Music Genre Classification

{ Charles Burlin, Matthew Creme and Raphael Lenain} Stanford University, CS 229

Predicting

Our goal was to develop techniques that would take a music file (.mp3, .au) as input, and output its genre, among a set number of genres. We looked at Country, Classical, Metal, Pop, and often times we would only try to classify amongst subsets of these genres.

We used different classification techniques, and we found that our algorithms could classify well when splitting music among up to four genres. We also used .midi files to look at music more analytically in order to classify amongst different composers.

Data

The first section of our data came from the Marsyas [1] data set which consists of 1000 .au files, each 30 seconds long, split into the ten genres. Features were extracted from the music files using Mel Frequency Cepstral Coefficients (MFCCs).

A training example consisted of a matrix: columns corresponding to the features, and rows corresponding to time steps. The .midi section of our data came from Classical Piano Midi Page (http://www.piano-midi.de/) and was processed using the music21 open source Python toolkit.

Models

SVMs - Supervised Learning technique using the Hinge loss and optimization through stochastic gradient descent. For a single training example x, the loss function is

$$l(x) = \max(0, 1 - x)$$

k-Nearest Neighbors - Apply PCA first to reduce to 3 dimensions. Given a testing example, find the k nearest neighbors using Euclidean distance (with the 2-norm) $d(x_1, x_2) = ||x_1 - x_2||$ and classify the example as the most represented class among the classes of the neighbors. The parameter k is chosen through training as well.

Decision Tree - We use Globally-optimal classification tree analysis to optimize the weights on the branches of our tree. The nodes are obtained from the features.

Back Propagation Neural Network - Given a neural network N and training examples $(x_1, y_1), \ldots (x_p, y_p)$, we would like to train the weights of N. We do so by using the softmax function $\frac{e^{x_{j,i}}}{\sum_i e^{x_{j,i}}}$ as the final activation function in the network, and running gradient descent to find optimal W, the weights, to diminish training error.

Features

A training example is first comprised of the MFCC coefficients. In addition, the delta and acceleration values of the MFCCs, calculated as $\Delta v_i = v_i - v_{i-1}$ and $\text{acc}(v_i) = \Delta v_i - \Delta v_{i-1}$ were used as features. Therefore, each training example was represented as a large matrix. Now, most of the algorithms that we used usually treat vectors as inputs, hence we either applied PCA to our matrix, or we flattened the matrix to an extremely large vector, and then used that as a training example.

Hence, our raw features are the 13 MFCC coefficients, the 13 velocity coefficients and the 13 acceleration coefficients for each time step. We were initially concerned with this choice of features because of the potential of over fitting, but through separating our data between training sets and testing sets, we were able to confirm that we did not over fit.

Composer classification

The loaded data was processed as to keep the melody and only the pitch was kept. A song was therefore reduced to a sequence of pitches with no timing. Then, we considered a pitch or a sequence of pitches as a state and to compute the probabilities of moving to another state from it by recording the number of transitions occurring in the training set. For each song in the test set, we found the nearest transition matrix in terms of L2 norm.

Discussion

Most of our algorithms were almost perfect on binary classification. When looking at four genres, they were doing fairly well, and we were rather surprised by that, especially considering the size of our feature space. Due to this, we tried applying PCA to our data, but this only improved our results in the case of k-NN. Hence, we successfully implemented music genre classification software. However, composer classification remains improvable, particularly when considering more composers.

Results

For each of our models, we used 70 of the 100 songs of each genre as training data and the other 30 songs as test data. Here are the results for the various models:

Model	Test-Statistic	Metal	Classical	Pop	Country	F-Score
Neural Network	Recall	93%	83%	93%	90%	0.9
	Precision	90%	93%	100%	79%	
SVM- Linear Kernel	Recall	97%	67%	93%	78%	0.847
	Precision	78%	95%	93%	83%	
SVM- Polynomial Kernel	Recall	97%	63%	97%	80%	0.838
	Precision	81%	83%	97%	77%	
K-NN with PCA	Recall	93%	73%	93%	77%	0.842
	Precision	76%	81%	100%	82%	
Decision Tree	Recall	90%	93%	67%	67%	0.793
	Precision	96%	90%	65%	67%	

We ran the described algorithm for composer classification with different depths, the depth being the dimension of a state vector: a depth of 3 means that we consider the last three transitions.

Composer Pair	Beethoven Liszt	Beethoven Mozart	Liszt Mozart
Optimal Depth	2	2	3
Precision	75%	100%	100%
Recall	60%	40%	83%
F-Score	0.67	0.57	0.93

References

[1] Marsyas *"Data Sets"*. marsyasweb.appspot.com

[2] I. Karpov *"Hidden Markov Classification for Musical Genres"*. Rice University. 2002.

[3] M. Nielsen *"How the backpropagation algorithm works"*. neuralnetworksanddeeplearning.com. 2016.

Future Research

Time permitting, we would have found more datasets to test the robustness of our results by training on one dataset and testing on another.

Considering additional musically relevant features such as beats per minute might help our composer classification.

Team

Three MS students in ICME, in the MCF track. Contact us at cburlin@stanford.edu, mcreme@stanford.edu, rlenain@stanford.edu.