Music Genre Classification

I. Problem Description

Music genre classification is widely discussed in the MIR (Music Information Retrieval) Society and has been of interest for a variety of reasons, including management of large music collections. As these collections are becoming more digital, genre classification is essential for creating recommendation systems, that can benefit collective music libraries, social networks, etc. Only limited agreement is achieved among human annotators when classifying music by genre and an automatic genre classifier aims to overcome those limitations by defining a standard for extracting features directly from the audio signal for classification. It is important that we extract the most robust set of features and establish a reliable ground truth for classifying into genres.

II. Related Work

Talupur et al. [1] describe a neural network approach that is trained for the classification tasks to determine the genre of audio files. Feature vectors are extracted using Fourier Transforms and fed into a neural network classifier. The algorithm - a linear vector quantization network - in this paper can only distinguish between four genres. Galkin et al. [2] also use neural networks for classification, specifically feed-forward neural networks. Their approach is limited to three genres as classification accuracy would drop significantly below the mentioned 72% if that number was increased. Tzanetakis et al. [3] apply Gaussian classifiers and Gaussian mixture models. They present a hierarchy of musical genres and an elaborate section on feature extraction. Yet their classification results in only 61% accuracy over ten genres. Salamon et al. [4] describe an approach using high-level melodic features for their classification. Various algorithms are compared including support vector machines, random forests, k-nearest neighbour networks and Bayesian networks. Recognition rates of over 90% are reported. This approach though requires the existence of a melody in an audio file, which is not the case for all genres.

III. Implementation

The main purpose of this project was to implement different classification algorithms and compare their performance when applied to a practical problem. Specifically, we performed music genre classification of songs on a dataset containing 150 songs. The three algorithms we used for classification were support vector machines (SVM), k-nearest neighbor (k-NN) classification, and decision trees (DT).

In general, three steps have to be performed to apply a classifier and evaluate the quality of the results:

1) *Data collection:* Retrieve a feature set for a representative sample of music.
2) *Classification:* Run the classification algorithm on this dataset.
3) *Validation:* Confirm the validity of the results using cross-validation.

In this section, we will describe each of the three steps in detail, specifically data collection and the three classifiers. All algorithms outlined in this section were implemented in Python mostly because interfacing with the song databases could be done most conveniently with the provided Python APIs and the number of machine learning toolkits available for Python.

A. Data Collection

The dataset we fed into our classification algorithms was a combination of two datasets available online. We retrieved a list of songs divided into 5 genres from [5] (henceforth called the Gettysburg dataset) and retrieved relevant features for that song selection from the Echo Nest [6] database using the provided Python API. The Gettysburg dataset not only provides the song names and their ground truth classification into genres but also local feature descriptors called MFCC (Mel Frequency Cepstral Coefficients). These are very low-level features that describe the song segment by segment resulting in a feature matrix of size n by m (where n is the number of segments and m the number of features per segment). Instead of using such a high number of features, we instead only used global features in our classification. Specifically, we selected ten features: time signature, energy, liveness, tempo, speechiness, mode, key, duration, loudness, danceability (as defined in [6]).

B. Classification

We used the Machine Learning Toolkit (Milk, see [7]) to implement our classification framework. Milk provides basic implementations for most machine learning algorithms including SVMs, k-NNs, and DTs. We conditioned our dataset to fit the required input format for Milk and extended the algorithms to perform multilabel classification.

*a) **Support Vector Machines**:* Support vector machines are a popular tool in supervised learning if no prior knowledge about the domain is available. As mentioned in [8], three properties make SVMs attractive:

1) *Maximum Margin Separation*: SVMs compute the decision boundary in such a way that the distance to the closest datapoint on either side is maximized, which helps SVMs to generalize well.

2) *Kernel Trick*: Generally, SVMs create linear hyperplanes to separate data, but not always can such a linear separator be found in the original input space. The original data can be mapped to a higher-dimensional space using kernel functions. In this space, data is more likely to be linearly separable. Since the linear separator in the high-dimensional space is nonlinear in the original space, the hypothesis space is greatly expanded.
3) *Nonparametric Method*: SVMs are nonparametric, which gives them the flexibility to represent complex functions. On the other hand, because most of the time only a small fraction of the training data is retained, they are also resistant to overfitting.

Typically, SVMs perform binary classification, where the classifier decides between two available classes. To extend binary SVMs to multilabel classification, one has to employ one of two schemes: a one-vs-one paradigm with voting (binary pairwise classification) or a one-vs-the-rest paradigm where a binary classification is done between one class and all the others. In this work, we employed the one-vs-one paradigm where each binary classifiers contributed to the final multilabel classification by voting. For the sake of completeness, we have also tried the one-vs-the-rest approach, but it consistently gave worse results. Additionally, we employed the kernel trick to map the given 10-dimensional feature vectors into a higher dimensional space (as outlined in [8]) where it can be more easily separated linearly. Specifically, we used radial basis functions of the form $\beta * e^{-\frac{||x_1 - x_2||}{\sigma}}$, with $\beta = 1$. Here x_1 and x_2 represent the feature vectors and σ the associated covariance.

*b) **k-Nearest Neighbor**:* k-Nearest Neighbor classification (k-NN) works directly in the feature space spanned by the feature vectors, in our case a 10-dimensional space. Given a sample data point, it examines the k nearest data points and determines a classification of the current sample using a voting mechanism. If k is selected to be 1 this algorithm simply returns the classification of the nearest neighbor, if $k > 1$ a majority vote is returned. To avoid draws in the voting, it is recommended to use an odd integer for k. Additionally, a distance-dependent weight is employed, weighing closer neighbors more heavily. Although it intuitively makes sense to pick k to be at least the number of labels used, we noticed no significant improvement beyond $k = 3$.

*c) **Decision Trees**:* Multilabel decision trees basically work the same way as binary decision trees as explained in [8] with the main difference that at each level, labels are split into two groups in a way that tries to balance the number of samples on each side (as opposed to the number of labels). A multilabel tree where the labels [1, 2, 3, 4] are distributed with probabilities [0.5, 0.125, 0.25, 0.125] would look like shown in Fig. 1.

C. Cross-Validation

Cross-validation refers to a commonly used technique in machine learning where the dataset is divided into a training set and a test set. The classifier is trained using the training set and its classification accuracy determined using the test set. This process is repeated n times and the average accuracy is reported. We used 10-fold stratified cross-validation, where $n = 10$ and the data set is split $80\%/20\%$ into training and test data.

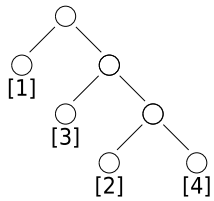

Fig. 1. Example of a multilabel tree.

TABLE I
AVERAGE ACCURACIES OF THE CLASSIFIERS

# of Genres	k-NN	SVM	DT
2	71.10 %	60.71 %	64.14 %
3	53.19 %	46.26 %	48.24 %
4	45.65 %	37.27 %	37.79 %
5	43.64 %	26.82 %	34.55 %

IV. EVALUATION

As mentioned in Section III, our dataset (see [5]) consisted of 150 songs in 6 genres (classical, country, jazz, pop, rock, techno). From this dataset, we only used the artist and title information for feature extraction from the Echo Nest database (see [6]) and genre information as our ground truth. Since most of the classical songs were not found on [6] and the remaining classical songs did not suffice to train the classifiers, we dropped classical music from our genre list. Eventually we ended up with 112 songs in 5 genres with 22 to 23 songs per genre on average.

In order to evaluate the results, we ran the k-NN, SVM and DT classifiers with subsets of the data consisting of songs from a combination of 2, 3, 4 and 5 genres respectively and obtained the accuracy rate in each case. The average rates across the combinations are shown in Table I. It turns out that the classification rate not only depends on the number of genres, but also on the combinations of genres selected. We achieved accuracy rates of up to 97.5% when Pop and Techno are chosen. Fig. 2 shows the accuracy rates of each of the combinations of genres with the shaded bars representing the average value. We also used Tableu, a visual data analysis tool in order to determine which features are more discriminant by plotting each feature against the other and viewing the difference in their mean value for each genre (see Fig. 4 in the appendix).

V. DISCUSSION

From Fig. 2, it can be seen that in each case of 2, 3, 4 and 5 genres, the k-NN classifier yielded the best accuracy rate, followed by Support Vector Machine and Decision Trees.

A better understanding of why this order occurred can be gained by examining the data distribution in space across its

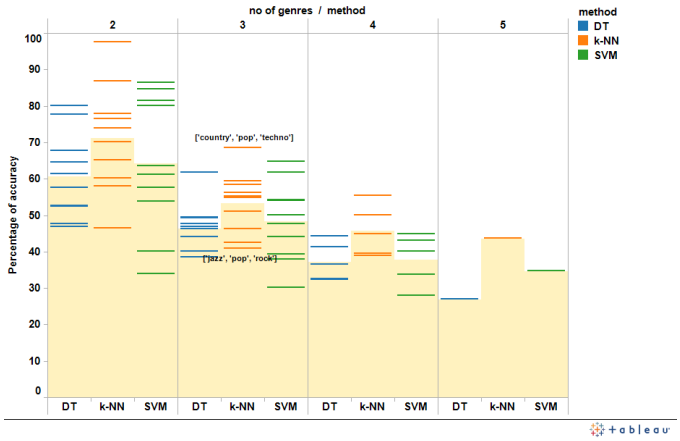

Fig. 2. Accuracy rates of the classifiers. Accuracies are reported in percent for each method broken down by the number of genres. The color indicates the method used.

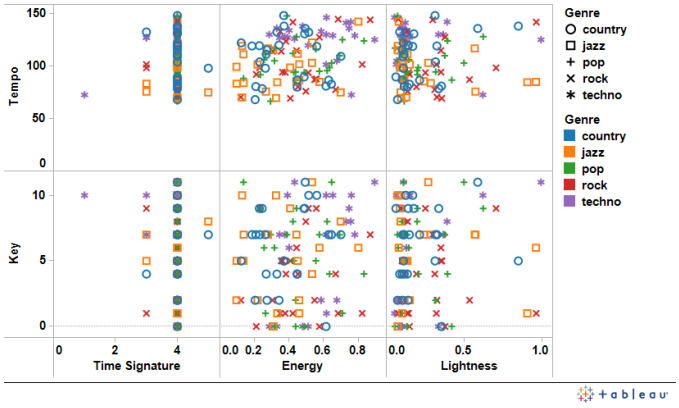

Fig. 3. Featurewise distinction of the data based on genre

10 dimensions (features). Fig. 3 plots a selection of features against one another for each data point in our set. This is a sample of the 5 most discriminant features from our visual analysis (see Fig. 3).

From Fig. 3, it can be seen that the division of the input space according to the genres is not clear and most importantly, does not have any distinct linear boundaries. For an SVM to be efficient, we need the classes to be linearly separable in the kernel space (i.e. the high-dimensional space after applying the kernel trick). Fig. 3 shows that the input space is not linearly separable, but given the low accuracy rates we have to assume that the kernel space does not separate well either. This is why we believe that k-NN has fared better in our classification problem.

For decision trees, the data constructs a complex tree structure with many spurious relationships that affect the accuracy rate. It would be interesting to see how well the tree balances and whether the classification rate improves after we perform a dimensionality reduction (i.e. feature selection of highly discriminant features) using a technique such as Principal Component Analysis. As we increased the number of genres for the songs to be classified the achieved accuracy decreases as expected. Not only do the number of genres affect the accuracy rate but also the particular genres selected as Fig. 2 shows. The accuracy rate for country, pop, techno is 68.46 while for jazz, pop, rock it is as low as 40.77. The five genres used in this paper exhibit fairly similar characteristics, which complicates reliable classification. We believe that one of the main reasons Salamon et al. [4] have achieved such high accuracy rates of up to 90 % is that the genres they used for classification are as disparate as opera and pop.

This leads us to question how discriminant the individual features are with respect to the genres. Thus, while selecting the data for the classification problem we learned that we need to consider two important factors:

1) Are the genres/classes significantly different from each other?
2) Are the features discriminant enough?

While 1) might not be fully under our discretion, we can definitely improve on 2) by combining feature extraction/dimensionality reduction methods such as Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) and feed the output of those into our classifiers.

While all of this holds, we can try to increase the accuracy of the classification problem by extending feature extraction directly from the audio files and increasing the size of our dataset. While feeding classifiers with more data would benefit k-NN and SVM classifiers, for multi-tree classifiers however, we expect that we end up overfitting, which will need to be countered by an appropriate pruning technique.

In our classification, we have used only global features. It would be interesting to see how our classification accuracies could be improved by making use of local features (such as commonly used MFCC features [4]) of a song such as pitch and timbre features, vibrato features and contour topology. Features such as these include calculating for example the mean pitch across the segments that make up an audio piece. The key here is to use the domain knowledge required for such analyses. Could we, by using sheer computing power determine which local features would be most appropriate? That is a question worth exploring.

References

[1] M. Talupur, S. Nath, and H. Yan, "Classification of music genre," 2001. [Online]. Available: http://www.cs.cmu.edu/~yh/files/GCfA.pdf

[2] A. Galkin, P. Rege, and R. Pocratsky, "24787 artifical intelligence and machine learning for engineering design - classficiation of music using neural networks," 2011. [Online]. Available: agalkin.com/media/music/proj_music_report.pdf

[3] G. Tzanetakis and P. R. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[4] J. Salamon, B. Rocha, and E. Gómez, "Musical genre classification using melody features extracted from polyphonic music signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, 25/03/2012 2012. [Online]. Available: files/publications/SalamonRochaGomezICASSP2012.pdf

[5] D. Turnbull. (2012, July) Music genre classification. [Online]. Available: http://modelai.gettysburg.edu/2012/music/

[6] T. Jehan and B. Whitman. (2012) The echo nest. [Online]. Available: http://the.echonest.com/

[7] L. P. Coelho. (2008 - 2012) Milk: Machine learning toolkit. [Online]. Available: http://packages.python.org/milk/

[8] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2010.

APPENDIX

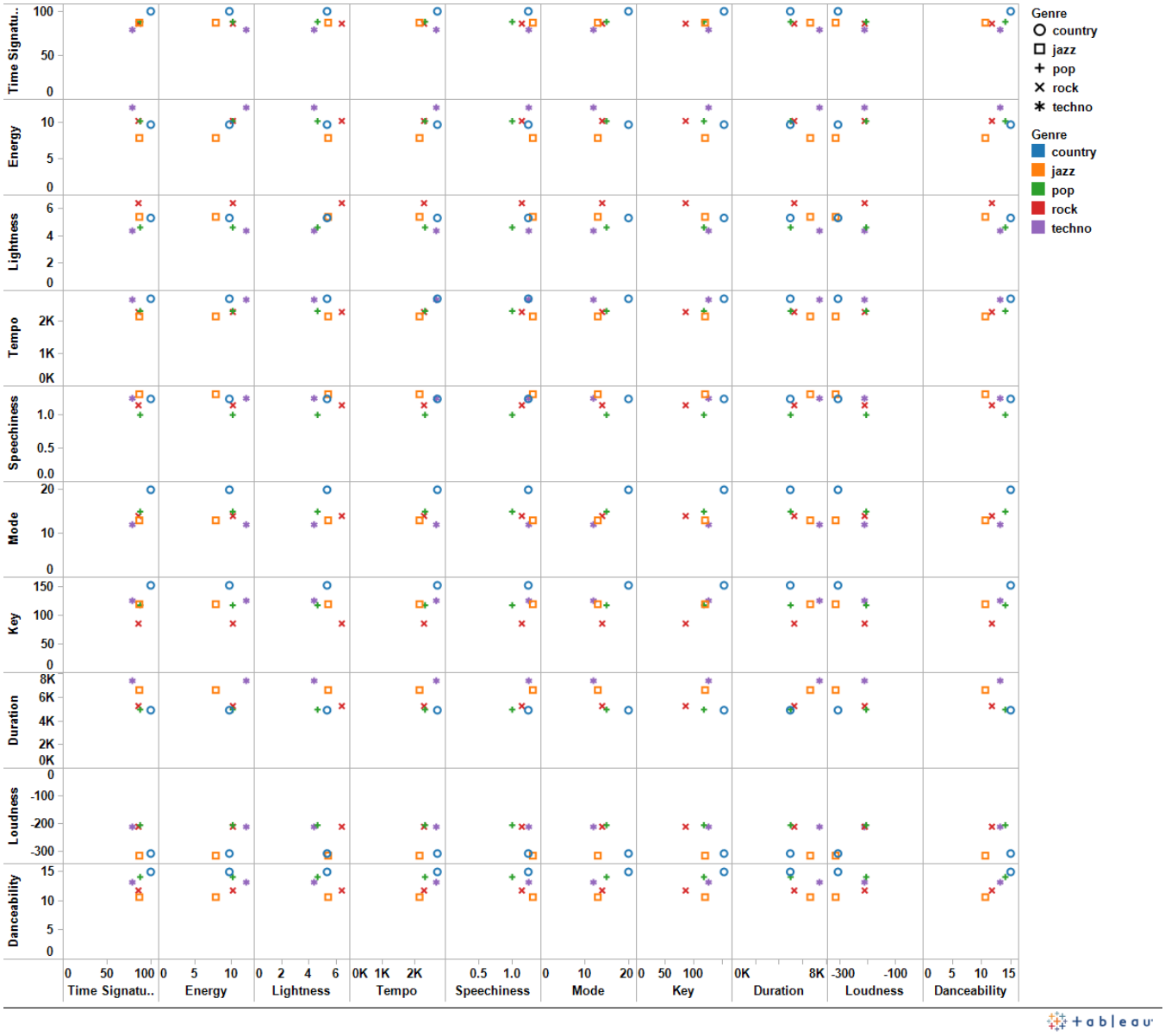

Fig. 4. The full 10 by 10 feature matrix. Shown are the average values for each feature in each genres.