Project Group 11 : Music Mood Classifier

Jorge Alvarez, Saaketh Medepalli, Jack Smith

Univeristy of Michigan

Winter 2021

Presentation Outline

1. Motivation and project goal
2. Dataset and Preprocessing
3. Featurization
    - Spectral Centroid
    - Spectral Bandwidth
    - MFCC
    - Chroma
4. Classification
    - KNN
    - SVM
    - Neural Network
5. Future Improvements
6. Demo and Questions

Motivation and Project Aim

- Music is an inherently emotional experience. We want to see if certain audio features can help machine learning techniques learn the differences between songs that communicate different emotions.
- **Multiclass Classification**-classify a song into one of four moods:
  - Happy
  - Sad
  - Calm
  - Hype
- Use different classification algorithms and compare the results:
  - Neural Network
  - K-Nearest Neighbors
  - Support Vector Machine (SVM)

Dataset and Featurization

- Dataset:
  - Created ourselves by downloading popular Spotify playlists categorized by mood.
  - Total of 729 songs which we then split up into train and test sets which contained 681 and 48 songs, respectively.
  - **Happy:** 220 **Sad:** 159 **Calm:** 169 **Hype:** 181
  - Needed each song to have the same number of samples, so we took the middle N samples of each song (differs based on classification method)
- We wanted to implement the audio feature extraction functions ourselves, the features most helpful for mood classification are:
  1. Spectral Centroid
  2. Spectral Bandwidth
  3. Mel-Frequency Cepstrum Coefficients (MFCC)
  4. Chromagram

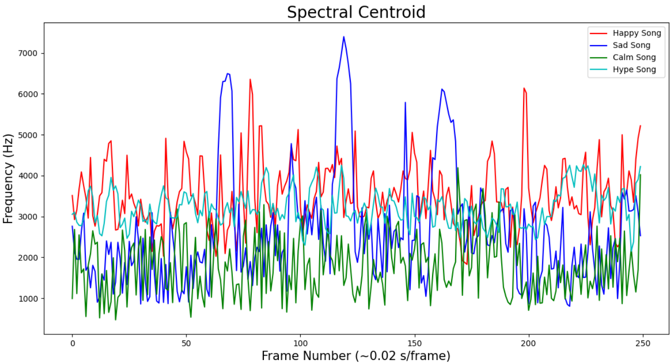
Spectral Centroid

Spectral Centroid is the weighted mean of the frequencies present in the signal.

- Calculation:
  1. Loop though all frames of the song
  2. Extract the magnitudes of the frequencies
  3. Extract the positive frequencies
  4. Calculate the spectral centroid over one frame by summing the product of all the magnitudes and frequency components divided by the sum of the magnitude components

$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)S(n)}{\sum_{n=0}^{N-1} S(n)}$$

Spectral Centroid Plot for Each Mood



- The hype and happy song in general had higher brightness as compared to the calm or sad song.

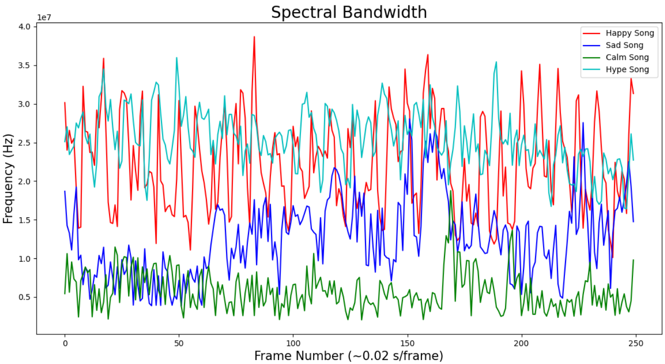
Spectral Bandwidth

Spectral bandwidth is a weighted standard deviation

- Calculation:
  1. Loop though all frames of the song
  2. Extract the magnitudes of the frequencies
  3. Extract the positive frequencies
  4. Calculate the spectral bandwidth over one frame by summing over each frequency and following the steps below:
     1. Subtract the frequency component with its corresponding spectral centroid frequency
     2. Square (1)
     3. Multiply the result from (2) with the frequency magnitudes
  5. Take the square root of the resulting vector

$$(\sum_k S(k)(f(k) - f_c)^2)^{1/2}$$

Spectral Bandwidth Plots for Each Mood



- The spectral bandwidth was higher for the happy and hype songs because the standard deviation is higher.

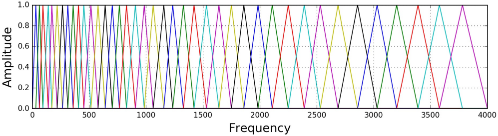
MFCC

Mel-Frequency Cepstral Coefficients (MFCC) perform filtering based on the sound perception by human hearing.

- Calculation:
  1. Include a pre-emphasis factor to amplify the high-frequency magnitudes
  2. Split the signal x into short time-frames with 500 sample overlap
  3. Apply a hamming window to each frame to deemphasize overlap
  4. Calculate FFT of each frame i and the power spectrum using:

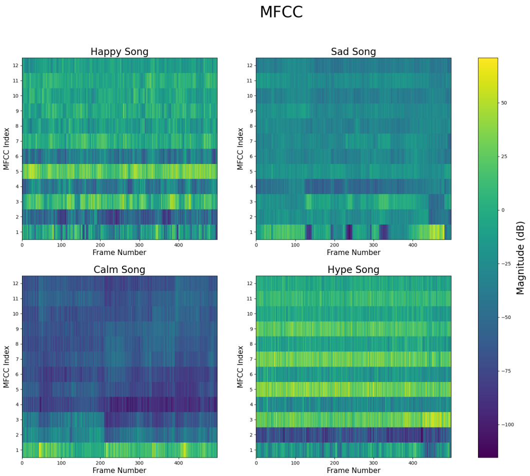
$$P = \frac{|FFT(x_i)|^2}{N}$$

  5. Apply triangular filters on Mel-Scale to adjust the perception of frequencies



  6. Apply DCT to Spectrogram to remove correlation between coefficients

MFCC Plots for Each Mood



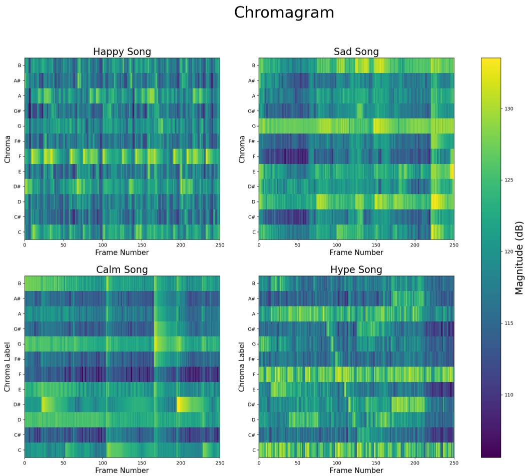
Chromagram

Similar to a spectrogram except the y-axis represents the 12 chroma labels rather than frequency.

- Calculation:
  1. Compute STFT of the signal with a frame size of 1000 samples (0.02 seconds).
  2. Loop through all possible notes (C1 to G9), and compute the upper and lower frequency bounds for each note.
  3. Loop through the STFT's calculated and compute total magnitude for each note in each frame.
  4. Loop through the notes and bin them into the corresponding chroma.

We are left with a matrix: $N_{frames}x12$ where 12 is the number of distinct chromas.

Chromagram Examples for Each Mood



K Nearest Neighbors (KNN)

- Calculation for each feature:
  1. Loop though all the songs, separate them between training and test data, and label them correctly
  2. Calculate the difference between the train and test data
  3. Take the norm of the difference vectors
  4. Find the indices for the K smallest normalized difference vectors
  5. Store the corresponding training labels associated with the smallest K indices
  6. Take the mode of the labels and compare it to the test label to determine mood classification

KNN - Individual Results

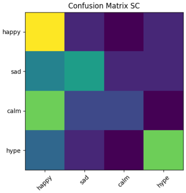


Figure: Spectral Centroid

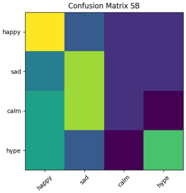


Figure: Spectral Bandwidth

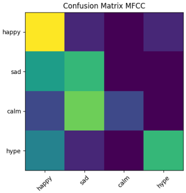


Figure: MFCC

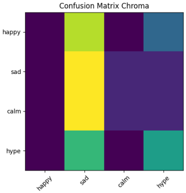


Figure: Chroma

KNN - Final Results

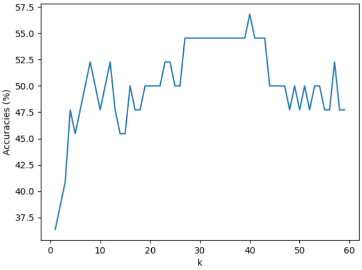


Figure: Accuracy for K neighbors

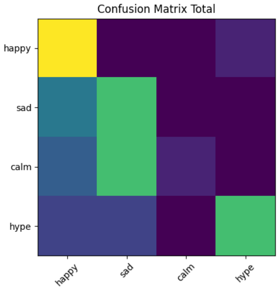


Figure: Total

- The optimal K was 40 with an accuracy of 57%

Support Vector Machine (SVM)

From class, the goal of a SVM is to find the hyperplane that maximizes the separation between points of different classes. For a multiclass classification problem, Python includes a *svm* library with a classifier called *SVC*:

- Procedure:
  1. For each feature, *svm.SVC* was called with equally sectioned classes on the training data for each feature
  2. Labels for each feature were predicted using the trained weights
  3. Mode of the label prediction for each feature was used to develop the final prediction
- Replacing the dot product with a more general 'kernel' operation allows for a transformed feature space - easier to nonlinearly separate the data
  E.g. of Kernel: Linear, Polynomial, Gaussian Radial Basis Function (RBF)

SVM - Individual Results

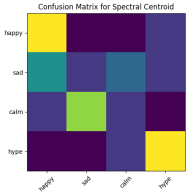


Figure: Spectral Centroid

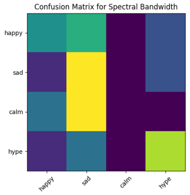


Figure: Spectral Bandwidth

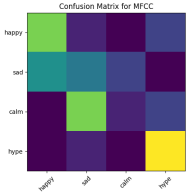


Figure: MFCC

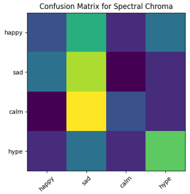


Figure: Chroma

SVM - Final Results

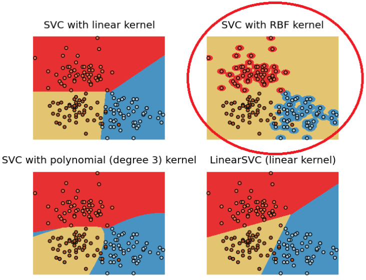


Figure: Kernels for SVM

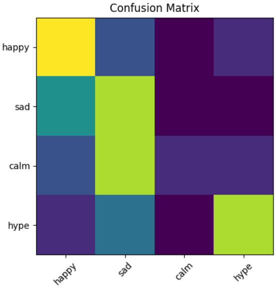
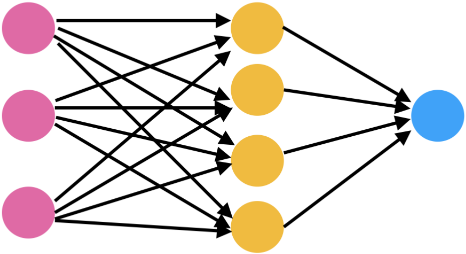


Figure: Total

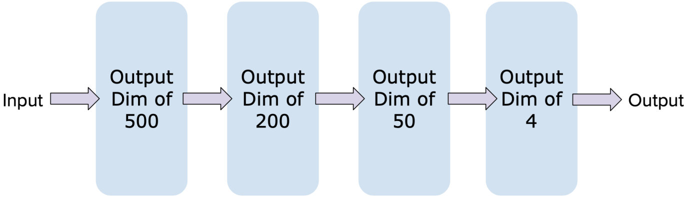
- A 53% accuracy was produced using the RBF Kernel SVC

Introduction to Neural Networks



- Neural networks are comprised of layers.
- Each layer multiplies the input to that layer by a matrix of weights, and then adds a bias, ie:
  - $\vec{y} = \mathbf{W}\vec{x} + \vec{b}$
- The parameters of $\mathbf{W}$ and $\vec{b}$ are updated after the loss is computed.

Our Network Architecture



- Our input vector was the featurized song.
- The output vector is dimension 4 which will allow us to compute probabilities for each mood.
- Each layer of the network except for the final one is followed by a ReLU.

$$relu(x) = max\{0, x\}$$

Training Details

- Used PyTorch for data loading and training.
- Train and test sets each had their own set of labels, which was a list of numbers 0-3. (0=Happy, 1=Sad, 2=Calm, 3=Hype)
- Initially, our input vectors were songs that were entirely featurized.
- Batch Size = 40, Learning Rate = 0.0001, Number of Epochs = 5
- Used cross entropy loss, which takes in raw logits, computes relative probabilities for each class, and compares to the actual label.

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i log(\hat{y}_i)$$

Initial Neural Network Results

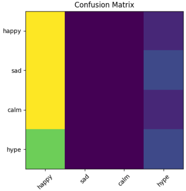


Figure: Spectral Centroid

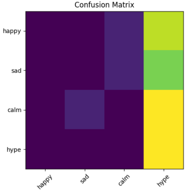


Figure: Spectral Bandwidth

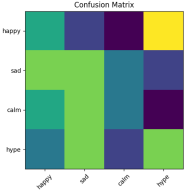


Figure: Chroma

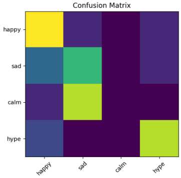


Figure: MFCC

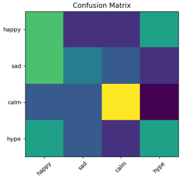
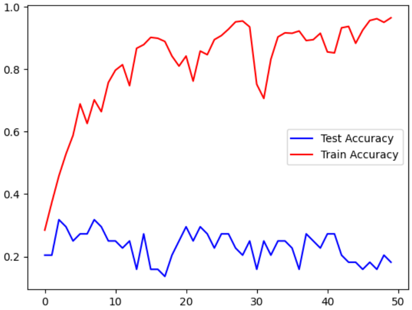


Figure: Total

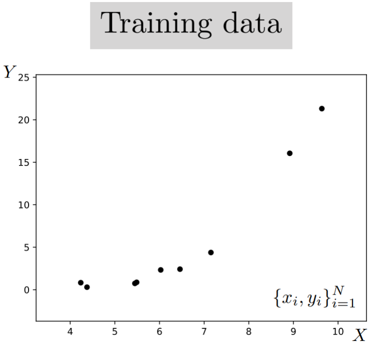
Initial Neural Network Results

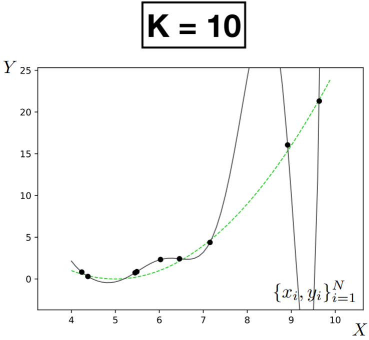
- It was evident that some of the features weren't working as well.
    - Spectral Centroid
    - Spectral Bandwidth
    - Chroma
- Initial training of the network yielded poor results for test data.



Overfitting

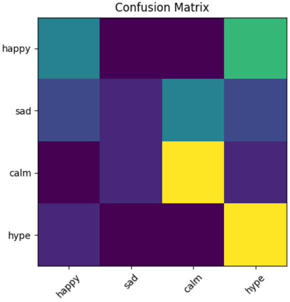
- We suspected that our model was overfitting the training data.

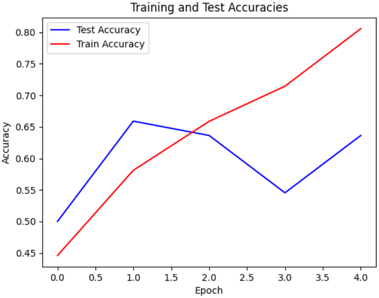




Neural Network Final Results

- We learned from our confusion matrices that MFCC seemed to be the feature the network learned best.
- After training just on the MFCC, we achieved much better results (60% accuracy).





Future Improvements

- Only the middle N samples were taken from each song, which might have not been as representative of the mood. As an improvement, we could take N samples from a random point in the song.
- For each of the features, use techniques such as mean normalization to reduce SNR and highlight important characteristics of mood.
- Try using more novel techniques such as LSTM RNNs on the pure audio signal and see how they compare.

Demo and Questions

Demo:

$\boxed{\text{Song 1}}$

$\boxed{\text{Song 2}}$

Questions?