Natural Language Processing

Individual Project: Music Genre Classification

Professor Cristina Tirnuaca

Universidad de Cantabria

by:

Oluwakemi Odusanya

May 22, 2017

Introduction

Music as a Natural Language

Having chosen Music Genre Classification, a study that might not seem immediately relevant to my Natural Language Processing Class, I am inclined to take it as a responsibility to discuss whether or not music is relevant to the topic. In the process of making this connection, I will start by defining the two terms to be connected – Natural Language Processing and Music. According to Oxford English Dictionary, Natural language Processing is the application of computational techniques to the analysis and synthesis of natural language and speech. What then is a natural language? By the same dictionary, it is a language that has developed naturally in use (as contrasted with an artificial language or computer code). Since music has always been natural forms of communication and expression of feelings, with even much higher levels of ambiguity, it might make sense to say that music is a natural language.

In the West African Mande Kingdom, before the advent of literature and writing, music was the way history and legendary stories were passed down. The griots families, the families in charge of preserving histories, will compose different forms of music, both with and without speech, for important events and pass these down from generation to generation. Folk songs have been used in different cultures as a way to convey moods for different events. In monotheist religions like Christianity, music has been a key aspect of worship and praise. According to a research neurological basis of music at Johns Hopkins University, our brains do not discriminate between music and language structurally.

Why then should we exclude musical analysis from NLP? Although music might be a way of communication and interaction, it is not necessarily a language. Perhaps because it almost always deliberately ambiguous. Charles Limb, an accomplished neuroscientist, once said “the meaning of music is fundamentally context-specific and imprecise thereby differing wholly from meaning in natural language”.

How can we classify music?

In deciding how to classify music, I applied similar methodologies and concepts that we have learnt in class, particularly genre classification. First, I started by choosing what features are important for classifying music, then extracted the features I had chosen. I then went on to train a model based on the feature extracted. After training, like in any other natural language processing, it is necessary to evaluate and test the model. In the following sections, I will be talking about these steps in details – feature extraction, training and evaluation.

In building a classifier, I first downloaded about two hour long clips of each of the classes I am working on – Malian Blues, Heavy Metal and Afro-Cuban. I then used a python **AudioSegment** library to split these clips into 1 minute clips. We then had about a 100 samples for each class. The first file in my submission, ‘featuresExtract’, shuffles the files in the given directory, splits the files into train and test files depending on the split percentage given. 0.8 for example means that 80 percent of the files will be for training and the other 20 for testing. In my case, I have used 0.8. After extracting these features, because this takes a long time to do, we don’t want to do it every time, the function saves the features extracted into npy files

In the trainer class, the features are loaded from the npy files and training algorithms are ran on these features to build a model. This model is then evaluated and saved into a ckpt file because we don't want to train a model on the same features every time either.

After all the pre-processing is done, the third class – 'musicClassifier', when run records a sound signal in real time for 30 seconds, extracts it features, loads the trained model then classifies the song recorded. This works within a very short time, much shorter than it will take to run each part every time.

Feature Extraction

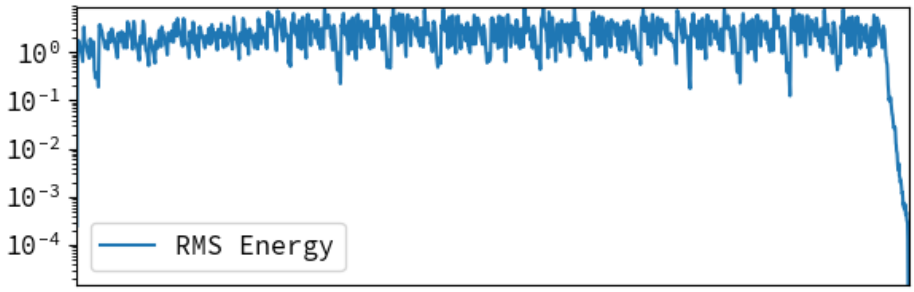
In this section, I am swamped by different questions – what makes up a genre or a certain kind of music? How important are the lyrics of a certain type of music? What about instrumental music with no speech? How do I as a human classify music to different genres and how can I train a computer to do it this way? How do other humans perceive and classify music? These questions are particularly hard because they are very subjective. Individuals differ on taste and perception of music and sound and are therefore bound to classify it differently. When I looked up the different genre listings there are, I found that these lists vary both in depth and width. In the question of whether speech is important or not, I decided it is not important. People can express love and happiness in the same lyrics yet different tempos or levels of aggression. Also a lot of genres are peculiar for their lack of lyrics for example classical music.

Luckily, there has been a lot of studies in this field, and it us not too hard to come up with what features to choose. The **Librosa** library is an open-source library that contains various functions

for audio file manipulations and feature extractions. By playing around with these features, I chose the following. I extracted the following features and found the mean for each of them.

Zero crossing rate: this is rate of change in zero crossing from window to window. Zero crossing is the point where the sign of a mathematical function changes. In this case, it is the instantaneous point where there is a zero signal. It mathematically represents how noisy the audio form is.

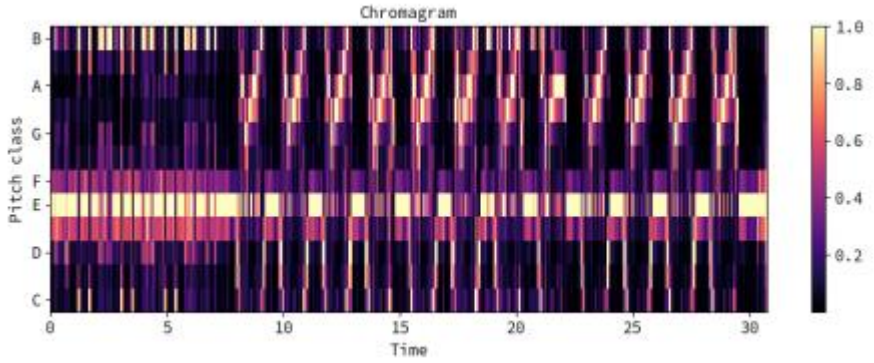
Root Mean Square Energy: This is a measurement of power of a signal. It is directly related to the area under an amplitude time graph of a music. If we imagined a music box to be a source of power that delivers current, the Root Mean Square Energy will be directly related to the current this music box delivers



log Power spectrogram

Mel Frequency Cepstral Coefficient (MFCC): these are a representation of the spectral envelope of a given signal [2] They are meant to extract features of the components of an audio signal that are relevant to identification. They are calculated by applying the discrete Fourier Transform and obtaining the spectral power of the signal [4].

Chromagram: This aims to capture or represent the 12 pitch classes: {C, C♯, D, D♯, E, F, F♯, G, G♯, A, A♯, B}. [3]



Tempo: This is a measurement of speed of an audio signal. It is calculated by the number of beats per minute

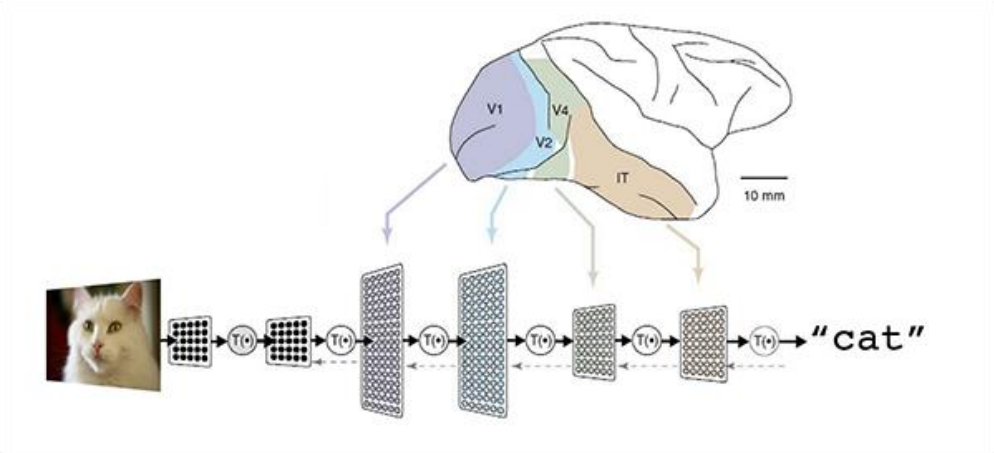
Tonal Centroid (Tonnetz): this is a pitch space defined by the network of relationships between musical pitches in just intonation [5]. It is particularly useful for western music, because they are usually arranged deliberately in tones.

Spectral Contrast: this is a measure of deviation of a sound from white noise[2]

Training

For the training part of this project, I used a deep learning library developed by Google Brain Team- **Tensorflow**. Tensorflow is relatively easy to learn with a lot of tutorials online that uses neural networks. Neural networks inspired by the way that the human brain works, whereby a collection of software neurons are created and connected together allowing them to message each other, then the network attempts to solve a problem repeatedly and gets better at it from repeatedly trying. With time and repetition, the networks come up with an ideal function to solve

the problems. Just like the human brain, as the series of networks get more accustomed to solving a certain problems of classification for example, they get better. The following link was very helpful for visualizing the neurons as the parameters change.



The image above is a representation of the layers in a 3-layer network. The layers are connected by 'synapses'

$$y = \sigma(S(\tanh x))$$

In my program, I have used 2 layers of networks that together to strengthen a network and create a non-linear function. Although researchers have used the neural connections of the brain to describe the connections, I find it easier to think of it as a composite function like the one above. At each trial of solving an equation, the model tries to tweak the parameters and constants associated with each layer of the formula, in order to come up with an ideal formula for prediction. In the case of my program, the value x will be the input features of the music that we have explained in the previous section. "S" is a symbol used to represent the sigmoid functions and "σ" if for representing the Softmax function. I started off with one softmax layer then added and a sigmoid layer.

Evaluations, improvements and Conclusion

Tensorflow has a set of functions that are very useful for evaluating models.

```
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
```

This returns an array of Booleans that says whether or not the predicted value is equal to the actual value. The accuracy is then the total number of 'Trues' divided by the size of the array.

My code currently performs relatively poorly with an accuracy if 90%. Moving forward, I would like to do the following in future to enhance my classifier:

- Add more training data to build a stronger network.
- Add more features: by reading more on important features and they significance, I can increase the accuracy of my model.
- Experiment more systematically on the hyper-parameters (learning rate, epochs and hidden units), I could improve the accuracy.
- Add extra layers to the neural network.
- Add more genres to expand the range of the classifications.

Bug unsolved

Ideally in order to use and run the classifier, I wanted to be able to run it from a different file, - 'musicClassifier' so That I am not having to run the training file every time I want to classify a music file. However, this portion is not working right and the values imported are not accurate. I have instead, included a recorder in the train file that takes in the music file and classifies it directly after training. I will be working on solving this in the future.

Files Submitted

In my submission, I have included the following files

- **musicClassifier.py**: contains the record sound and get genre functions but doesn't classify with the correct parameters. See bug above
- **featuresExtract**.py: this contains the functions that extracts features for music file and the main extracts them, given corresponding file paths
- **train.py**: this contains the train function that takes in npy files that have features stored in them
- **trainFeatures.npy, labels.npy, testFeatures.npy, testLabels.npy:** These are pre-processed data for features of 300 music files that I downloaded on my computer.

Reference and Useful tutorials

1. https://psmag.com/social-justice/music-language-75521

2. http://smcnetwork.org/files/proceedings/2009/174.pdf

3. https://en.wikipedia.org/wiki/Chroma_feature

4. https://es.wikipedia.org/wiki/MFCC

5. http://www.nyu.edu/classes/bello/MIR_files/tonality.pdf

6. https://pdfs.semanticscholar.org/6c16/dbc951be60e1edb1942389c3cb68eddccffc.pdf

https://www.tensorflow.org/versions/r0.10/tutorials/mnist/beginners/

http://nullege.com/codes/search/pydub.AudioSegment

http://deliprao.com/archives/100

http://www.jessicayung.com/code-explained-training-a-model-in-tensorflow/

https://www.tensorflow.org/get_started/mnist/pros