Audio Analysis and Music Genre Classification using Machine Learning

CSC 59929 - Introduction to Machine Learning

May 21, 2020

Abtahi Chowdhury

Professor Erik Grimmelmann

1. Introduction

With the current growth of online music streaming services like Spotify and Google Play Music, being able to analyze and classify music into genres can greatly affect both the company's user base and the user's listening experience. However classifying music into genres is a very complicated task due to the nature of how analyzing audio works as well as the plethora of different categories that music can be classified into.

Music genre classification falls under the area of Music Information Retrieval (MIR), which as the name implies, is the science of retrieving information from music. This area of research has many applications, some examples being automated music recommendation systems, music recognition systems, music generation, and music classification. However this doesn't just apply to the realm of music. Much of the research in this field is also used in the field of speech recognition.

2. How is Audio Represented?

In order to understand how to analyze audio, a good understanding of how audio is represented is needed. This involves both the physical and digital representations, both of which will be covered in this section.

In physics, sound is the vibration of a transmission medium as a wave propagates through it. Using the examp le of sound moving through air, it is the periodic compression and expansion of the molecules in the air. The key point in this being that it is periodic, meaning it

repeats some pattern of movement over time. This sinusoidal movement means that it can be represented analytically with a sine function or even the summation of many sine functions.

Sinusoidal functions have 2 main attributes to keep track of when analyzing: amplitude and frequency. The amplitude represents the "loudness" of the sound measured in decibels, and the frequency represents the pitch of the sound measured in hertz. Figure 1 shows a simple sine function of amplitude 1 and frequency $1/2\pi$.

However, a computer will need to store audio digitally. This is done using analog digital conversion (ADC). The ADC process involves sampling an audio signal at fixed time intervals and quantising these samples into bits. The rate at which these samples are taken is called the sample rate measured in hertz with the actual data being sampled being the amplitude of the function at that given time.

Real-world audio is many times more complex than the simple sine waves shown so far. Figure 2 is an example of real-world audio data represented in a waveform. In the next section the actual analysis process will be discussed in detail.

3. Analyzing Audio

3.1. Fourier Transform

In order to analyze audio, a method to break down complex sinusoidal functions is needed. This method is known as the fourier transform. It is used to decompose a complex periodic wave into the sum of sine waves of the form $A\sin(2\pi ft + \varphi)$ where A is the amplitude, f is the frequency and φ is the phase shift. An example of this can be seen in figure 3 where a

complex wave is decomposed into $0.5sin(4 * 2\pi t) + 1.5sin(1.5 * 2\pi t)$. However, the fourier transform is used on analog waves and not on digital waves. For that we will need the fast fourier transform.

3.2. Fast Fourier Transform and Short Time Fourier Transform

The fast fourier transform (FFT) is an algorithm that is used on digital wave functions to convert them from functions in the time domain into functions in the frequency domain in O(nlogn) time. This new graph that is created is called the spectrum graph and can be seen in figure 4. The one downside to using the spectrum is that it carries no time information. To fix this, the short time fourier transform (STFT) is used instead to make a spectrogram. The STFT computes the FFT several times at different time intervals, thus preserving time information. This can be seen in figure 5 where the x-axis represents the time, the y-axis represents the frequency, and the color represents the magnitude.

4. History of Music Classification

4.1. Traditional Machine Learning Preprocessing Pipeline

Traditionally, the preprocessing step for any audio data leaned more towards feature engineering. This means things like the waveform and spectrogram were used to extract different features. A few examples of these features are amplitude envelope, 0-crossing rate, spectral flux, and spectral centroid. These features would then be used in different classification machine learning algorithms as training data. However, with the advent of deep learning and neural

networks, it became more efficient to use mel-frequency cepstral coefficients as the goto audio data feature.

4.2. Modern Machine Learning Preprocessing Pipeline

Mel-frequency cepstral coefficients (MFCC) are compact representations of the amplitude spectrum of audio data. They are designed to mimic the human auditory system by modeling the timbre (a.k.a. the perceived sound quality) of audio data and have performed very well when compared to other short-time features. Due to this, they have been used extensively in speech recognition alongside music classification and therefore, a more careful description of MFCCs will be given.

In order to compute the MFCCs of a piece of audio data, the following steps are performed at each interval of the STFT:

1. Perform the fft on the audio data at that frame
2. The output spectrum is then run through a mel-scaling function to convert the frequencies into a perceptual pitch scale.
3. The output of the mel-spectral spectrum is then run through a log-scaling function to smooth out the amplitude.
4. Finally a discrete cosine transformation (DCT) is performed on the mel-spectral log-scaled coefficients.
5. Keep a certain amount of these resultant coefficients and discard the rest. Generally for music analysis around 13-40 coefficients are kept.

Figure 6 shows an example of a MFCC plot and figure 7 shows a spectrogram made using the mel-spectral spectrum. The coefficients obtained in the MFCC computation process are then fed into a neural network as training data.

5. Overview of the Data and Models

5.1. GTZAN Audio Dataset

The GTZAN audio dataset is a collection of 30 second audio clips classified into 10 different genres. The 10 genres are blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock, with each genre containing 100 tracks in wav format. In order to work with the dataset, it needs to be processed first. This will be done using the python library librosa. Using librosa, the MFCCs can easily be extracted from the audio files and stored into a json file to be used in machine learning models.

5.2. Models Used

The GTZAN audio dataset will be used to train 5 different models. Those models being k-nearest neighbors, support vector machine, neural network, convolutional neural network, and recurrent neural network. Each model was trained using the MFCCs of 70% of the samples with the remaining 30% being used for validation.

6. Training the Models

6.1. K-Nearest Neighbors

The first model used to classify the dataset is the k-nearest neighbors classifiers from the scikit-learn library. For this model the weights and the number of neighbors are varied. The number of neighbors ranged from 1 to 10 and the weights used were uniform and distance. After training the model, the validation samples are used to test it with the accuracy of the model being recorded at each interval. At the end of all the iterations, a confusion matrix is made using the model with the best accuracy.

Looking at figure 8, it can be seen that the accuracy of the model decreases as the number of neighbors increases regardless of the weights used. Both uniform and distance weights decrease at the same rate, however when using distance weights the fall off starts at 3 neighbors as opposed to uniform weights that fall off at 2 neighbors. Overall both had a max accuracy of around 0.55 at 1 neighbor. Looking at the confusion matrix in figure 9, it can be seen that the model was able to classify classical and metal music very well with an accuracy of 94% and 90% respectively. This can possibly be correlated to the two genres being very distinct from the other genres with classical usually softer tones and metal having harsher tones. The model also struggles with reggae and hip hop with accuracy of 26% and 19% respectively.

6.2. Support Vector Machine

The second model used to classify the dataset is the support vector machine from also from the scikit-learn library. For this model the kernel and the regularization factor are varied. The regularization factor ranged from 10 to 100 with a step size of 10 and the kernels used were linear, poly, sigmoid, and rbf. Much like the knn models, the accuracies of the test samples were recorded and plotted and a confusion matrix was made for each kernel using the model with the best accuracy.

Looking at figure 10, it can be seen that there is a very distinct separation in accuracy based on the kernel used. In fact, for the most part, the regularization factor did not play a big part in the accuracy of the model. The accuracy of the poly kernel went up as the regularization factor went up, however the rbf kernel still maintained a higher accuracy even though it's accuracy went down as the regularization factor went up. Overall the rbf kernel had the best accuracy of 0.64 at a regularization factor of 20. Looking at the confusion matrix in figure 11, it can be seen that there is an overall improvement in the accuracy from the knn model. This is most notable in hip hop and reggae as both saw an over 200% increase in accuracy from the knn model. However classical, rock and metal all saw a slight decrease in accuracy from the knn model.

6.3. Neural Network

The third model used to classify the dataset is a normal neural network made using the tensorflow library. The architecture of the network consists of one flatten layer followed by 3

hidden dense layers and the output layer. The hidden layers use the relu activation function and have a size of 512, 256, 64 neurons respectively. They also utilize L2 regularization and dropout to combat overfitting. The output layer uses the softmax activation function and has a size of 10 neurons, one for each possible genre. The model was compiled with the Adam optimizer with a learning rate of 0.0001 and the sparse categorical crossentropy loss function. The model was then trained for 100 epochs and the test and validation accuracy and error per epoch was plotted alongside a confusion matrix after the training was completed. The reason 100 epochs was chosen was because the model will start overfitting between 70 and 100 epochs.

Looking at figure 12, it can be seen that around 65 to 70 epochs into the training is when the model starts to show signs of overfitting. However, due to the randomness of the dropout layers this can vary but will usually stay less than 100 epochs. The actual accuracy measure levels off around 58%-60%, making it slightly worse than the rbf-svm. Looking at the confusion matrix in figure 13, it can be seen that it is similar to the confusion matrix for the rbf-svm, however with slightly worse scores along the diagonal, which makes sense due to the accuracy being lower than the rbf-svm.

6.4. Convolutional Neural Network

The fourth model used to classify the dataset is a convolutional neural network (cnn) made using the tensorflow library. The architecture of the network is 3 hidden convolutional layers, 1 hidden flatten layer, 1 hidden dense layer, and the output layer. Each of the 3 convolutional layers are the same with each consisting of a convolutional layer, a max pooling layer, and a batch normalization layer. The convolutional layers have 32 filters, a 2-by-2 kernel,

and use the relu activation function. The max pooling layers have a 2-by-2 pool size and 2-by-2 stride length. The dense layer at the end has a size of 64, uses the relu activation function and has a dropout layer with a probability of 30%. Finally the output layer uses the softmax activation function and has a size of 10 neurons, one for each possible genre. The model was compiled with the Adam optimizer with a learning rate of 0.0001 and the sparse categorical crossentropy loss function. The model was then trained for 30 epochs and the test and validation accuracy and error per epoch was plotted alongside a confusion matrix after the training was completed. The model was trained for 30 epochs because the model will start to overfit at around 30 epochs.

Looking at figure 14, it can be seen that the accuracy of both the training and test data increase very rapidly and start to level off around 30 epochs. Due to the dropout layer, the model might take more or less epochs to level off, however this is usually in the range of 1-2 epochs, making this model very stable. The accuracy itself after 30 epochs levels off around 70% making this the model the best one out of the ones tested so far. Looking at the confusion matrix in figure 16, this is seen when looking at the diagonal of the matrix. Overall this has much better results than the rbf-svm or regular neural network. It can be seen that the lowest score is a 51% on country music, whereas the rbf-svm had a lowest score of 38% on rock music and the regular neural network had a lowest score of 34% also on rock music.

6.5. Recurrent Neural Network

The final model used to classify the dataset is a long short term memory recurrent neural network (lstm-rnn) made using the tensorflow library. The architecture of the network consists of 2 hidden LSTM layers, 1 dense layer, and the output layer. Both LSTM layers have a size of 64

with the first layer having its return_sequences parameter set to true. The dense layer has a size of 64, uses the relu activation function and has a dropout layer with a probability of 30%. Finally the output layer uses the softmax activation function and has a size of 10 neurons, one for each possible genre. The model was compiled with the Adam optimizer with a learning rate of 0.0001 and the sparse categorical crossentropy loss function. The model was then trained for 30 epochs and the test and validation accuracy and error per epoch was plotted alongside a confusion matrix after the training was completed. The model was trained for 30 epochs because it starts to overfit around 20-25 epochs.

Looking at figure 17, it can be seen that the accuracy of both the test and train data increase very rapidly, but the test accuracy levels off around 30 epochs. However the accuracy itself levels off at 61% making it worse than the convolutional neural network. This is also present in the confusion matrix in figure 19. However, the difference between this network and the convolutional network is that this one can accept data samples of varying sizes whereas all the data samples must be the same size for a convolutional network. This means that audio clips of different sizes (i.e. entire songs) can be used to train a recurrent neural network.

7. Conclusion

7.1. Discussion

Looking back at all the different models used, it can be seen that out of all of them, the convolutional neural network, with an accuracy of 70%, is the best when classifying music. The support vector machine and recurrent neural network come in at a close second at 64% and 61%

respectively. However, the recurrent network stands out because unlike all the other models, this one can use features of varying length. This means that it can be used on real world data (i.e full length songs) of varying length, something the other models can not do.

7.1. Future Experiments

For future experiments, instead of using constant fixed length data (like the samples in the GTZAN dataset) real world data can be used to train the recurrent neural network to possibly get a better accuracy rating then the convolutional neural network. Another possible future experiment is to use a LSTM network trained using real world data to generate music.