NTNU
Kunnskap for en bedre verden

Department of Electronic Systems

TTT4275 - Estimation, detection and classification

Music genre classification project

Classification group 13:
A. Å. Pedersen
K. S. Grøtterud

April, 2022

Abstract

This report describes the implementation and results of a k-Nearest-Neighbors classifier and a shallow neural network for music genre classification, as a part of the course TTT4275. The code was implemented in Matlab.

The classifier performance was measured by creating confusion matrices and calculating error rates for each separate experiment. First, the results showed that the selection of the features for the music tracks made a significant impact on the classifier performance. Moreover, the number of features used made an even bigger impact. This shows that providing more information, in terms of the number of features, generally gives better performance.

When applying kNN ($k = 5$) with four features, the error rate was initially 68.2% which improved to 63.6% when exchanging the least separable feature with the most separable feature among all 63 features. When applying kNN ($k = 11$) with all features, the error rate became 34.8%.

To further improve the performance, a shallow neural network was implemented. The performance improved to an error rate of 22.7% which was our best result obtained in this project.

Distinguishing between similar genres proved to be a challenge for all the stages of this project. The lack of unique genre characteristics led to poor performance in the first parts of the task. However, this was expected as music genres are rather ambiguous making it hard to construct a mathematical relation between features and genres.

Table of Contents

1 Introduction

This report covers the *Music genre classification* project in the course *TTT4275 Estimation, detection and classification*. The report examines how different parameters influence the performance of the k-Nearest-Neighbors classifier and how it compares to a self-selected classifier, namely a shallow neural network. This shows to be quite relevant for real-life applications. Throughout the world around us, classification is a big part of technology and our daily lives. The decisions we make daily involve classification, and teaching computers to do the same is valuable. Incoming spam emails in your inbox must be filtered out. Autonomous vehicles must be able to comprehend their surroundings in order to drive safely. It is clear that creating accurate classifiers is necessary in today's society to streamline manual tasks and enable further technological progress.

In the following sections, we will build intuition and background in classification theory, before presenting our implementation and findings. First, the theory relevant to this project is described in Section 2 before the task is presented in Section 3. The implementation and results are shown in Section 4, which is divided into the two main parts of the project: the kNN classification, and the neural network. Finally, the conclusion is presented in Section 5.

2 Theory

The theory section will briefly introduce topics in classification that are necessary to understand the implementation and results of this project. First, a general introduction to classification will be given. Then the classification techniques applied in this project will be presented. Finally, the techniques used to measure classifier performance are pres

2.1 Classification

Classification is a method for separating sets of data into classes based on their features. This could be as simple as determining if a person is male or female based on their shoe size, or more complex like assigning a diagnosis based on observed symptoms of a patient. The difficulty of a classification problem will among others depend on how separable the features are and the amount of available training data.

2.2 Preprocessing of data

Normalizing the data at hand is important to ensure that every feature of the data has the same range. This ensures that each feature weighs the same, so that no feature influences the classifier more than others. With *z-score normalization*, the data is scaled to have a zero mean, $\mu = 0$, and a standard deviation of $\sigma = 1$. Another way of normalizing data is the *min-max* method. In this method, the data of each feature is scaled to fit within the range $[0,\ 1]$. The scaling of the features is calculated as follows

$$\text{Z-score: } x_i' = \frac{x_i - \bar{x}}{\sigma}, \qquad \text{Min-max: } x_i' = \frac{x_i - min(x)}{max(x) - min(x)}$$

where x_i is an element of feature x, x_i' is the scaled version of the feature, $\bar{x}$ is the mean value of x, and σ is the standard deviation of x.

2.3 Nearest Neighbor classification

In nearest neighbor classification, the distance between an observation and a set of reference data is computed. The observation will be assigned the same label as the one of the closest reference.

An improved version of the nearest neighbor classifier looks at the k nearest neighbors (kNN), instead of only the nearest one. The principle is illustrated in Figure 1. The observation is classified based on which class makes up the majority among the k nearest references. If there are multiple classes that make up the majority vote, one of them must be selected based on a chosen decision rule. In Figure 1, with $k = 3$, the sample would be classified as a member of *Class B*, but with $k = 5$, it would be classified as a member of *Class A*.

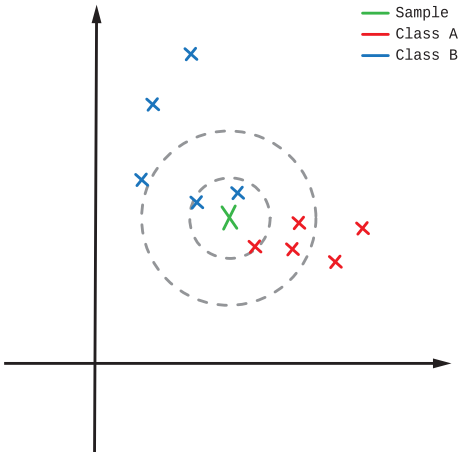


Figure 1: kNN classification with $k = 3$ and $k = 5$.

The distance measurement will impact the results of the kNN classification. There are several ways of defining distance, so the choice of distance measurement must be done with respect to the application. One of the most common distance measurements is **Euclidean distance**. For a Cartesian coordinate system of N dimensions, the Euclidean distance between the points p and q is defined as

$$d(p,q) = \sqrt{\sum_{i=1}^{N}(p_i - q_i)^2} \tag{1}$$

2.4 Neural networks

Neural networks classify data by mimicking the behavior of the human brain. The network is divided into several layers of abstraction, where the first layer has a neuron for each feature and the last layer has a neuron per class. Between the input- and the output layers are the "hidden layers". The depth of the hidden layers distinguish a shallow network from a deep network.

All nodes are connected to every node in the previous layer. Each connection is given a weight used to compute a weighted sum which is then used as input in an activation function that outputs the activation number. If the activation number exceeds a given threshold, the data is passed to the next network layer. Finally, the activation number of each neuron in the output layer indicates how well the system thinks a given input agrees with the corresponding class.

Equation 2 shows how the activation number of the first node in layer m is computed, where $\omega_{m-1,i}$ is the weight assigned to the connection between the respective node and the nodes $i = 0, 1...n$ in the previous layer $m-1$, and b_i is a bias. The bias is used to decide the threshold of which the node fires off. Training a neural network refers to tuning the weights and biases so that the system solves the problem at hand.

$$a_i^{(m)} = \sigma(\omega_{m-1,0}a_0^{(m-1)} + \omega_{m-1,1}a_1^{(m-1)} + ... + \omega_{m-1,n}a_n^{(m-1)} + b_i) \tag{2}$$

Equation 3 displays the logistic Sigmoid function which is a common activation function.

$$\sigma(x) = \frac{1}{1 + e^{(-x)}} \tag{3}$$

2.5 Performance evaluation

The performance of a classifier is measured by computing a *confusion matrix*. The confusion matrix rows indicate the target classes of the test set, while the columns indicate the classes predicted by the classifier. An example of a confusion matrix is given in Figure 2. The goal is to get the main weight of the matrix on the diagonal, as this indicates correct classification. This allows for analysis of potential biases of the classifier, and gives an overall impression of the classifier performance.

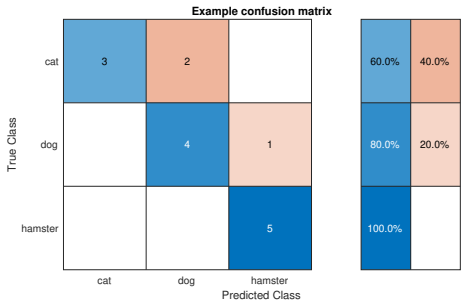


Figure 2: Confusion matrix for pet classification

Furthermore, the *error rate* can provide a picture of the total performance. Error rate is computed by dividing the number of wrong classifications by the total number of observations as shown in Equation 4.

$$err = \frac{\text{number of wrong predictions}}{\text{total number of observations}} \cdot 100 \tag{4}$$

The target error rate of a classification problem varies with the application. In safety critical systems such as perception systems for autonomous cars, the error rates must be low, while less critical systems can accept a higher percentage.

3 The task

The task of this project is to classify the music genres of given data collected from audio tracks. In total, there are 10 genres represented in the data set: pop, metal, disco, blues, reggae, classical, rock, hip hop, country, and jazz. The 63 model features in the data set are the mean and/or standard deviation of commonly used audio features. The data set of 1000 30-second audio tracks are divided into a training set of 800 tracks, and a test set of 200 tracks. There is also given a 10-second data set and a 5-second data set where the tracks are components of the 30-second tracks.

In the first part of the project, a k-Nearest-Neighbors (kNN) classifier is designed to classify the music genres of the test set. A given set of four features is used in the classifier. In the second part, the overlap between the genres pop, disco, metal and classical is examined for the four features. Furthermore, one by one feature is removed from the classifier, and the performances are compared. Next, one of the four given features is replaced with a feature of our choice. Finally, a new self-chosen classifier is designed, with as many features and as much data as we see fit.

In all parts of the project, the performance of the classifier is evaluated using a confusion matrix along with the error rate.

4 Implementation and results

In this section, the implementation of the project will be presented, and the results will be discussed. First, the utility functions used in all parts of the project will be briefly explained. Next, the first three parts of the project will be discussed as they all use kNN classification. The final part discusses the implementation and results of a neural network and how this compares to kNN.

4.1 Utility functions

The utility functions were implemented for code readability. Table 1 provides brief explanations of their implementation.

`dataExtraction(filename):`	Extracts the numerical data from the file, and splits it into a training set and a test set on the format [features, label] for each row in the data file.
`normalizeSet(dataSet):`	Performs column-by-column *z-score normalization* using the Matlab function `normalize(X)`, and returns the full data set normalized, and with each reference's label in the final column.
`getErrorRate(predictedLabels, trueLabels):`	Calculates the error rate of the prediction.

Table 1: Utility functions.

4.2 Music genre classification with kNN

The kNN algorithm was implemented to classify observations. A flow diagram illustrating the algorithm is shown in Figure 3. The implemented kNN function `knn(k, observation, trainingSet)` follows the principles presented in Section 2.3. If there are multiple majority classes among the k nearest references, the class closest to the observation is chosen. The Euclidean distance was found using the Matlab function `dist(X, Y)`.

Figure 3: kNN algorithm.

In the first part, the four given features *spectral_rolloff_mean*, *mfcc_1_mean*, *spectral_centroid_mean* and *tempo* were used along with a kNN classifier with $k = 5$ to classify the test set of the 30 second music track data. The classification was done by iterating through the test set, running the `knn` function for each iteration and storing the resulting labels. The resulting confusion matrix is shown in Figure 4 and gave an unsatisfactory error rate of $err = 68.2\%$. From Figure 4, it is clear that some genres stand out in terms of the misclassification percentage. For instance, the blues tracks were labelled more as jazz. According to Wikipedia 2022a, jazz is characterized by, among others, *blue tones* which is a trait shared with the blues genre. This could explain the frequent wrong classification of blues tracks. Furthermore, country was also misclassified frequently as jazz. The trend of the classifier being somewhat jazz-biased can be explained by the fact that jazz contains elements from many genres, and a lot of improvisation, which can lead to a lot of atypical data in the jazz training tracks. In addition, rock is often misclassified as metal, which again makes sense since metal is a sub-genre of rock music, according to Wikipedia 2022b.

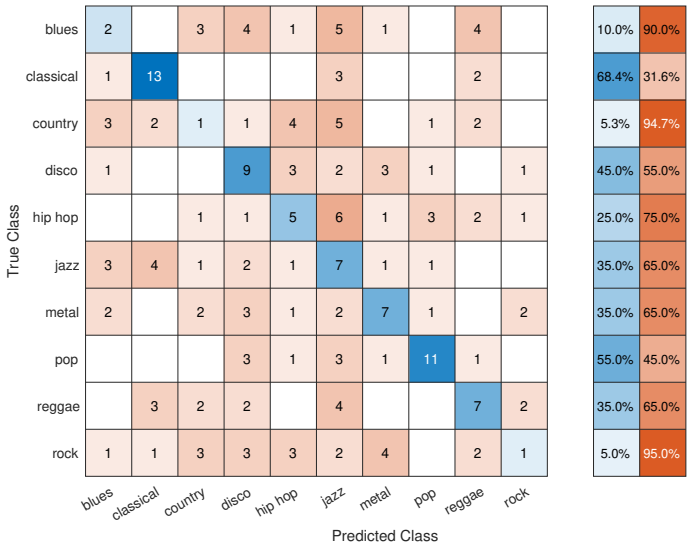


Figure 4: The initial confusion matrix. $err = 68.2\%$.

The second part of the task consists of comparing the distribution of the four features for the classes pop, metal, disco and classical. Furthermore, one by one feature was removed from the classification starting with the least separable one.

Figure 5 shows the distribution of the features for the four given genres. From this, it is clear that both *tempo* and *mfcc_1_mean* overlap significantly. *Tempo* gives the most overlap for all genres, while *mfcc_1_mean* overlaps for all genres except classical. The remaining two features also overlap some, but their mean values are more separated.

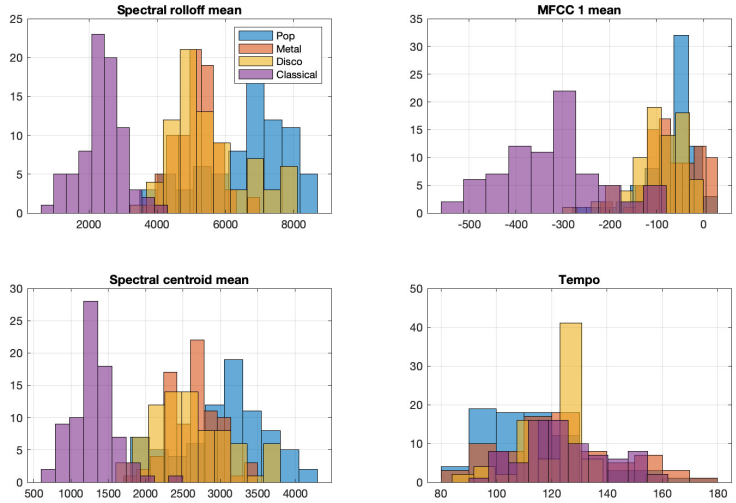


Figure 5: The overlap between the selected genres for the four features.

Furthermore, the scatterplots in Figure 6 show the correlation between the four features. The plots agree with the histograms in terms of the overlap between genres. When studying the plots containing *tempo*, the data seems almost random as there is no clear pattern along the y-axis. All genres appear to be distributed between 80 and 180 bpm, but centered around 120 bpm, independently of the feature on the x-axis. *Tempo* shows almost no correlation with the other features. The plots not containing *tempo* all have a higher correlation, and follow a more specific pattern where the points for each genre are more grouped together. These plots show a positive correlation between the features.

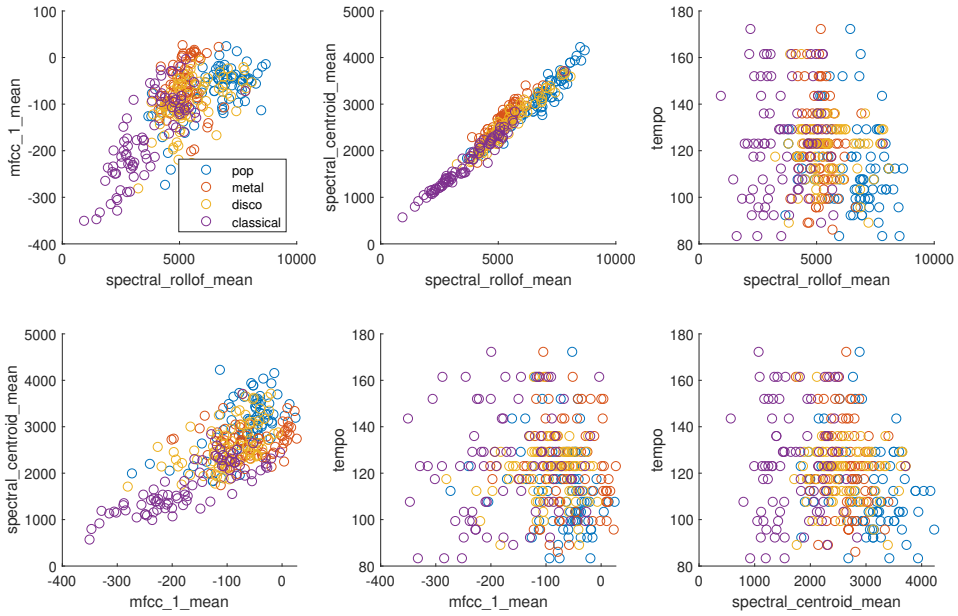


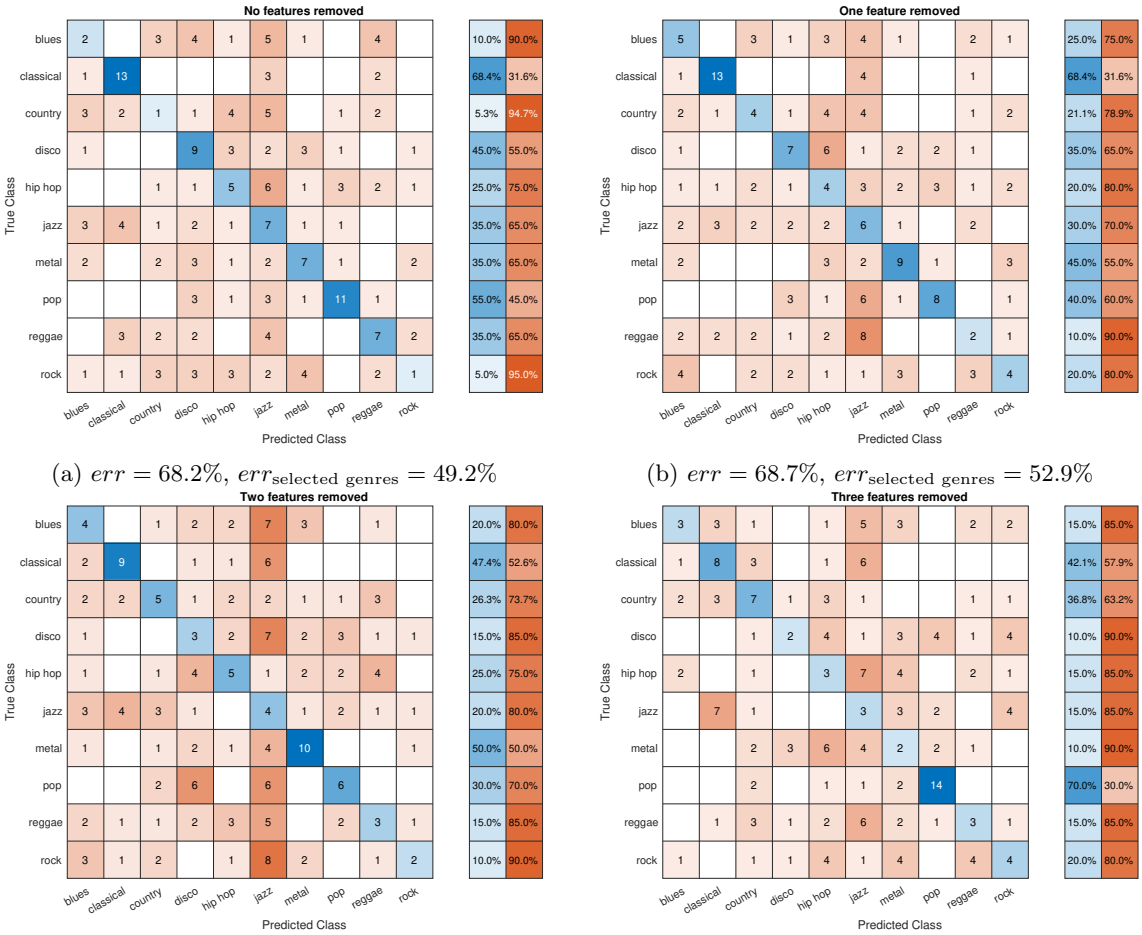
Figure 6: Scatter plots for pairs of features.

The classification was done removing one by one feature, first removing *tempo* which we identified as the most overlapping one, then removing *mfcc_1_mean*, then *spectral_centroid_mean* resulting in classifying with only *spectral_rolloff_mean* in the end. As seen in Figure 7, the trend is that the performance of the classifier gradually becomes worse as the features are removed. This is expected since all of these features, even the most overlapping ones, provide some information about the classes and can give higher accuracy. However, we see that the increase in error is the smallest when removing *tempo* which can be explained by the significant overlap between the genres as mentioned earlier. The rest of the features overlap less, making the consequence of removing them more significant as the data that they contribute with gives more insightful information than *tempo* does.

These conclusions resonate well with the practical meaning of the features. For example, the *tempo* only represents the audio track's beats per minute which can vary a lot within a genre. Hence, *tempo* is not sufficient to define a genre. On the other hand, *spectral centroid* is harder to comprehend. It represents the frequency that the energy of the sound spectrum is centered upon [Librosa 2022]. Although it is harder to visualize this as opposed to the tempo of a song, one can imagine that the energy of a musical track is higher in a metal track than in a reggae track for instance. We can in fact see that they never get classified as each other until the *spectral centroid* feature is removed as seen in Figure 7.

With linearly separable features for each genre, the performance of the classifier would undoubtedly be improved. The scatter plots in Figure 6 show that they are clearly not, which limit the possibilities for classifier accuracy. Moreover, it is important to stress that getting linearly separable features for music tracks is hard. Music genres are not explicitly defined in the same way as cats and dogs, or the digits 0-9. They can be described more like a spectrum rather than discrete classes. As a result, an accurate music classifier will be challenging to design no matter the amount of available features.

In the third part of the project, one of the four earlier used features is replaced before classifying the data with the kNN classifier and generating the confusion matrix.
From the histogram- and scatter plots in part 2, it is clear that tempo is the least separable feature. The replacing feature is chosen by calculating the mean for every genre within each feature. Then



(a) $err = 68.2\%$, $err_{\text{selected genres}} = 49.2\%$

(b) $err = 68.7\%$, $err_{\text{selected genres}} = 52.9\%$

(c) $err = 74.2\%$, $err_{\text{selected genres}} = 64.4\%$

(d) $err = 75.3\%$, $err_{\text{selected genres}} = 67.0\%$

Figure 7: The resulting confusion matrices from excluding one by one feature.

each feature is given a score based on the distance between each mean, where the score is given as

$$score = \sum_{i=1}^{N} \sum_{j=1}^{N} |\bar{x}_{i,j} - \bar{x}_{j,i}|, \quad i \neq j, \quad N = \text{number of genres} \tag{5}$$

Equation 5 was implemented as the function `scores(trainingset)`. Using the `scores` function on all features, `spectral_centroid_var` was found as the feature with the highest score. The histogram of `spectral_centroid_var` is shown in Figure 8. Compared to Figure 5 it is clear that `spectral_centroid_var` is significantly less overlapping than `tempo`.

Ultimately, the data was classified with the kNN classifier and the features `spectral_rollof_mean, mfcc_1_mean, spectral_centroid_mean` and `spectral_centroid_var`. The result is displayed in the confusion matrix in Figure 9. The total error rate was $err = 63.6\%$ which is an improvement from $err = 68,2\%$ with `tempo`. Given the above it can be concluded that four features do not provide enough data for the classifier to perform satisfactory.

4.3 Music genre classification using a Neural Network

To further improve the performance of the classifier, kNN was applied with $k = 11$, and using all 63 features. In addition, the data from the 10 second music tracks was used together with the 30 second data in the training set. The 5 second data was omitted since the tracks are so short that they may contain a higher concentration of atypical data for their genre.

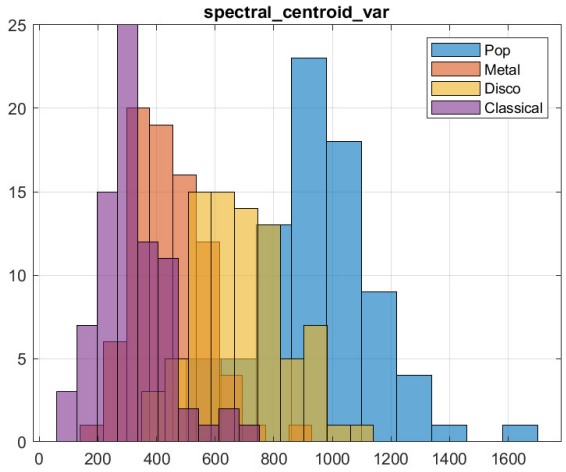


Figure 8: The histogram of the least overlapping feature, according to the `scores` algorithm.

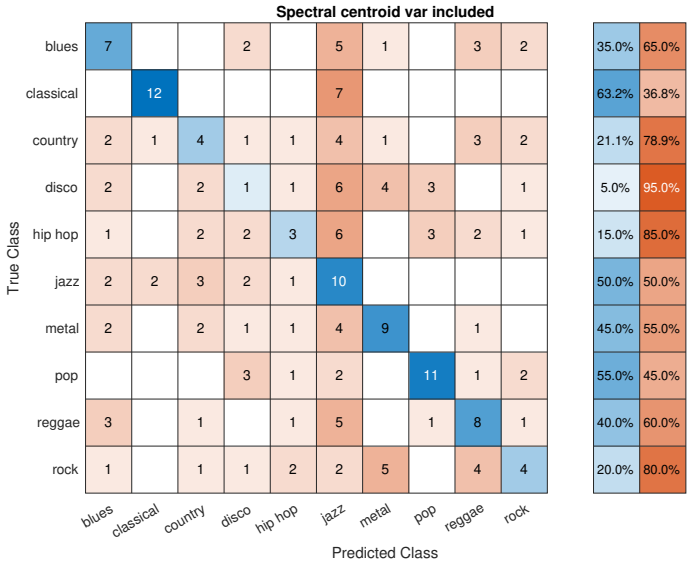


Figure 9: The confusion matrix after excluding tempo and including the best feature. $err = 63.6\%$.

As seen in Figure 10a, the performance was significantly improved from the previous parts, although an error rate of 34.8% is still not a satisfying result. A few of the genres have overlap with other genres, and are therefore often misclassified based on the given features. To improve the performance further, a more complex classifier is needed. The lack of separability demands a smarter way to recognise the patterns of the data set. Therefore, a neural network was implemented to attempt to push the error rate below a target error rate of 25%.

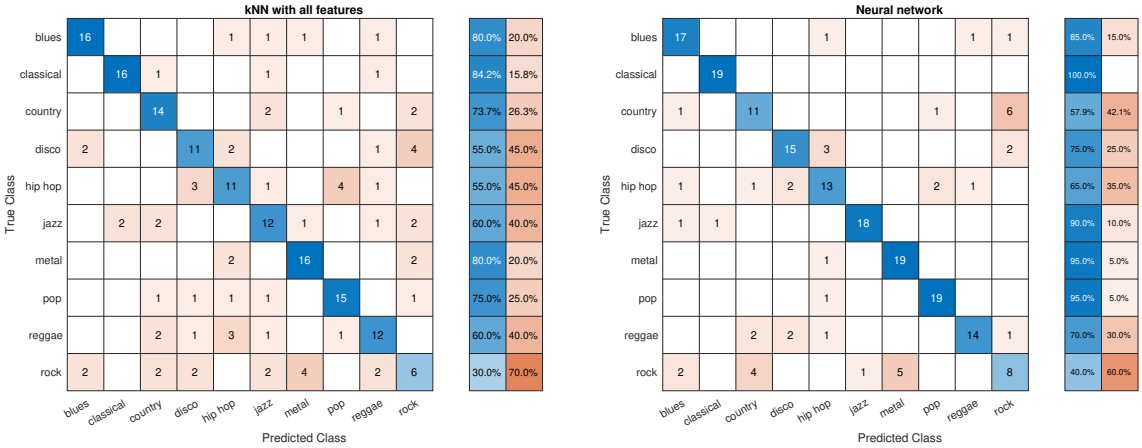
The neural network was implemented using the Matlab function `feedforwardnet`. The features and labels of the training set were then fed into the network using the `train()` function which uses the training set to tune the parameters of the `feedforwardnet`. The `feedforwardnet` was applied with a single hidden layer of 52 neurons, and the Scaled Conjugate Gradient (SCG) algorithm. This algorithm was selected due to the structure of our problem, where there are multiple inputs and outputs. Here, the inputs are the 63 features, and the outputs are the 10 genres. This structure seemed to match well with the example problem `CHOLESTEROL` found in Mathworks 2022 where the SCG algorithm gave the best classification results. Hence, the SCG algorithm was selected. The size of the hidden layer was selected based on one of many existing rules of thumb. We decided on the rule

$$n_{\text{neurons}} = \frac{2}{3} n_{\text{inputs}} + n_{\text{outputs}} = 52$$

as given in Heaton 2008.

The test data was labelled using the generated network. The output from the network when applied to the test data is a set of activation numbers for each genre, for each sample. The maximum valued number indicates the best fitting genre for the given sample, hence the sample is classified accordingly.

The result from the implemented neural network is presented in Figure 10b and shows a great improvement from the previous parts with an error rate of 22.7%. This error rate is well within the target of 25%.



(a) kNN with $k = 11$. $err = 34.8\%$

(b) Neural Network. $err = 22.7\%$

Figure 10: kNN versus Neural Network using all 63 features.

Like before, rock stands out as the most challenging class to label as it has a lot of overlap with other genres. This shows that even with a complex and accurate classifier, some of the music tracks that resemble multiple genres will be wrongly classified. However, the genres which performed the best earlier have improved even more. This indicates that the genres which have distinct characteristics are simple for the neural network to classify, such as classical which has a 100% hit rate.

One of the advantages to the neural network is that the computation time of classification is quite fast compared to kNN, once the model is trained. This provides more efficiency, though at the expense of complexity. One can discuss whether this level of complexity is needed when attempting to classify music genres, but reverting to kNN will make the error rate worse. In this case, as in many others, there is a trade-off between complexity and accuracy. We drew the line at a shallow neural network as going deeper seemed rather excessive.

5 Conclusion

This report has covered the implementation and results of a kNN classifier and a shallow neural network to classify music tracks, as a part of the course TTT4275.

The kNN classifier provided a wide range of performance based on the features included in the training- and test sets. With $k = 5$ and the four given features *spectral_rolloff_mean*, *mfcc_1_mean*, *spectral_centroid_mean* and *tempo*, the error rate came out at 68.2%. The most heavily misclassified tracks belonged to the genres blues, country and rock. The blues and country tracks were often classified as jazz which can be explained by the fact that jazz contains elements from many different genres. Furthermore, metal is a subgenre of rock, making it natural that some rock tracks may be wrongly classified as metal.

When gradually removing features from the classifier, the performance worsened accordingly. This was expected since all features provide some information, though with varying quality. The *tempo* feature showed the most overlap between the genres pop, metal, classical and country and was

therefore removed first, resulting in the error rate 68.7%. This is only marginally worse than before, probably due to the amount of overlap between the genres for this feature. Then, *mfcc_1_mean* was removed giving an error rate of 74.2%. Finally, *spectral_centroid_mean* was removed, leaving *spectral_rolloff_mean* as the only feature to classify the test set. This gave an error rate of 75.3%.

The most overlapping feature, *tempo*, was replaced with the one found to be the least overlapping. The separability of each feature was calculated using the distance between the means of each respective feature for each genre. This method found that *spectral_centroid_var* was the least overlapping. When exchanging *tempo* with *spectral_centroid_var*, the error rate became 63.6% which is an improvement from the initial classification.

Finally, a kNN classifier with $k = 11$ using all features was implemented, leading to an error rate of 34.8%. Although better than the previous error rates, this could be further improved. A shallow neural network with a single layer of 52 neurons was implemented, resulting in an error rate of 22.7%. While kNN only focuses on the distance between

Throughout all experiments, the tendencies of overlapping genres propagates to the classifier performance, especially of rock. It is important to stress that music genres are not discretely defined, but rather a spectrum with overlapping genres and no clearly defined bounds where one genre stops, and another one starts. Without linear separability, the classification problem immediately grows in complexity. The level of complexity of the given data exceeded what we could manage to achieve with kNN. However, the neural network was able to do the job well in comparison, though not perfectly.

Bibliography

Heaton, Jeff (2008). *Introduction to Neural Networks for Java*. Accessed: 2022-05-01. Heaton Research.

Librosa (2022). *Feature extraction*. URL: https://librosa.org/doc/main/feature.html#spectral-features (visited on 29th Apr. 2022).

Mathworks (2022). *Choose a Multilayer Neural Network Training Function*. URL: https://se.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html (visited on 1st May 2022).

Wikipedia (2022a). *Jazz*. URL: https://en.wikipedia.org/wiki/Jazz (visited on 29th Apr. 2022).

— (2022b). *Metal*. URL: https://en.wikipedia.org/wiki/Heavy_metal_music (visited on 29th Apr. 2022).