Submitted to:

INDIAN INSTITUTE OF MACHINE LEARNING, KOLKATA

MUSIC GENRE CLASSIFICATION

Author:

Shubham Tribedi [a]

[a] Department of Computer Science and Business Systems, Sister Nivedita University

Project Mentor:

Sayantan Chakraborty

Technical Director, Indian Institute of Technology and Management

Acknowledgements

I would like to express my special thanks of gratitude to our Project Mentor: Sayantan Chakraborty who gave me the golden opportunity to do this wonderful project on the topic of Music Genre Classification, which also helped me in doing a lot of Research and I came to know about so many new things. I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

I am are over whelmed in all humbleness and gratefulness to acknowledge my depth to all those who have helped me to put these ideas, well above the level of simplicity and into something concrete.

I would like to thank my faculties who helped me a lot in gathering different information, collecting data and guiding me from time to time in making this project, despite of their busy schedules, they gave me different ideas in making this project unique.

Thanking you,

Shubham Tribedi

B. Tech CSE (TCS).

Preface

The purpose of this project is to provide a conceptual introduction to statistical or machine learning techniques. Our objective was to automatically classify different musical genres from audio files. We will classify these audio files using their low-level features of frequency and time domain. For this project we need a dataset of audio tracks having similar size and similar frequency range. GTZAN genre classification dataset is the most recommended dataset for the music genre classification project and it was collected for this task only.

Firstly, the dataset was downloaded and the corrupted files were removed, Then the approaches such as Multiclass Support Vector Machines, K-Means Clustering, K-Nearest neighbours, Convolutional Neural Networks were tried and the K-Nearest neighbours algorithm showed the best results for this problem. It is a popular machine learning algorithm for regression and classification which makes prediction on data points based on their similarity measures i.e., distances between them. The Features were extracted after that on the basis of which the predictions were to be done. Then using the model predictions were made using some test .wav files.

Keywords: GTZAN, genre, frequency, classification, distance, features, noise, speech, vectors, neighbours, split, dataset, accuracy, directory, prediction.

Contents

1. Introductory Concepts

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.

1.1 Introduction

Companies nowadays use music classification, either to be able to place recommendations to their customers (such as Spotify, Soundcloud) or simply as a product (for example Shazam). Determining music genres is the first step in that direction. Machine Learning techniques have proved to be quite successful in extracting trends and patterns from the large pool of data. The same principles are applied in Music Analysis also. Sound is represented in the form of an audio signal having parameters such as frequency, bandwidth, decibel etc. A typical audio signal can be expressed as a function of Amplitude and Time. These sounds are available in many formats which makes it possible for the computer to read and analyze them. Some examples are:

- mp3 format
- WMA (Windows Media Audio) format
- wav (Waveform Audio File) format

In this project we are going to develop a machine learning project to automatically classify different musical genres from audio files. We will classify these audio files using their low-level features of frequency and time domain.

1.2 Dataset

For this project we need a dataset of audio tracks having similar size and similar frequency range. GTZAN genre classification dataset is the most recommended dataset for the music genre classification project and it was collected for this task only.
The GTZAN genre collection dataset was collected in 2000-2001. It consists of 1000 audio files each having 30 seconds duration. There are 10 classes (10 music genres) each containing 100 audio tracks. Each track is in .wav format. It contains audio files of the following 10 genres:

- Blues
- Classical
- Country
- Disco
- Hip-hop
- Jazz
- Metal
- Pop
- Reggae
- Rock

1.3 Test Data

For this project the testing data was downloaded from https://uweb.engr.arizona.edu/~429rns/audiofiles/audiofiles.html and it was used to test whether the classification of audio files into specific genres was done correctly or not. Various WAV Files such as Sampling rate: 11025Hz, 8-bits/sample, Sampling rate: 16000Hz, 16-bits/sample, Sampling rate: 44.1kHz, 16-bits/sample were used. The test data was put together in March 4, 2005 and last modified to the most accurate model in April 24, 2009

1.4 Motivation

To our group this topic was motivating because we finally could see how all the mathematics we studied in school and high school combined and also those which we studied in university applied in real life. It was not only interesting but also it was very useful.

The thought that we can extract something useful from a data and make predictions was interesting. Training the computer to make predictions seemed very interesting and it could also bring major breakthroughs in the existing scenario.

We had many questions in our mind like:

- Can computers take decisions (even smart smarter ones) just like humans?
- Can computers help humans in doing their tasks of daily living?
- Can we build a smart eco-system where users get feedback and systems can update their actions?
- Can we develop technology to learn from human behavior?

To answer all of these we got only one answer and that is machine learning and this was the most powerful tool to do so. Hence with the motivation of discovering how can we train a computer to make predictions based on its learning we started this project and ended creating a model which was accurate and could easily classify any audio file in the respective genre.

1.5 Contribution

The first part was to collect the data required and pre-processing the same. The second part was to train and test the model which we created. The other tasks such as Feature Extraction and Using the models to train the model were also done by me.

2. Proposed Methodology

There are various methods to perform classification on this dataset. Some of these approaches are:

- Multiclass support vector machines
- K-means clustering
- K-nearest neighbors
- Convolutional neural networks

We will use K-nearest neighbors' algorithm because in various researches it has shown the best results for this problem.

K-Nearest Neighbors is a popular machine learning algorithm for regression and classification. It makes predictions on data points based on their similarity measures i.e., distance between them.

2.1 Collection of Data

We used the GTZAN Genre Collection Dataset for our project. This dataset was used for the well-known paper in genre classification " Musical genre classification of audio signals " by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002.
Unfortunately, the database was collected gradually and very early so there are no titles. The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions. A similar dataset which was collected for the purposes of music/speech discrimination. The dataset consists of 120 tracks, each 30 seconds long. Each class (music/speech) has 60 examples. The tracks are all 22050Hz Mono 16-bit audio files in .wav format. The data was collected from Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals) which is an open-source software framework for audio processing with specific emphasis on Music Information Retrieval applications. It has been designed and written by George Tzanetakis (gtzan@cs.uvic.ca) with help from students and

researchers from around the world. Marsyas has been used for a variety of projects in both academia and industry.
Marsyas is a software framework for rapid prototyping and experimentation with audio analysis and synthesis with specific emphasis to music signals and Music Information Retrieval. The basic goal is to provide a general, extensible and flexible architecture that allows easy experimentation with algorithms and provides fast performance that is useful in developing real time audio analysis and synthesis tools.
A variety of existing building blocks that form the basis of most published algorithms in Computer Audition are already available as part of the framework and extending the framework with new components/building blocks is straightforward.

2.2 Pre-Processing of Data

The preprocessing involved in preparing the GTZAN dataset is resampling to .wav at 22050 Hz. The preparation scripts require the following packages to be installed beforehand:

- ffmpeg version N-81489-ga37e6dd
- numpy 1.11.2+mkl
- librosa 0.4.0
- h5py 2.6.0

The steps involved in pre-processing were downloading the dataset and executing the script:

```
import os
from fnmatch import fnmatch
SRC_PATH = 'G:\dataset_gtzan\\gtzan-master'
f = open('gtzanact0wavworkfile.txt', 'w')
filePathString = []
scriptLine = []
for path, subdirs, files in sorted(os.walk(SRC_PATH, topdown=False)):
    for name in sorted(files):
        if fnmatch(name, "*.au"):
            filePathString = path + '\\' + name
            scriptLine = 'ffmpeg -y -i "' + filePathString + '" -ab 128k -acodec pcm_s16le -ac 1 -ar 22050 "'+ filePathString.replace('.au','.wav') + '"\n'
            f.write(scriptLine)
```

The data in the dataset was in au format this was needed to be converted into .wav format for further training. Then the spectrogram transformation was configured within the Dataset Transformer and the MCLNN-Ready hdf5 were generated for the dataset.

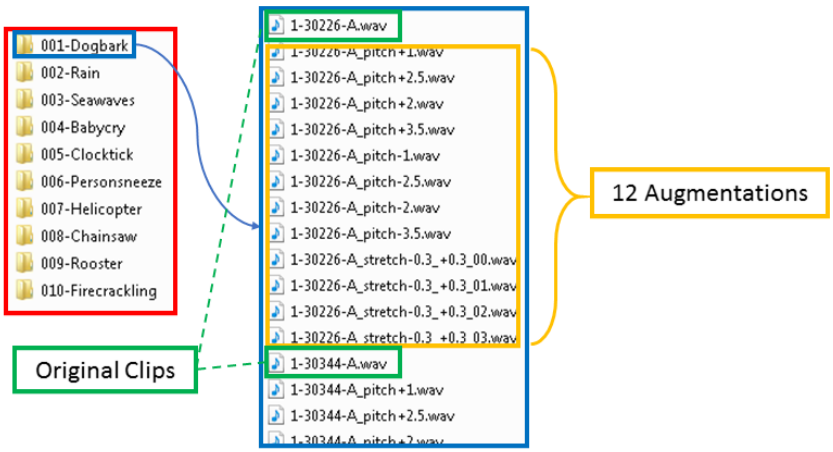
The transformation involves the generation of a single file containing the intermediate representation of a signal, e.g. spectrogram in case of sound.

The transformer loads each file in order from the dataset, applied the intermediate representation and stores the resulting transformation to a single hdf5 file (referred later as Dataset.hdf5) for the whole dataset with augmentation.

Augmentation is a method to apply certain controlled deformations to the dataset while keeping the properties of the original sample to a certain extent. This process enhances the generalization of a model during training.

The below figure shows the folder structure expected while generating the. hdf5 for an augmented ESC10 dataset. This dataset was augmented using 12 variants.
Accordingly, the figure shows each original file of the dataset and the accompanying 12 versions.

The order of the files should match the structure shown below in terms of the original clip followed by its augmented variants, since the index generation is dependent on it.



Then the index from the csv were loaded. It was done because if the dataset is accompanied with a CSV file, specifying the samples filename and category will be needed.

The below figure shows a chunk of the CSV file released with the Urbansound8k. The file is not exactly the original one, but rather a modified version in terms of the rows ordering in the csv file without changing the data and the sequence folder added.

The indices of the two highlighted columns are required for the configuration as shown in the below listing.

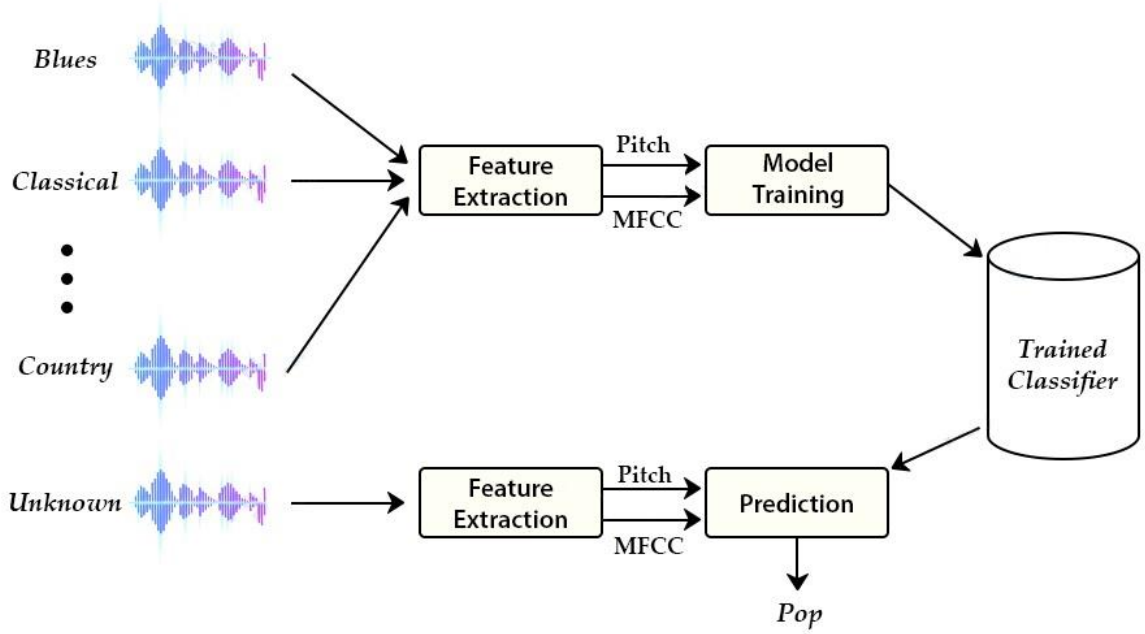
dataset_seq	per_class_seq	slice_file_name	fsID	start	end	salience	fold	classID	class
0	0	127873-0-0-0.wav	127873	0	2.04075	1	1	0	air_conditioner
1	1	134717-0-0-0.wav	134717	0	4	1	1	0	air_conditioner
2	2	134717-0-0-1.wav	134717	0.5	4.5	1	1	0	air_conditioner
3	3	134717-0-0-12.wav	134717	6	10	1	1	0	air_conditioner
4	4	134717-0-0-13.wav	134717	6.5	10.5	1	1	0	air_conditioner
5	5	134717-0-0-14.wav	134717	7	11	1	1	0	air_conditioner
6	6	134717-0-0-15.wav	134717	7.5	11.5	1	1	0	air_conditioner
7	7	134717-0-0-16.wav	134717	8	12	1	1	0	air_conditioner
8	8	134717-0-0-18.wav	134717	9	13	1	1	0	air_conditioner
9	9	134717-0-0-19.wav	134717	9.5	13.5	1	1	0	air_conditioner
10	10	134717-0-0-2.wav	134717	1	5	1	1	0	air_conditioner

Then the indices for the folds were generated thus we got the final pre-processed data in the form of .wav files and the data were now ready to be used for training our model.

2.3 Proposed Model and Mathematical Background

AS K-Nearest neighbor algorithm showed best result for this model hence it was used for the final evaluation as well. More about this algorithm will be discussed in the next section (2.4 Algorithm).

The following diagram shows the proposed model through a diagrammatic representation:



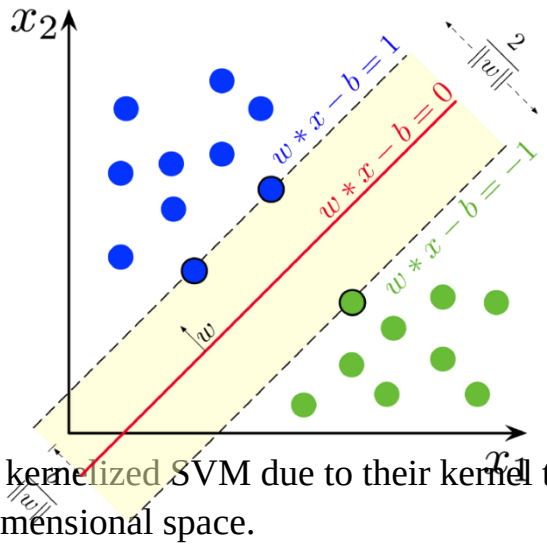
2.4 Algorithm

There are various algorithms to perform classification on this dataset. Some of these approaches are:

- Multiclass support vector machines
- K-means clustering
- K-nearest neighbors
- Convolutional neural networks

Multiclass support vector machines:

The objective is to find a hyperplane in an n-dimensional space that separates the data points to their potential classes. The hyperplane should be positioned with the maximum distance to the data points. The data points with the minimum distance to the hyperplane are called Support Vectors. Due to their close position, their influence on the exact position of the hyperplane is bigger than of other data points. In the graphic below the Support Vectors are the 3 points (2 blue, 1 green) laying on the lines.



SVMs are also called kernelized SVM due to their kernel that converts the input data space into a higher-dimensional space.
The input space X consists of x and x'.

$$\phi(x_i)$$

represents the kernel function that turns the input space into a higher-dimensional space, so that not every data point is explicitly mapped.
The kernel function can also be written as

$$k(x, x')$$

How the function is defined and useful for laying hyperplanes depends on the data:

Kernel Functions

The most popular kernel functions, that are also available in scikit-learn are linear, polynomial, radial basis function and sigmoid. In the following you can see how these four kernel functions look like:

- Linear Function $k(x_i, x_j) = x_i * x_j$
- Polynomial Function $k(x_i, x_j) = (1 + x_i * x_j)^d$
- Radial Basis Function (RBF) $k(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2)$
- Sigmoid Function $k(x_i, x_j) = \tanh(\alpha x^T y + c)$

Mathematical Concept Behind SVM

The following formula poses the optimization problem that is tackled by SVMs. It is explained further down (scikit-learn, n.d.):

$$\min_{\omega, b, \zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

$$y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i$$

where ζ_i denotes the distance to the correct margin with $\zeta_i \geq 0,\ i = 1, \ldots, n$

where C denotes a regularization parameter

were $w^T w = \|w^2\|$ denotes the normal vector

where $\phi(x_i)$ denotes the transformed input space vector

where b denotes a bias parameter

where y_i denotes the i-th target value

The objective is to classify as many data points correctly as possible by maximizing the margin from the Support Vectors to the hyperplane while minimizing the term

$$w^T w$$

In other words, the objective can also be explained as finding optimal w and b that most samples are predicted correctly.

Mostly not all data points can be allocated perfectly so that a distance to the correct margin is represented by

$$\zeta_i$$

The normal vector creates a line that runs through the coordinate origin. The hyperplanes cut this straight line orthogonal at a distance

$$\frac{b}{\|w\|_2}$$

from the origin.

For the ideal case (Bishop, p.325 ff., 2006)

$$y_i(\omega^T \phi(x_i) + b)$$

would be ≥ 1 and followingly perfectly predicted. Having now data points with distance to their ideal position, lets us correct the ideal case of being ≥ 1 to

$$\geq 1 - \zeta_i$$

Simultaneously a penalty term is introduced in the minimization formula. C acts as a regularization parameter and controls how strong the penalty is regarding how many data points have been falsely assigned with a total distance of

$$\sum_{i=1}^{n} \zeta_i$$

Multiclass Classification using Support Vector Machine:

In its most simple type SVM are applied on binary classification, dividing data points either in 1 or 0. For multiclass classification, the same principle is utilized. The multiclass problem is broken down to multiple binary classification cases, which is also called one-vs-one. In scikit-learn one-vs-one is not default and needs to be selected explicitly (as can be seen further down in the code). One-vs-rest is set as default. It basically divides the data points in class x and rest. Consecutively a certain class is distinguished from all other classes.

The number of classifiers necessary for one-vs-one multiclass classification can be retrieved with the following formula (with n being the number of classes):

$$\frac{n * (n-1)}{2}$$

In the one-vs-one approach, each classifier separates points of two different classes and comprising all one-vs-one classifiers leads to a multiclass classifier.

K-Means Clustering:

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.

Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

AndreyBu, who has more than 5 years of machine learning experience and currently teaches people his skills, says that "the objective of K-means is simple: group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset."

A cluster refers to a collection of data points aggregated together because of certain similarities.

We define a target number k, which refers to the number of centroids we need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.

In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.

How the K-means algorithm works:

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids

It halts creating and optimizing clusters when either:

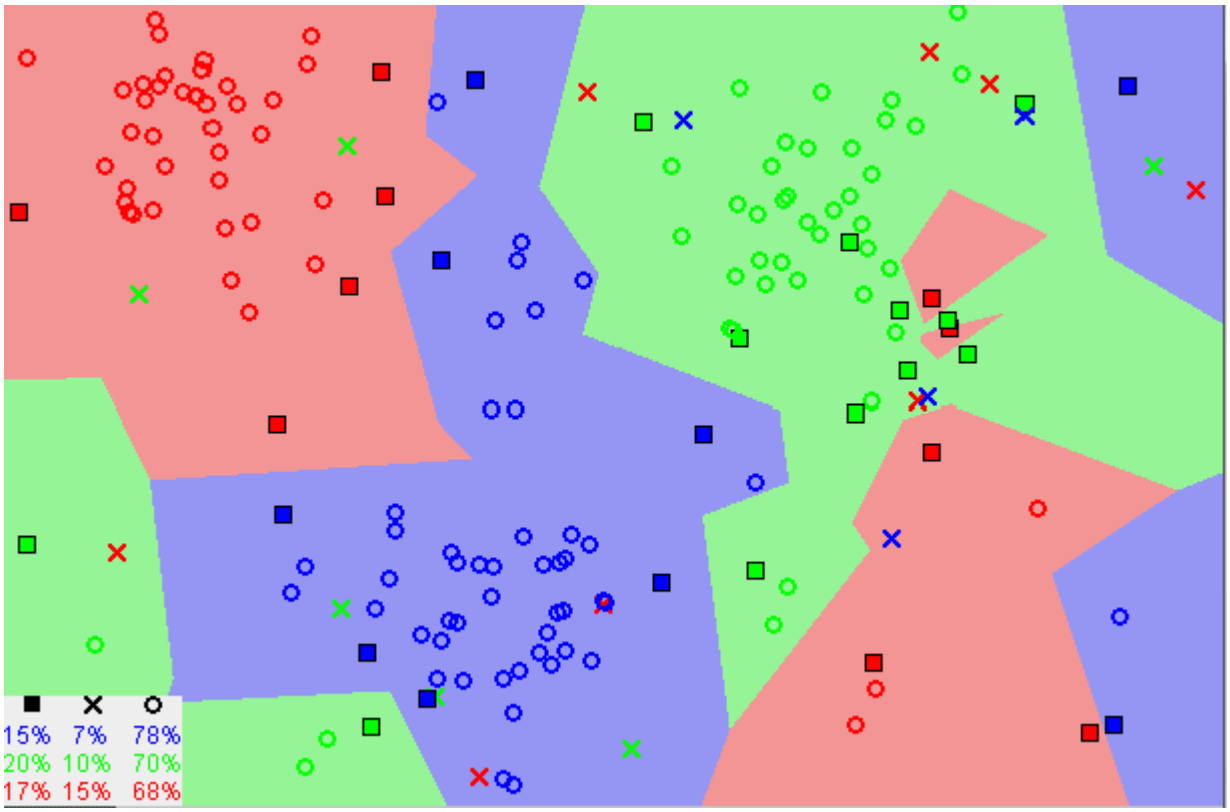
The centroids have stabilized — there is no change in their values because the clustering has been successful. The defined number of iterations has been achieved. K-means clustering is an extensively used technique for data cluster analysis.

It is easy to understand, especially if we accelerate our learning using K-means clustering. Furthermore, it delivers training results quickly. However, its performance is usually not as competitive as those of the other sophisticated clustering techniques because slight variations in the data could lead to high variance.

Furthermore, clusters are assumed to be spherical and evenly sized, something which may reduce the accuracy of the K-means clustering Python results.

K-Nearest Neighbours:

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.



Notice in the image above that most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

Choosing the right value for K

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (top left corner of the coloured plot above), but the green is the single nearest neighbour. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.

Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.

In cases where we are taking a majority vote (e.g., picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Advantages

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.

- The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

Disadvantages

- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

The k-nearest neighbours (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

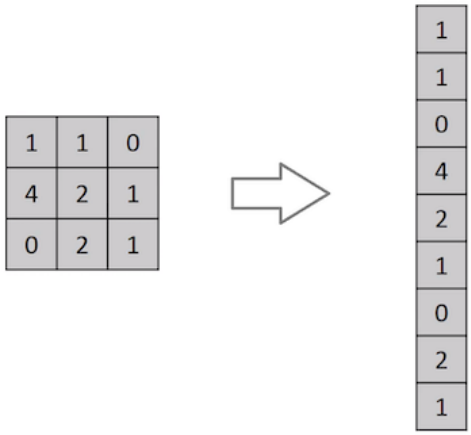
In the case of classification and regression, we saw that choosing the right K for our data is done by trying several Ks and picking the one that works best.

Convolutional neural networks:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Why ConvNets over Feed-Forward Neural Nets?



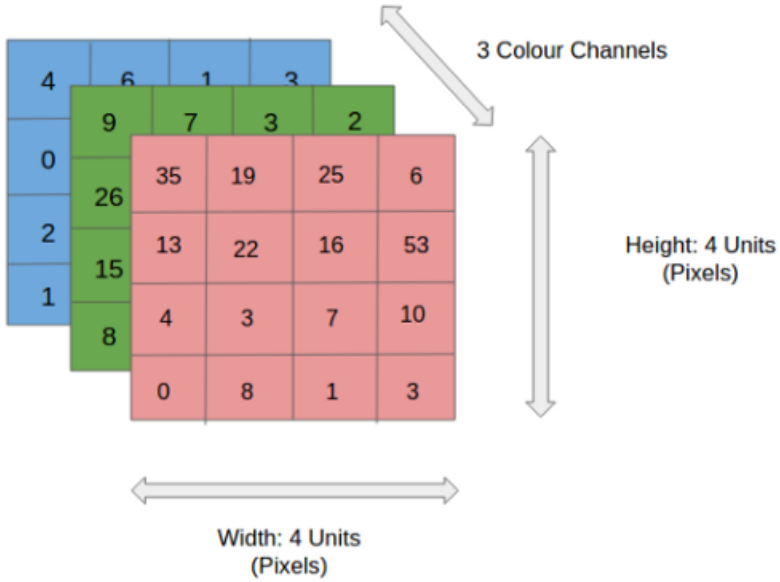
Flattening of a 3x3 image matrix into a 9x1 vector

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g., 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes?

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

Input Image:



In the figure, we have an RGB image which has been separated by its three colour planes — Red, Green, and Blue. There are a number of such colour spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

We can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

2.5 Feature Extraction

In machine learning, pattern recognition, and image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g., the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

Feature extraction involves reducing the number of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power, also it may cause a classification algorithm to overfit to training samples and generalize poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. Many machine learning practitioners believe that properly optimized feature extraction is the key to effective model construction.

Results can be improved using constructed sets of application-dependent features, typically built by an expert. One such process is called feature engineering. Alternatively, general dimensionality reduction techniques are used such as:

- Independent component analysis
- Isomap
- Kernel PCA
- Latent semantic analysis
- Partial least squares
- Principal component analysis

- Multifactor dimensionality reduction
- Nonlinear dimensionality reduction
- Multilinear Principal Component Analysis
- Multilinear subspace learning
- Semidefinite embedding
- Autoencoder

After the Preprocessing is done and the dataset is loaded, we now need to extract the features on the basis of which the prediction will be done. For this task we will be using MFCC (Mel Frequency Cepstral Coefficient), more about this will be discussed in a later part of the report. We took rate and signature to be the basis of our classification. The features: rate and signature were chosen because they were the most efficient and a trend was found in them which could be used for differentiating the music's into their respective genres. After the features were extracted, they were dumped into the my.dat datafile for storing the features all in one place. All the features of all the different files irrespective of their genres are stored in the my.dat data file which was to be used at a later stage for prediction.

2.6 Mel Frequency Cepstral Coefficient

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

MFCCs are commonly derived as follows:

Take the Fourier transform of (a windowed excerpt of) a signal.
Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.

Take the logs of the powers at each of the mel frequencies.
Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
The MFCCs are the amplitudes of the resulting spectrum.
There can be variations on this process, for example: differences in the shape or spacing of the windows used to map the scale,[3] or addition of dynamics features such as "delta" and "delta-delta" (first- and second-order frame-to-frame difference) coefficients.

The European Telecommunications Standards Institute in the early 2000s defined a standardized MFCC algorithm to be used in mobile phones.

These are state-of-the-art features used in automatic speech and speech recognition studies. There are a set of steps for generation of these features:

Since the audio signals are constantly changing, first we divide these signals into smaller frames. Each frame is around 20-40 ms long
Then we try to identify different frequencies present in each frame
Now, separate linguistic frequencies from the noise

To discard the noise, it then takes discrete cosine transform (DCT) of these frequencies. Using DCT we keep only a specific sequence of frequencies that have a high probability of information.

2.7 Training and Testing

In a dataset, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a dataset is divided into a training set, a validation set (some people use ‘test set’ instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

In Machine Learning, we basically try to create a model to predict the test data. So, we use the training data to fit the model and testing data to test it. The models generated are to predict the results unknown which is named as the test set. As we pointed out, the dataset is divided into train and test set in order to check accuracies, precisions by training and testing it on it.

The proportion to be divided is completely up to you and the task you face. It is not essential that 70% of the data has to be for training and rest for testing. It completely depends on the dataset being used and the task to be accomplished. For example, a simple dataset like Reuters. So, assume that we trained it on 50% data and tested it on rest 50%, the precision will be different from training it on 90% or so. This is mostly because, in Machine Learning, the bigger the dataset to train is better. It now depends on us, what precision or accuracy we need to achieve based on our task.

- Training Set: Here, we have the complete training dataset. We can extract features and train to fit a model and so on.
- Validation Set: This is crucial to choose the right parameters for our estimator. We can divide the training set into a train set and validation set. Based on the validation test results, the model can be trained (for instance, changing parameters, classifiers). This will help us get the most optimized model.
- Testing Set: Here, once the model is obtained, we can predict using the model obtained on the training set.

3. Experimental Result and Discussion

After training and testing the model we now have an accurate model which can classify the music files into their respective genres. The experimentation, prediction and results are provided in the following sections.

3.1 Dataset

The Dataset was the best dataset which could be used for training such model which needs to classify music files based on their genres. This Dataset was the most accurate trainer which could train the model and test it accurately

3.2 Experimentation

The procedure starts with the dataset having all the features extracted and then a prediction model which can predict any music into the respective genre. All the features are extracted in a common data file which could be referred when the prediction is being done

3.3 Prediction & Result

The Predictions were accurate as the music files were classified into the correct genre. Our Model achieved an accuracy of 72% which is a good standard accuracy keeping in mind that now a days the genres are mixed up together and remixes are made. Hence this accuracy was enough for us to declare it a model which could predict the genre of a music file.

4. Conclusion and Future Work

Although regression is an important problem in data analysis, it has not been dealt with extensively by the ML community. Classifications of audio tracks on the basis of musical genre has been a repeated topic of constant research for several years and a challenge which has presented increasingly small performance gains in recent times. Classification of music genre can serve a myriad range of purposes - from empowering automated sorting of tracks in musical apps to audio detection from a musical context. While the problem poses a difficult terrain to tread on, it could help creators of musical applications more easily and effectively distinguish and classify tracks without the need for large-scale human intervention. It could also help virtuosos and musical enthusiasts take a sample track and figure out its respective genre should they choose to concentrate on exploring that particular field. Now, classification problems require employing appropriate machine learning algorithms to certify a label to examples from the problem domain. The efficacy of the designers lies in their efficient use of proper methods to predict correctly the genre of the input sample. The goal is hence to create a succinct model by appropriate and rigorous training of datasets to correctly predict the values of one or more outcomes. Thus, classification problems fall under the umbrella of supervised machine learning where the goal is to appropriately map from the input variables to the output variable and to approximate this mapping function Y=f(x). For our particular scenario, it was crucial to find an appropriate target dataset and extract features that would amplify the chances of correct prediction. The goal was to find audio tracks with similar sizes and frequency ranges and we opted for using the GTZAN Dataset for this purpose. GTZAN was collected across a decade spanning from 1999 to 2009, comprising of 1000 tracks, each having a duration of 30 seconds. The millennial track set heralded a hundred Western tunes each from one particular genre with ten genres in total including Country, Disco, Blues, Classical, Hip-Hop, Jazz, Metal, Pop, Reggae and Rock. All of the tracks were of course instrumentals devoid of lyrics and intelligible noise. The problem thus narrowed down to feature extraction for training the model. This was an

incredibly important step as the correct interpretation of test data depended on the variables of comparison. The best way to differentiate between different musical genres was decided to be the pitch and the mfcc. The pitch of sounds, allows an ordering on the basis of frequency range. In other words, the sensation of frequency is known as the pitch on the basis of which we can demarcate between a high or low sound. The higher the pitch, the higher is the frequency and vice versa. On the other hand, MFCCs or Mel Frequency Cepstral Coefficients are used in automatic speech recognition systems. Mel scale is a scale that relates the perceived frequency of a tone to the actual measured frequency. It scales the frequency in order to match more closely what the human ear can hear (humans are better at identifying small changes in speech at lower frequencies). This scale has been derived from sets of experiments performed on human subjects. We know that the range of human hearing is 20Hz to 20kHz. Now, imagine a tune at 300 Hz and another similar sound at 400 Hz. The latter would appear as slightly higher pitched. Now, imagine another tune at 900Hz and 1kHz. In this case, the difference in pitch between the two sounds will be perceived as far less severe though it is exactly the same in both cases, viz., 100Hz. The mel scale tries to capture such differences. The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope. MFCCs separate linguistic frequencies from noise, the latter being discarded and helps in identification of different frequencies distinctly. So, with the features being chosen, we needed an approach for training the model. Multiclass SVM, K-means clustering, K-nearest neighbors and Convolutional Neural Networks were some approaches available for choosing. But KNN being the most efficient was preferred which makes prediction on data points based on their perceived similarities.

5. References

- Our Training Dataset: http://marsyas.info/downloads/datasets.html
- Our Testing Dataset: https://uweb.engr.arizona.edu/~429rns/audiofiles/audiofiles.html
- Our Reference Research Paper: " Musical genre classification of audio signals " by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002.