Chapter 6

Contrast between Simple and Complex Classification Algorithms

Divy Dwivedi, Ashutosh Ganguly, V V Haragopal

divydwivedi@gmail.com
ashutoshganguly1997@gmail.com
haragopal_vajjha@hyderabad.bits-pilani.ac.in

Abstract

The central theme of this chapter is to explore classification algorithms on complex datasets which have a high number of features. The dataset used in this project is the GTZAN dataset, which has 60 quantified features extracted from 10 genres of music. The chapter addresses Music Information Retrieval (MIR) and sheds light on the features involved in audio signal processing, its importance and ways to model it. Many studies have underlined the classification using complex machine learning models but the absence of any project which highlights the use of well known statistical classification methods encouraged us to tread this path. By the end of this chapter, one will have a detailed understanding of K nearest-neighbors, Fisher Linear Discriminant Analysis, Quadratic Discriminant Analysis and Feedforward Neural Networks. The chapter will also give one contrast between the results obtained after fitting the dataset in these classification models. MIR is one of the fastest developing fields that have many applications in Music Informatics; many audio streaming applications use classification techniques extensively to provide their users with the best recommendations and quantify their subjective music taste. Another popular application of audio signal processing is in the fast-growing podcast and audiobooks industry, recommendations of podcasts and contextual understanding of books to generate useful summaries, narrate them using digital assistants are a few use cases of the same. Therefore it will be fruitful to know what techniques can be best suited for the job.

Keywords: Machine Learning, Deep Learning, Statistics, Music Classification, Linear Discriminant Analysis(LDA), Quadratic Discriminant Analysis(QDA), K Nearest Neighbors(KNN), Neural Networks(NN), Audio Signal Processing

6.1 Introduction

Statistical methods form the backbone of many industries which use data as a primary source of information for insights, modelling and predictions. Statistics mainly deals with contrivances to understand data and extract valuable information from the same. A prominent topic in statistical learning is classification. Classification is the process of tagging a particular data point to its corresponding category to group it with other similar data points. Data can be qualitative and quantitative, a data set is said to be quantitative if one can assign numerical values to explanatory variables to make sense of the data. If one can get meaningful numerical answers to questions like ‘How much?’ or ‘How many?’ then quantitative data is into play. On the other hand, qualitative data is non-numerical data that approximates and characterizes the subjectivity of the variables, for example, the door is black, the sky is blue. For the example of qualitative data, one cannot typically measure the value because the colour states the quality of the door. In this chapter, one primarily learns the approaches to quantify and predict the qualitative responses, identify useful variables for classification, and fit algorithms to see their performance on the data set.

In our chapter, the authors have taken up the dataset of various music genres. Music taste for each individual may vary largely, therefore music is divided into many different genres which further have their subcategories. The first challenge in this classification is identifying the features based on which one can differentiate a genre of music from another. Music genres are hard to systematically and consistently describe due to their inherently subjective nature.
Therefore, this chapter answers the following questions:

- What features of music can be used to classify them into various genres?
- How accurately can music genres be classified using classical statistical methods and complex algorithms like neural networks?

Here the authors have used the GTZAN dataset and have classified them into ten categories. By the end of this chapter, one will have a clear understanding and appreciation for the various statistical methods as well as the capabilities of a neural network. It makes sense that one compares the results of different methods over the same dataset as that broadens the perspective of a reader to get a practical understanding of the process involved and singularly look at the performance of the algorithm. The following section explores the features of music which are going to be used in the coming sections to classify the dataset into the respective categories.

Few research papers have outlined in detail the wants of the GTZAN dataset [1]. They explore the faults and inefficacies of the dataset. It is highly recommended to go through the paper to get a deeper understanding of the dataset and more importantly how to deeply analyse a dataset.

6.2 Data Preprocessing and Feature Extraction

This section entails the information about the dataset used and the features used to classify the musical tracks into their respective genres. The GTZAN dataset contains music of ten different genres, each sample of music being thirty seconds long. The ten genres in the dataset are Rock, Pop, Classical, Blues, Country, Disco, Jazz, Metal, Reggae and Hip-hop. The music files are of .wav format and are single-channel recorded at twenty-two thousand and fifty (22050) Hz [2]. The total number of musical pieces in the dataset is 10000. Additionally, the authors have used python 3 as the platform for running and creating our model. Python libraries make it easier for analysis. Here, 'librosa' is used to extract the features and analyse the audio files. The authors have used NumPy and Pandas for data preprocessing and feature extraction, as for modelling Scikit Learn is used from which KNN LDA and QDA models have been directly imported. TensorFlow and Keras are used for training the neural network, finally, Matplotlib is used for graphical representations of results.

6.2.1 Data Preprocessing

After extracting the features mentioned above, let's split the dataset into training and testing datasets in a 4:1 ratio. The authors have resolved the categories. One does so using the LabelEncoder and OneHotEncoder library from python. The authors have extensively used IBM SPSS Statistics to analyze the dataset. On extraction, one sees that in total there are 60 features that have been considered in the chapter. Of those, let's drop 'length' and 'filename' which refer to the length of each audio file, i.e. 3 seconds long, and the name of the audio file respectively.

6.2.2 Feature Study

This section entails the features used to classify the audio files. Feature extraction is an important part of a study. This can make a huge difference in the results and inferences obtained from the dataset. Therefore it is highly recommended to make a thorough study of the dataset to identify the best features. The following features have been extracted from the audio files to make a reasonable classification.

1. **Chroma**

The chroma is a set of mid-level features representing the tonal content; it is used since it is not affected by the change in instrumentation, timbre or dynamics. Chroma features are important while doing semantic analysis of the audio file, examples could include recognition of the set of chords being played and/or how similar or dissimilar two audio files are based on their harmonic nature. One first performs Short-Time Fourier Transforms on the audio files, due to its exponential nature and then take a logarithm of it to bring the representation into a linear scale. In simple words, this feature consists of twelve bands that represent musical notes namely: C, C#, D, D#, E, F, F#, G, G#, A, A#, B [3]. All frequencies for an audio file are associated with

one of these bands and hence an audio file has now been converted into a linear and discrete visual representation of it.

2. **Root-mean-square(RMS)**

RMS is an important feature since it can be used to represent the energy captured by a wave of fixed length. RMS is an indicator of loudness, for instance, louder sounds will contain more energy hence a metal genre sound wave will have a higher RMS than a wave belonging to the blues genre [3]. An interesting fact about the RMS is that it is less sensitive to outliers because for each frame of an audio signal, rather than taking a single value, the sum of the root square of the mean is taken, which makes it a good feature to take into account. It can be easily calculated using the following formula:

$$rms = \sqrt{\frac{x_1^2+x_2^2+...+x_n^2}{n}} \qquad (6.1)$$

3. **Centroid**

Centroid is a feature in the spectral feature set that can be calculated by weighing the frequencies and then taking their mean.

$$centroid(t) = \frac{\sum_i S[k_i,t]*freq[k_i]}{\sum_j S[j,t]} \qquad (6.2)$$

Where i and j range from 0 to $n-1$. Here t represents the time frame (1s), n is frequency values per second, S represents a spectrogram of the sound wave which can be generated using Fourier transformation on the wave and $freq$ is the collection of frequencies found in the kth row of the spectrogram. Spectral centroid can be used to understand the brightness of the sound wave. Intuitively a certain collection of higher frequencies would make a wave sound brighter than if it were replaced by the same collection of frequencies in the lower octave.

4. **Bandwidth**

Bandwidth is another feature in the spectral feature set that can be calculated by subtracting the modulus of the highest and modulus of the lowest frequencies. It can be measured using the following formula.

$$bandwidth(t) = \sum_i S[k_i,t] * (freq[k_i,t] - centroid[t]^p)^{1/p} \qquad (6.3)$$

Where i takes values from 0 to $n-1$. The rest of the symbols have the same meaning as in the Spectral centroid. The default value of p is set to 2, which makes it a second-order bandwidth. The unit of bandwidth is Hertz. This can be used to specify the dynamic range of the sound.

5. **Zero-Crossing Rate**

This feature gives us a count of how many times the wave crosses the x-axis. It is an important feature to take into account because it gives us a representation of the smoothness of the wave by telling the number of times it becomes from positive to negative and vice versa. For instance, the zero-crossing rate of an audio file of the metal genre will be higher than that of audio belonging to the blues genre. It can be calculated using the following expression.

$$zcr = \frac{1}{2}\sum_i |sgn(s_i) - sgn(s_{i+1})| \quad (6.4)$$

Where i takes values from $t * K$ to $(t+1) * K - 1$, sgn is the signum function, s_i is the amplitude of the ith frame and K is the number of frames in the audio signal. One of the major applications of ZCR is in distinguishing between high pitched sounds and percussive sounds.

6. **Rolloff**

The third feature in the set of spectral features, roll-off gives an idea about the shape and size of the sound wave. The roll-off (spectral) for an audio wave is a benchmark frequency decided by the modeller below which a certain constant amount of wave energy (say 50%) lies.

7. **Tempo**

This is the number of beats a fixed length of our sound wave displays. Here the fixed length is 1 second hence the unit beats per second (bps)

8. **Mel-Frequency Cepstral Coefficients**

The Mel frequency cepstral coefficients (MFCCs) are a set of features just like chroma or spectral. MFCCs were developed at MIT during the late 1960s to study the echoes in seismic audio [4]. It also is used to model the characteristics of the human voice. MFCCs are one of the most important characteristics of the sound which will be used in this project. For calculating the cepstrum, first take the discrete Fourier transform of the signal, followed by a logarithm which is finally again taken a Fourier inverse of, hence the spectrum of a spectrum called cepstrum.

9. **Harmony**

Harmony is the superposition of sounds, which can be analysed by a human ear. This means when particular sound frequencies with specific amplitude and frequency, pitches(tones or notes) or chords occur simultaneously. It is easier to understand the harmony in musical terms, for instance, in a piano staff, the key A4 has the standard frequency like 440 Hz and the pitch of the same is 69. Now, one knows that the notes repeat in a cycle of 12, therefore, this brings us to a pitch 81, which is A5, having the frequency 880 Hz, an integral multiple of 440 Hz (two times). Thus, if for an audio wave, consider that A4 is a fundamental frequency, then A5 is harmony for the same.

6.3 Data Modeling

6.3.1 Fitting Linear Discriminant Analysis

Linear Discriminant Analysis is a classification method similar to logistic regression with a different way to model $P(Y = m|X = x_m)$. Here rather than calculating the probability earlier mentioned, let's try to calculate $P(X = x_m|Y = m)$ and then flip it around using the Bayes theorem [5]. The assumptions involved in this method are:

1. π_k denotes the prior probability that a randomly chosen item from the dataset belongs to the m_{th} class.
2. Let $f_k(X)P(X = x_m|Y = m)$ denote the density function of X for an observation that comes from the mth class. The larger $f_k(X)$ the higher is the probability that a data point (observation) belongs to the mth class.
3. For LDA, $f(x)$ can be assumed of the form

$$f(x) = \frac{1}{(2\pi)^{p/2}|\sum|^{1/2}} exp(-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)) \quad (6.5)$$

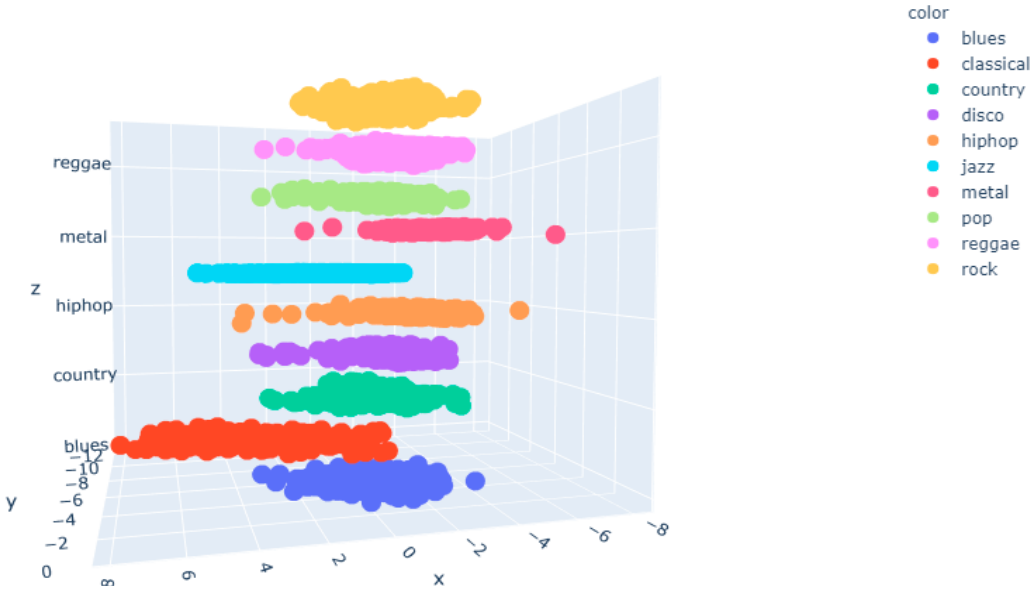
A fundamental assumption for deriving the above equation is that observations come from a MGD(Multivariate Gaussian Distribution). μ_m is the class-specific mean vector and $\sum$ is the common covariance matrix which is common to all M classes.

Now, the function above is substituted into Bayes' theorem and the posterior probability for each observation is calculated to assign the respective class to it [3]. The function more generally used is a rearrangement of posterior probability which can be obtained by taking a log and using little algebra, the following is obtained

$$\delta_m(x) = x^T \sum^{-1} \mu_m - \frac{1}{2}\mu_m^T \sum^{-1} \mu_m + log\pi_m \quad (6.6)$$

The Bayes' classifier assigns an observation $X = x$ to the class for which the above function yields the largest value. The dataset is resolved into 3 components namely 'ld1', 'ld2'and 'ld3', and then the explained variance ratio is calculated in order to observe how much of the total variance is being captured by resolved components. As a rule of thumb one should keep adding components as long as they do not reach an explained variance ratio of > 0.7, in this case, the authors got the explained variance ratio matrix to be [0.47109338 0.21177223 0.10037413], the components add up to 0.78, hence good to go.
Below images show how a complex and intertwined dataset now becomes easily classifiable with distinct borders.

(a) 2-D plot of LD1 vs LD2 in x and y axis respectively

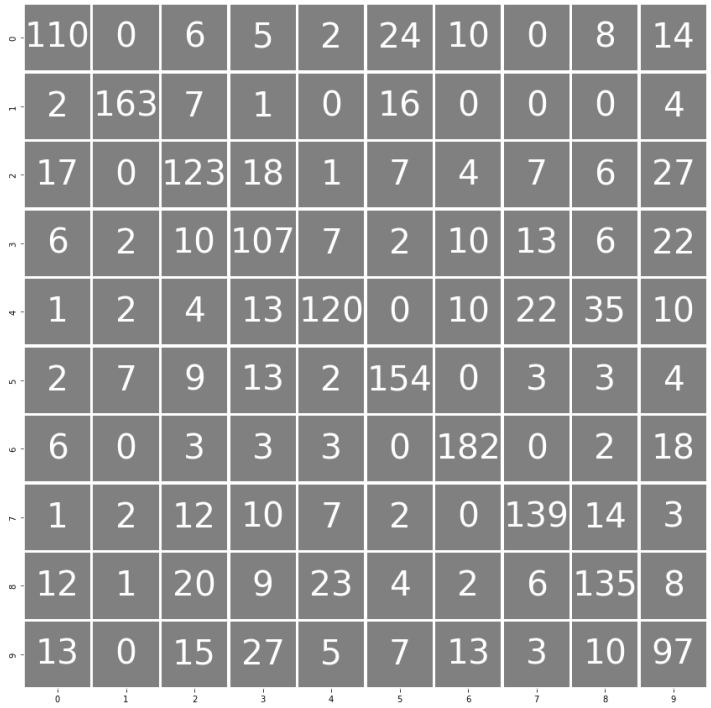


(b) 3-D plot of LD1, LD2, LD3 in x, y and z-axis axis respectively

Figure 6.1: 2d and 3d plots, after resolving it in 2 and 3 components

Upon fitting this LDA on the music genre training dataset, the training set accuracy obtained was 68.4%. However, the training set accuracy is of little importance, the true measure of a model is denoted by the test set accuracy, which is evaluated to be 67%. The test set accuracy obtained,

though not very high, still is acceptable, considering the training set accuracy as the peak. This accuracy is less compared to the complex neural network which can be used for the same classification problem, but the computation cost of the model will be far higher than a simple LDA which gives 67% accuracy. The confusion matrix of the LDA classification is shown below.



Confusion Matrix for the LDA on test set

Figure 6.2: Results on the test set from LDA.

The classes 0 to 9 represent blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock respectively. The result can be improved if the cross-validation set is used for the classification. Upon using the cross-validation set, the training set accuracy increases to 77.4% and the cross-validation set accuracy tends to increase to approximately 68%, which is a slight improvement to the normal training test split. The results of the confusion matrix are shown in Fig 6.2.

The results obtained in the LDA at a glance might not seem very significant and accurate but, if considering the computation costs involved and the results obtained, they tend towards decent results. However, the results obtained from the LDA can still be improved. As mentioned earlier, the LDA is a linear classification algorithm that tends to assume that the covariance matrix is common to all the classes, which may result in a high bias for a model. Therefore, how significantly the model will differ in accuracy if somehow an algorithm is used that tends to reduce the bias by increasing the degree of the classifier. In the coming section, the Quadratic

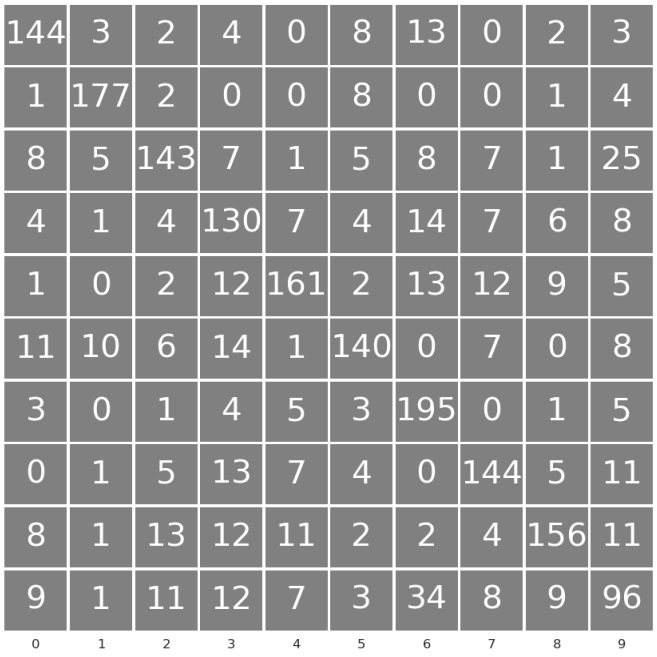
Discriminant Analysis is discussed which gives surprising results. Fig 6.2 underlines the performance of the classifier concerning the actual values of the test set. This will give us a visual intuition of the performance of the LDA algorithm.

6.3.2 Fitting Quadratic Discriminant Analysis

Quadratic Discriminant Analysis works on a similar logic as of LDA, however, there is a fundamental difference between the two [1]. Like LDA, predictions using QDA assume that the data points which are extracted from each class follow an MGD (Multivariate Gaussian Distribution) However explained above, unlike LDA which assumes a common covariance matrix; QDA assumes and calculates covariance matrix for each class, $XN(\mu_m, \sum_m)$ where μ_m and $\sum_m$ are the class-specific vectors and covariance matrix for the m^{th} class. The Quadratic Discriminant Analysis calculates δm(x) and classification occurs to that class for which the entity $\delta_m(x)$ is the highest.

$$\delta_m(x) = -\tfrac{1}{2}(x-\mu_m)^T \sum_m^{-1}(x-\mu_m) - \tfrac{1}{2}log|\sum_m| + log\pi_m \quad (6.7)$$

We performed a similar analysis for QDA as well, the results obtained were as expected, the accuracy of the test set (which matters) and the training set improved drastically. The confusion matrix for QDA is shown in Fig 6.3. The classes 0 to 9 represent the following
0 - blues
1- classical
2- country
3 - disco
4 - hip-hop
5 - jazz
6 - metal
7- pop
8 - reggae
9 - rock



Confusion Matrix for the QDA on test set

Figure 6.3: Results on the test set from QDA.

On fitting the dataset to the Quadratic Discriminant model, the accuracy of the training set shot to 81% and that of the test set came to 74%. This is a huge difference from the LDA which only gave out an accuracy of 68%. Thus, it is safe to assume that the Music Genre Dataset doesn't follow a linear model.

Figure 6.3 underlines the performance of the classifier for the actual values of the test set. This will give us a visual intuition of the performance of the QDA algorithm.

6.3.3 Fitting k-Nearest Neighbors

K- nearest neighbours is a non-parametric classification algorithm [7]. The idea of this algorithm is to assign a class to a new incoming data point by measuring its distance from K nearest points and then deciding its class depending on the plurality of votes of K nearest members. For instance, if $K = 1$ the incoming data point will be assigned to the class of the nearest point. KNN applies Bayes' rule and classifies the test observation to the class with the highest probability.

The distance calculated is Euclidean distance and the following formula can be used to calculate it.

$$d(a, b) = \sqrt{\sum_i (a_i - b_i)^2} \qquad (6.8)$$

What is the most optimal value of K is debatable and has been time and again discussed but since the advancement of processors one can quickly perform a check of Accuracy vs K and decide the most optimal K. Below is a graph showing accuracy on the y-axis and K on the x-axis, K varies from 1-100.

To develop better intuition, let us consider a few values for K. The value of K has to be chosen in such a way that it forms a balance between the complexity, bias and variance of the decision boundary. A very low value of K might result in a function that is highly flexible and overfits the training set, which might result in an excellent performance of the model on the training set but the model might poorly perform on an unknown dataset, the test set. On the other hand, if the value of K is chosen to be very large, this might lead to underfitting, where the decision boundary may not be flexible enough to classify the observations correctly.

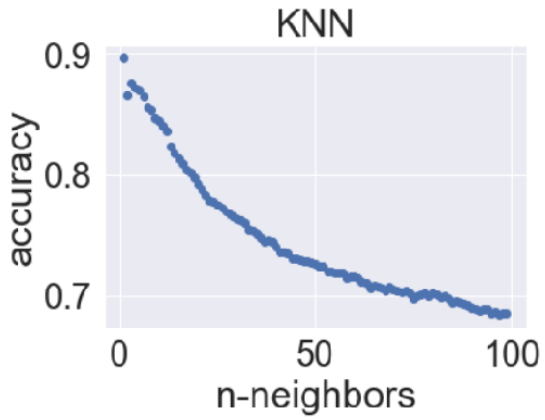
Fitting KNN in the dataset yielded a whopping accuracy of 94.4% on the test set whereas it produced an accuracy of 87.5%.

The dataset chosen in this chapter tends to be one of the most elegant examples to show how beautifully can the KNN perform at low values of K and give us a decision boundary which is close to the Ideal Bayes decision boundary, moreover, performs competitively with the more complex classification algorithms using Neural Networks.

	0	1	2	3	4	5	6	7	8	9
0	165	2	1	1	2	5	1	0	2	0
1	1	182	1	0	0	9	0	0	0	0
2	10	4	174	6	2	2	0	0	7	5
3	3	2	3	164	4	1	0	0	2	6
4	2	0	4	7	187	0	2	5	9	1
5	5	10	4	1	2	170	0	0	1	4
6	0	0	1	6	5	0	201	0	0	4
7	0	2	7	16	3	3	0	154	4	1
8	1	2	4	2	5	0	0	3	202	1
9	4	3	9	10	4	4	4	2	1	149

(a) Confusion Matrix for the KNN on test set

Fig6.4 (a) underlines the performance of the classifier to the actual values of the test set. This will give us a visual intuition of the performance of the KNN algorithm.



(b) Accuracy vs k

Figure 6.4: Results on the test set from KNN.

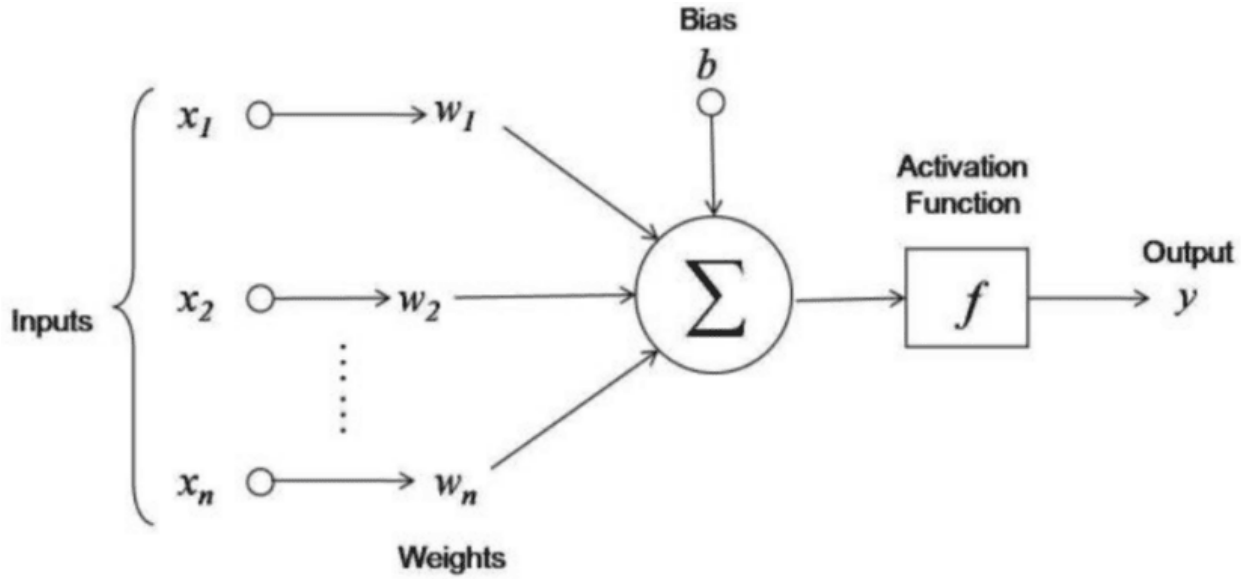
6.3.4 Feedforward Neural Networks

Neural networks form the bleeding-edge research domain for the field of artificial intelligence. These are the quintessential models being used to solve a large spectrum of problems from computer vision to natural language processing. Feedforward Neural Networks or Multilayer Perceptrons form the building blocks of the state of art machine learning models like Convolutional Neural Networks and Recurrent Neural Networks [8]. When trained with appropriate data and hyperparameters, they can perform surprisingly better than any other algorithms known. Deep learning has been used for achieving many ambitious goals like making AI proficient in Chess and Go, improving the performance of self-driving cars, predicting financial markets. They have been realised using a complex system of neural networks. It is of extreme importance to look into this idea to develop a deeper understanding of these algorithms and their architectures.

Feedforward Neural Networks are the culmination of many functions stacked together, giving them the name Networks, they are a kind of mesh of simple nonlinear functions wrapped together to form complex functions [9]. The model is blended with a directed acyclic graph depicting how the functions are formed together, for instance, let's say four functions are chained together in the form $f(x) = f_1(f_2(f_3(f_4(x)))$, these four functions form the most basic

structure of a neural network. Here f_1 is called the first layer, f_2 is called the second layer and so on. This multi-layer sequential structure gives rise to the name deep learning. The final layer of the structure is called the output layer. The fundamental working of a neural network is to evaluate a function $f(x)$ for a dataset such that it is closest to the hypothetical function $f^*(x)$ from which the dataset is assumed to have originated. Each point in the training dataset is used to get approximate $f^*(x)$ and all the points are then used to remove noise to evaluate the most concerning ones. Each point x is labelled with a $y \approx f^*(x)$. The algorithm must itself decide based on the input x and the output y how to use the layers to get the desired output, but the training dataset does not mention what the role of each layer should be, since the training dataset does not tell the layers involved in how to implement the functions, they are called the hidden layers.

Let's now delve deeper into the architecture and mathematics of how exactly a neural network works. The following diagram represents the architecture of a basic neuron and has the following properties



x-input
w-weight
b-bias
f-activation function
y-output
With the initial input x let's calculate z = wx + b
We then apply the activation function upon z to obtain final output y, it's important to note that w, x and b are vectors and hence a dot product of w and x is taken.

Once y is obtained, the value of a cost function is calculated which is basically the sum of the differences between the network's output y and the actual value of y which is taken from the training data set.
The next step then is to minimize the cost function which is done by backpropagation where the partial derivatives of z, w and b with respect to C (cost function) is calculated.
We repeat the whole process until a desired state of accuracy is reached.
With this short description of a neural network, let's dive into the practical implementation of the same for our music genre dataset. Along the way, the authors will keep pausing and give necessary theory wherever needed.

6.3.5 Fitting Feedforward Neural Networks

6.3.5.1 Gradient Descent

The initial setup for making the model stays the same as the previous sections. First extracting the features from the audio file and then splitting the dataset into validation and the training sets. Further, the training set is divided into mini-batches of 128. Using mini-batches is an important step for any neural network. Mini batch gradient descent is the middle ground of the batch gradient descent and the stochastic gradient descent. Here neither the whole dataset is used like in the case of batch gradient descent nor a single data point is used like in the case of stochastic gradient descent, rather the training set is divided into mini-batches of fixed smaller size, generally in the size of exponents of 2. The mini-batch is then fed to the neural network as input and the mean gradient is then evaluated on the mini-batch which is then used to update the weights associated with the neural network. This series of steps is repeated for all the mini-batches for a certain number of epochs to get the optimal result.

6.3.5.2 Activation Functions

Choosing the correct activation function for a neural network is a crucial step. It is important to mention that a neural network has to have a nonlinearity introduced in the system. Failure to do so makes a neural network nothing more than a linear regression function. This is fairly intuitive as the composition of many linear functions is nonetheless a linear function.
There are many possible functions to be the non-linear function required, however, not every function can be used to get the best results. Krizhevsky et al., 2012 extensively mentions the finer details of functions like sigmoid, softmax, tanh, ReLU [9][10][12]. The authors have used the ReLU function for the hidden layers and the softmax function for the output layer. ReLU function $f(x)$ is defined as follows

$$f(x) = max(0, x) \qquad (6.9)$$

$f(x)$ although seems a simple function, has come out to be the first choice of activation function for most neural networks. It works surprisingly well and has been shown in Krizhevsky et al., 2012. The softmax function $\sigma(z)$ is shown as follows

$$\sigma(z) = \frac{e^z}{\sum_{j=1}^{k} e^{z_j}} \qquad (6.10)$$

The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The output of the function is always between 0 and 1 which can be used as a probability score. The input can be positive or negative but the output is always a positive value bounded by 0 and 1. This function seems similar to another function that is known very well, the sigmoid function, right? Often this function is referred to as soft argmax function or multi-class logistic function. This is a generalization of the binary classifier- a sigmoid function that is used as an output layer for binary classification problems. The sigmoid function is the special case of the softmax function.

6.3.5.3 Regularization

It is fairly easy to make a neural network or any model in fact which can perform at 100% accuracy on the training set. However, this might be counterproductive to our needs and has an antithetical result. When the accuracy of a model is increased on a training set, it is trained using the training dataset only. Given the strong computational power it has, it can evaluate weights in a way to accurately fit the training dataset. The statistical condition for a model performing with such high accuracy on the training set is the dilemma of getting a compromise between bias and variance.

First, let's understand the significance of bias and variance in a model and for that, one needs to understand bias and variance. Bias is the difference between the actual value and the predicted value on a training set. The function $f(x)$ which needs to be evaluated, if assumed to be a simple polynomial, would lead to loss of information as the function will not be able to capture the relevant details about the target variable and the input data causing underfitting of the model. On the other hand, if $f(x)$ is assumed to be a very complex polynomial, it will learn the data in a way to fit the function rather than understanding the data which will lead to overfitting of the model. Variance indicates how much the target function for a model would change if the training dataset provided is changed. Naturally, our accuracy of the results is directly related to the estimate of the target function $f(x)$. One would want our model to generalise the problem in the best possible way so that when put to test by the real-world data that our model is not trained on, it gives out accurate results. It would defeat the purpose of the model if it drastically changes the

estimated target function on changing the training dataset. For an ideal model, one should strive to have low bias and low variance.

To avoid overfitting, which turns out to be the more frequent problem, there are many ways to do so. The one more important to us here is the Dropout. It is one of the most used methods of regularization in a neural network model [8][11]. Dropout simply is like the name suggests. It means leaving out or ignoring a few of the neurons or activation units during the training process. Fundamentally, during each stage of training, each neuron can be kept with a probability of p which conversely can be stated as that each neuron can be dropped off with a probability of $1 - p$. This brings randomness in the model and in each stage of training a different pair of neurons are activated, compelling the model to make use of all the units and understand the dataset rather than learning the dataset. Thus dropout forces the model to learn more robust features that help in making more accurate predictions.

6.3.5.4 Model Building

After the data processing and selecting the relevant activation functions, comes the task of aligning them in a sequence to get the best possible results. Each layer is followed by a dropout layer with a value of 0.3 and there are 5 such layers which are ultimately followed by an output layer. The first layer has 1024 neurons, the second layer has 512 neurons, the third layer has 256 neurons, the fourth layer has 128 neurons, and the fifth layer has 64 neurons. The activation function used in all the previously mentioned layers is the ReLU function defined above. The final layer is having 10 neurons, corresponding to the ten genres of music classification the authors are performing. It uses the softmax function. The model runs for 700 epochs and gives a fantastic accuracy of 93% on the test set. The results of the test set have been mentioned in the confusion matrix shown in Figure 6.6.

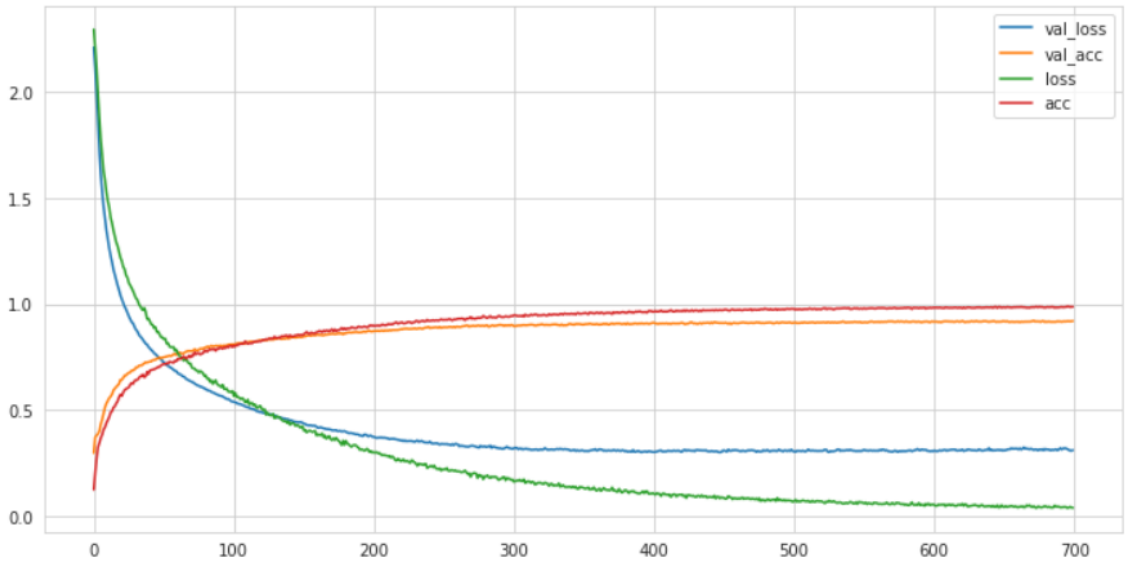


Figure 6.5: Results of the Neural Network Model

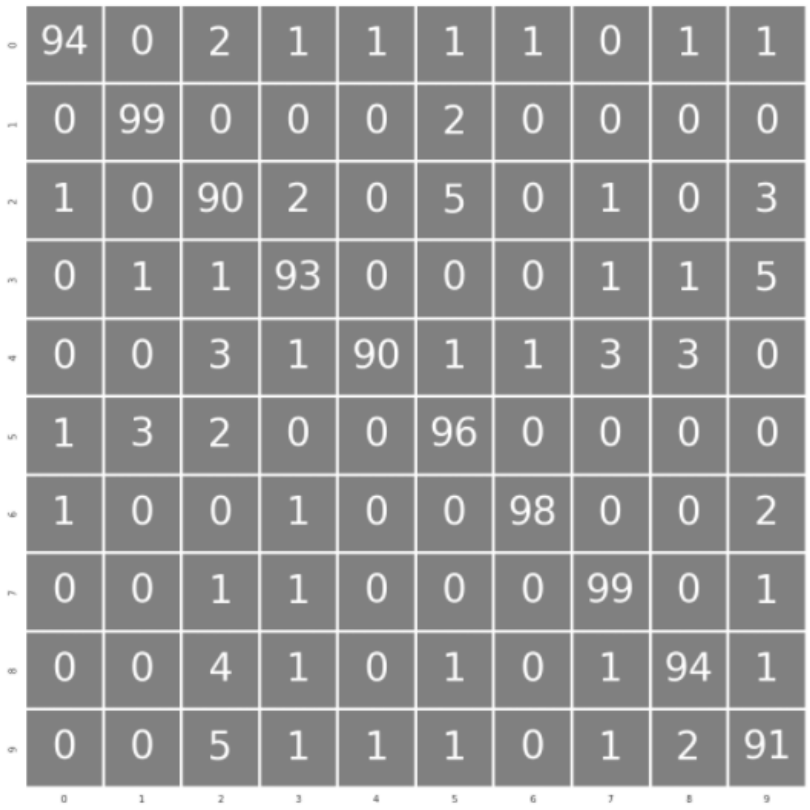


Figure 6.6: Confusion Matrix for the test set on Neural Network

It is rather fascinating to see a graphical representation of the results obtained in a complex model such as this. Readers are encouraged to always follow a graphical approach to understand the working of the model as this provides a better understanding and intuition of the model. Now let's observe Figure 6.5 to make some reasonable observations about the model. As one sees, the cross-entropy loss(green) associated with the model keeps on decreasing with every epoch signifying that the model is working and learning in the desired direction. However, this does not tell us that the model is well and good. This is only confirmed by observing the loss of the validation set(blue). The validation loss goes in the same direction as the green line. The value of validation loss seems to be greater than the training loss, and when looked at carefully, it makes sense. The validation set is simultaneously used during training for monitoring the learning of the model, as that is the kind of a pseudo test set on which the model evaluates its performance after being trained on a batch of training data, therefore that data is relatively new for the model, hence the increased loss value. The same thing can correspond with the accuracy(red) and validation set accuracy(yellow). Implementation of the proposed model with different variations can be found in GitHub[13][14].

6.4 Conclusion

After applying all the three fundamental classification algorithms, the authors have a very conclusive result. The performance of the algorithm can be arranged in the following order LDA<QDA<KNN<Neural Networks. The Neural Network gives us the best results of about 93% accuracy, which can be considered very high based on the simplicity of the network.

LDA is closely related to logistic regression. It is a parametric classification algorithm that assumes the decision boundary to be linear, the only difference between the two is in the terms of the number of classes in which both of them are classified. Logistic Regression works best for the cases in which the number of classes to classify is two whereas, LDA can serve for the linear boundary of more than two classes, KNN is a non-parametric algorithm that makes no assumptions about the shape of the decision boundaries. QDA serves as an intermediate classification model between linear LDA and non-parametric KNN. QDA assumes the decision boundary to be quadratic which works in this case because it serves as the equivalent point between the bias and the variance. It tends to settle the conflict between the low bias caused due to LDA and the high bias caused due to KNN.

This conversation, however, changes with the introduction of models as complex as neural networks. Neural networks have proved to be very effective and most importantly accurate at complex tasks that can only be dreamt of using the classical methods at hand. The interesting part is that a neural network is a kind of black box. One can fine-tune it to give very accurate results but the stage of progress can not be known and it is an area of research. Few methods are used by which one can get a rough idea as to what has been the process of learning till that step but that is still far from being called a definitive method. Nevertheless, neural networks are turning out to be a very effective tool for a variety of problems and they have reached a level where they can outperform humans. An example is image classification. There are networks like ResNet, GoogleNet, VGGNet, that have outperformed humans in the classification task. Amidst all the good characteristics of the neural network, there are a few problems associated with them. The biggest one is the computational complexity of neural networks. If not fine-tuned in the most efficient way, they pose a serious challenge in training them. To get an intuition about the same, our fairly simple neural network that on a practical scale, takes a small dataset, calculates over 757,000 parameters to train the model and this took a fair amount of time. Therefore, one needs to be very careful with the implementation of a neural network.

From the analysis of a music dataset for over more than one thousand tracks, one can conclude that the music tracks have very distinguishing properties which allow them to form a very well defined cluster. This observation is the key reason why the Neural Network classification approach performs so well even though it is one of the simplest classification algorithms in

classical statistics. It will be an interesting problem to see if the same approach can be applied to the genres of Indian Classical music. The hypothesis that the authors currently have suggests that it can be pretty difficult to classify Indian classical music as they possess a lot of similarities in the way it can be perceived by the listeners. However, this will be an interesting topic to study.

Bibliography

[1]B. L.Sturm, "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use", arXiv preprint arXiv:1306.1461, 2013.
[2]Jihae Yoon, Hyunki Lim, and Dae-Won Kim, "Music Genre Classification Using Feature Subset Search," International Journal of Machine Learning and Computing vol.6, no. 2, pp. 134-138, 2016.
[3]Atti, Andreas Spanias, Ted Painter, Venkatraman (2006), “Audio signal processing and coding”,Hoboken, John Wiley & Sons.
[4]Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014
[5]https://stats.oarc.ucla.edu/spss/dae/discriminant-function-analysis
[6]James, Gareth, et al. “An Introduction to Statistical Learning with Applications in R”, Springer, 2017.
[7]Flederus, D.R. (2020), “Enhancing music genre classification with neural networks by using extracted musical features.”
[8]B. L. Sturm, "On music genre classification via compressive sampling," in Multimedia and Expo (ICME), 2013 IEEE International Conference on (pp. 1-6), IEEE, July 2013.
[9]Ian J. Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016.
[10]V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In Proc. 27th International Conference on Machine Learning, 2010.
[11]G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012
[12]Krizhevsky et al., 2012 “ImageNet Classification with Deep Convolutional Neural Networks”
[13] https://github.com/divydvd
[14] https://github.com/AshutoshGanguly123/Classification-Algorithms.git