick but had memory issues when increasing size of lattice.
- Developed visualization using SOM for million songs!