HW 5: Music Genre Classification

Anamol Pundle

March 13, 2015

Abstract

Machine learning techniques are utilized in this work to classify music from several different bands and genres after training the algorithm with music from the same bands and genres. A random assortment of fifteen songs from each band/genre is compiled, and spectrograms of these data are made. An 'economy' singular value decomposition of the spectrograms is performed, subsequent to which randomly selected songs from each band/genre are used to train the algorithm, and the rest are used as test sets. This is repeated several hundred times and an average accuracy percentage is calculated. Three different analyses are performed using two machine learning algorithms. Accuracy percentages are found to vary from 40% to 80%.

1 Introduction and Overview

Music genres are instantly recognizable to us, whether it be jazz, classical, blues, rap, rock, etc. One can always ask how the brain classifies such information and how it makes a decision based upon hearing a new piece of music. The objective of this work is to attempt to write a code that can classify a given piece of music by sampling a five second clip. This is done using machine learning algorithms, namely *linear discriminant analysis* and the *naive Bayes classifier*.

Three different analyses are conducted in this work. Three bands/genres of music are chosen for each analysis. Fifteen songs of each band/genre are chosen and a five second sample from song is selected, after which a spectrogram is constructed. The spectrogram is decomposed into its constituent principal components, after which certain rows from the unitary rotation matrix $'V'$ are chosen as the training set and test set (without overlap). The LDA and naive ayes are then applied to these cases. Cross validation of the training and tests sets is also done.

The first analysis is the classification of music from three bands playing different genres of music. The bands/musicians chosen for this analysis are Pink Floyd (progressive/psychedelic rock), John Coltrane (jazz) and Nirvana (grunge). The second analysis is the classification of bands from the same genre, in this case, grunge. The bands chosen are Nirvana, Soundgarden and Pearl Jam. The third analysis is the classification of songs by different genres.

In this case, an assortment of songs by different artists in the genres of grunge, jazz and classic rock are used.

2 Theoretical Background

2.1 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant are methods used in statistics, pattern recognition and machine learning to find a linear combination of features which characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification. LDA is closely related to analysis of variance (ANOVA) and regression analysis, which also attempt to express one dependent variable as a linear combination of other features or measurements. However, ANOVA uses categorical independent variables and a continuous dependent variable, whereas discriminant analysis has continuous independent variables and a categorical dependent variable (i.e. the class label). Logistic regression and probit regression are more similar to LDA, as they also explain a categorical variable by the values of continuous independent variables. These other methods are preferable in applications where it is not reasonable to assume that the independent variables are normally distributed, which is a fundamental assumption of the LDA method. LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the classes of data. PCA on the other hand does not take into account any difference in class, and factor analysis builds the feature combinations based on differences rather than similarities. Discriminant analysis is also different from factor analysis in that it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made. LDA works when the measurements made on independent variables for each observation are continuous quantities. When dealing with categorical independent variables, the equivalent technique is discriminant correspondence analysis.

2.2 Naive Bayes Classifier

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate preprocessing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

3 Algorithm Implementation and Development

Similar algorithms are used for each of the test cases, as given below.

- Compile a set containing 5 second snippets of 15 songs per band/genre and save as a wav file downsampled to 11kHz and converted to mono for smaller size.
- Load wav files in matlab using wavread. Convert wav data into single precision float using single function.
- Create a spectrogram of each wav file. This is not elaborated upon as an entire HW report dedicated entirely to this process was written by the author (HW 2: Gabor transforms).
- Reshape each spectrogram such that the time-frequency data of each song snippet is contained in one single row.
- Create a new matrix Sgtot which contains the spectrogram data of each song, with songs of each band/genre being grouped together.
- Perform an 'economy' singular value decomposition of Sgtot, using the flag $'0'$ in the function svd. The $'V'$ matrix is a dimensionally lower matrix and can be used in machine learning algorithms efficiently. The rows of the matrix are the song snippets and the columns are the modes or the principal components.
- Choose ten songs of each band/genre randomly (without replacement) for the training set. The remaining five songs become the test set.
- Stack up the songs of the training set in a matrix, with a certain number of modes chosen (this is a variable to be adjusted to get maximum accuracy).
- Do likewise with the test set.
- Construct a matrix which characterizes the answers to the training set, with each band/genre being assigned a number of 1, 2 or 3.
- The training set, test set and the answers to the training set are passed to the classify function and the NaiveBayes function (followed by the predict function).

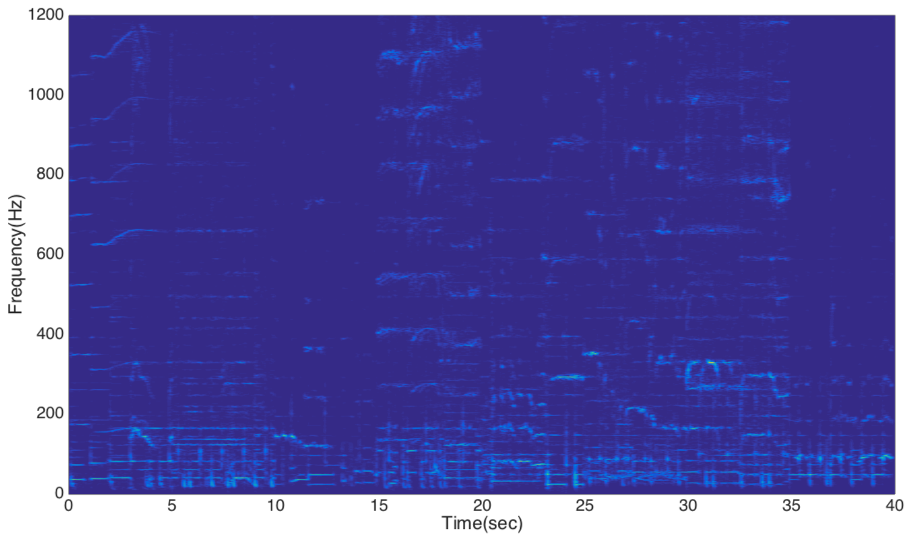


Figure 1: Spectrogram of eight five second samples of Pink Floyd songs

- Calculate the cumulative error in identifying songs of every band/genre by comparing the actual answers to the predicted answers of the test set.
- Calculate the error by band/genre also.
- Repeat steps 7-13 one thousand times. Average the error over the 1000 iterations.
- Adjust the number of principal components used, and repeat steps 7-14 until the accuracy is maximized.

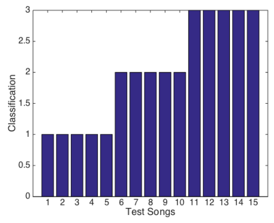
4 Computational Results

Figure 1 shows the spectrogram generated by a sample of five second snippets of eight Pink Floyd songs. Gilmour's lead guitar is seen to be composed of several overtones, as seen in 0-5 secs, 15-20 secs and 25-30 secs (guitar solos in Breathe, Time and Dogs). Other, relatively quieter songs such as Fearless (10-15 secs) and San Tropez (35-40 secs) do not seem to have many overtones, since the lead guitar is not present in those pieces.

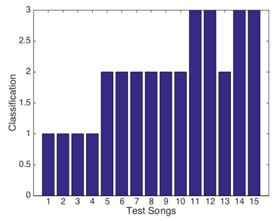
The individual results of the three tests are described below.

4.1 Test 1: Pink Floyd, John Coltrane and Nirvana

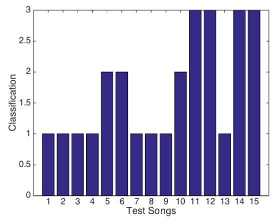
The three bands are all of distinctly different genres, so we expect that both machine learning algorithms would give a reasonably good classification. Indeed, we see that this is the case, though LDA outperforms the naive Bayes classifier



(a) Actual grouping



(b) Grouping predicted by LDA



(c) Grouping predicted by Naive Bayes

Figure 2: Histograms showing actual and predicted grouping of test songs

algorithm. the average accuracy for the naive Bayes in prediction is 66.60% with a standard deviation of 9.82, while that of the LDA is 78.20% with a standard deviation of 9.93%. Figure 2 shows the actual and predicted grouping of test songs from one simulation. Good prediction is observed, especially for the first and second bands (Pink Floyd and John Coltrane).

Figure 3 shows the percentage accuracy averaged over 100 tests for each individual band. Accuracy for Pink Floyd and John Coltrane is seen to be very high, but observed to be low for Nirvana.

Tests were also run with a different number of principal components used to train the algorithm. The number of modes used to train and test both algorithms was varied from 1 to 10. This is shown in Figure 4. It is seen that LDA generally gives better accuracy than naive Bayes, and the accuracy peak at 4 principal components for the LDA and 1 principal component for naive Bayes.

4.2 Test 2: Pearl Jam, Soundgarden and Nirvana

Since all three bands are from the same genre(grunge), the same geographical area(Seattle) and roughly the same time(90's), we expect roughly the same musical qualities in each band. Both the LDA and naive Bayes show that this is the case; the average accuracy for the naive Bayes in prediction is 43.33% with a standard deviation of 10.56%, while that of the LDA is 50.33% with a standard deviation of 10.85%. Figure 5 shows the actual and predicted grouping of test

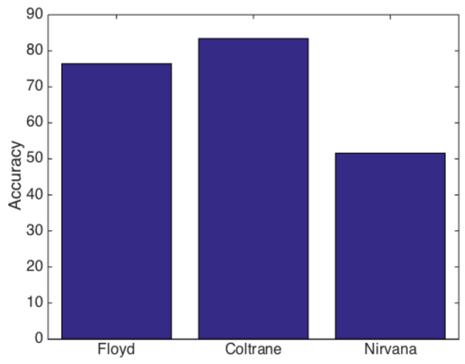


Figure 3: % Accuracy for individual bands

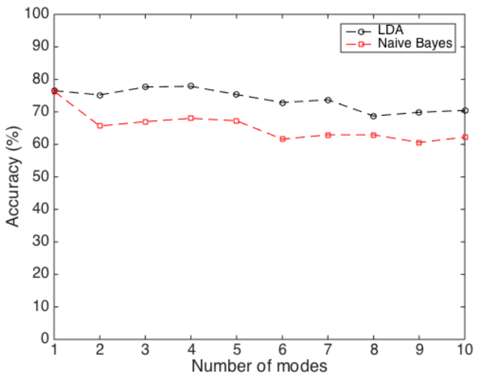
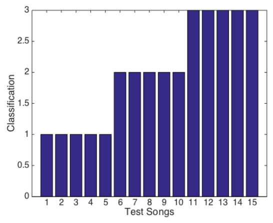
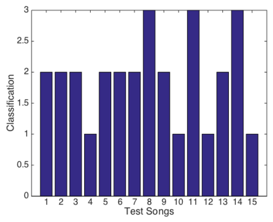


Figure 4: Accuracy as a function of number of modes used for classification


(a) Actual grouping


(b) Grouping predicted by LDA

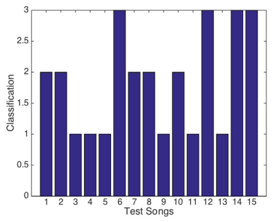

(c) Grouping predicted by Naive Bayes

Figure 5: Histograms showing actual and predicted grouping of test songs

songs from one simulation. No real prediction is observed.

Figure 6 shows the percentage accuracy averaged over 100 tests for each individual band. The accuracy for all three hovers around 50%.

Tests were also run with a different number of principal components used to train the algorithm. The number of modes used to train and test both algorithms was varied from 1 to 10. This is shown in Figure 7. It is seen that LDA generally gives better accuracy than naive Bayes, and the accuracy peak at 6 principal components for the LDA and 3 principal components for naive Bayes.

4.3 Test 3: Jazz, Classic Rock and Grunge

The final test involve testing songs by several artists in three different genres. The genres chosen are jazz, classic rock and grunge. The average accuracy for the naive Bayes in prediction is 67.00% with a standard deviation of 11.11%, while that of the LDA is 65.2% with a standard deviation of 10.09%. Figure 8 shows the actual and predicted grouping of test songs from one simulation. In this case, it is seen that the LDA and naive Bayes predict approximately the same level of accuracy.

Figure 9 shows the percentage accuracy averaged over 100 tests for each individual band. The accuracy for jazz is quite high (approximately 80%), while that of classic rock is lower (60%) and that of grunge is scraping the bottom of the barrel (50%). We'd do about the same if we flipped a coin.

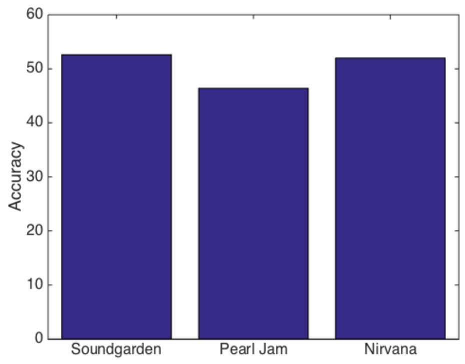


Figure 6: % Accuracy for individual bands

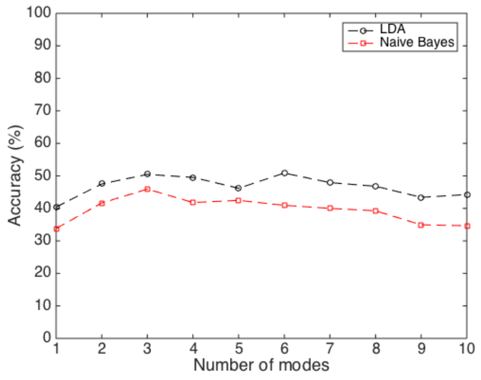
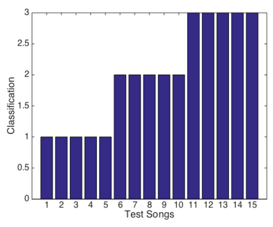
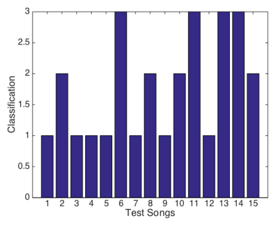


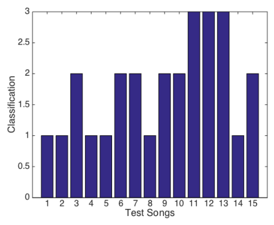
Figure 7: Accuracy as a function of number of modes used for classification



(a) Actual grouping



(b) Grouping predicted by LDA



(c) Grouping predicted by Naive Bayes

Figure 8: Histograms showing actual and predicted grouping of test songs

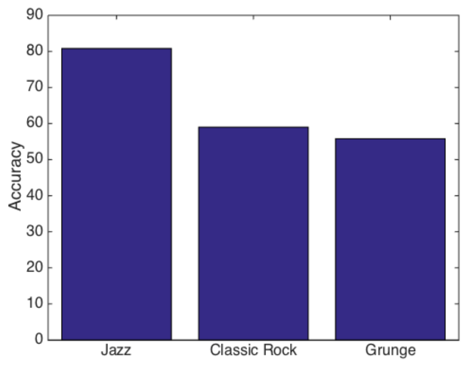


Figure 9: % Accuracy for individual genres

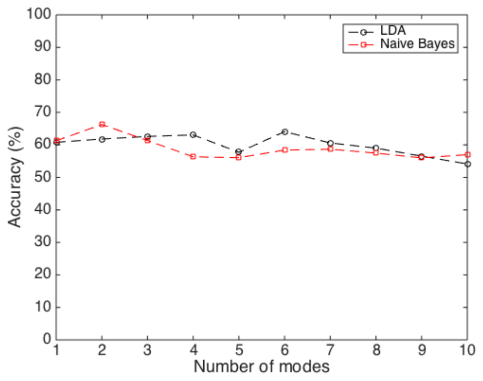


Figure 10: Accuracy as a function of number of modes used for classification

Tests were also run with a different number of principal components used to train the algorithm. The number of modes used to train and test both algorithms was varied from 1 to 10. This is shown in Figure 10. Unlike the first two cases, there is clear cut winner between LDA and naive Bayes. The accuracy peaks at 4 principal components for the LDA and 2 principal components for naive Bayes.

5 Summary and Conclusions

Two machine learning algorithms, the LDA and the naive Bayes classifier are applied to the problem of identifying musical genres. Three analyses are conducted, one with three bands/musicians belonging to three distinct genres, the second with three bands belonging to the same genre, and the third with several bands belonging to three distinct genres.

It is found that the LDA performs better than the naive Bayes classifier in two out of three tests. Another interesting result is that the inclusion of all principal components is not necessary for a good prediction model; 3-5 modes generally give the highest accuracy. Jazz music is exceedingly well-predicted by both algorithms over a large number of artists and styles; this confirms the authors' suspicions that all that jazz is, in fact, the same frickin' thing, and leads him to question the musical taste of jazz lovers across the world.

For the first test, the average accuracy for the naive Bayes in prediction is 66.60% with a standard deviation of 9.82, while that of the LDA is 78.20% with a standard deviation of 9.93%. Pink Floyd and John Coltrane have accuracies of approximately 75% and 83%, respectively, while Nirvana has a lower accuracy of about 50% (LDA).

For the second test, the average accuracy for the naive Bayes in prediction is 43.33% with a standard deviation of 10.56%, while that of the LDA is 50.33% with a standard deviation of 10.85%. The prediction accuracies of all bands is approximately 50% with the LDA.

For the third test, the average accuracy for the naive Bayes in prediction is 67.00% with a standard deviation of 11.11%, while that of the LDA is 65.2% with a standard deviation of 10.09%.

References

1. Kutz, J. Nathan, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, September 2013

2. Matlab help

Appendix I: MATLAB functions used

`wavread`: reads wave file into MATLAB

- `y = wavread(filename)` loads a WAVE file specified by the string filename, returning the sampled data in y. If filename does not include an extension, `wavread` appends .wav

`fft`: One dimensional discrete Fourier transform

- `Y = fft(x)` returns the discrete Fourier transform (DFT) of vector x, computed with a fast Fourier transform (FFT) algorithm.

`svd`: Singular value decomposition.

- `[U,S,V] = svd(X)` produces a diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that `X = U*S*V'`.

`single`: Convert to single precision.

- `Y = single(X)` converts the vector X to single precision. X can be any numeric object (such as a DOUBLE).

`classify`: Discriminant analysis.

- `CLASS = classify(SAMPLE,TRAINING,GROUP)` classifies each row of the data in `SAMPLE` into one of the groups in TRAINING. SAMPLE and TRAINING must be matrices with the same number of columns. GROUP is a grouping variable for TRAINING. Its unique values define groups, and each element defines which group the corresponding row of TRAINING belongs to.

`fitNaiveBayes`: Create a Naive Bayes classifier object by fitting to data.

- `NB = fitNaiveBayes(TRAINING, C)` builds a NaiveBayes classifier object NB. TRAINING is an N-by-D numeric matrix of predictor data. Rows of TRAINING correspond to observations; columns correspond to features. C contains the known class labels for TRAINING, and it take one of K possible levels.

`predict`: Computes the k-step ahead prediction.

- `YP = predict(MODEL, DATA, K)` predicts the output of an identified model MODEL K time instants ahead using input-output data history from DATA. The predicted response is computed for the time span covered by DATA.

```
% APPENDIX II: MATLAB CODE


clear all; close all; clc;

y = wavread('Allthatjazz');
z = wavread('ClassicWrok');
x = wavread('Grunge');

deodato_rec = 75;
floyd_rec = 75;
y = single(y); z = single(z); x = single(x);

v = y'/2;
w = z'/2;
x = x'/2;
Fs = length(v)/floyd_rec;
Fs2 = length(w)/deodato_rec;
Fs3 = length(x)/deodato_rec;
a = 100;

L = length(v)/Fs;
L2 = length(w)/Fs2;
L3 = length(x)/Fs3;
k=(2*pi/(2*L))*[0:(length(v)-1)/2 -(length(v)-1)/2:-1]; ks=fftshift(k);

if rem(length(k), 2) > 0

    ks = ks(length(ks)/2:end);
else
    ks = ks(length(ks)/2-1:end);
end
k = single(k); ks = single(ks);
tfinal = length(v)/Fs;
t = single(1:length(v))/Fs;

Sgt_spec = []; tslide = 0:0.1:tfinal; tslide = single(tslide);
Sgt_spec2 = []; Sgt_spec3 = []; Spec1 = []; Spec2 = [];
Spec3 = [];

for ii = 1:length(tslide)
    g = exp(-a*(t-tslide(ii)).^2);
    Sg = g.*v; Sg2 = g.*w; Sg3 = g.*x;
    Sgt = fft(Sg); Sgt2 = fft(Sg2); Sgt3 = fft(Sg3);
    Sgt = fftshift(Sgt); Sgt2 = fftshift(Sgt2); Sgt3 = fftshift(Sgt3);
    LSgt = length(Sgt);
    Sgt = Sgt(int64(LSgt/2):end); Sgt2 = Sgt2(int64(LSgt/2):end);
    Sgt3 = Sgt3(int64(LSgt/2):end);
    Sgt = single(Sgt); Sgt2 = single(Sgt2); Sgt3 = single(Sgt3);

    Sgt_spec = [Sgt_spec abs((Sgt))]; Sgt_spec2 = [Sgt_spec2 abs((Sgt2))];
    Sgt_spec3 = [Sgt_spec3 abs((Sgt3))];
```

```
        if rem(ii+50, 50) == 0
            Spec1 = [Spec1; Sgt_spec];
            Spec2 = [Spec2; Sgt_spec2];
            Spec3 = [Spec3; Sgt_spec3];
            Sgt_spec = [];
            Sgt_spec2 = [];
            Sgt_spec3 = [];
        end



    end

    Sgtot = [Spec1; Spec2; Spec3];
    clear Sgt_spec; clear Sgt_spec2; clear Sgt_spec3;
    clear Sgt1; clear Sgt2; clear Sgt3;
    [u,s,v] = svd(Sgtot',0);

    nberr = []; ldaerr = [];
    nbstd = []; ldastd = [];
    for kkk = 1:1

        clc;
        rowbeg = 1;
        rowend = 4;
        trainp = 10;
        testp = 15 - trainp;
        allanswers = [ones(15,1); 2*ones(15,1); 3*ones(15,1)];
        Efinal = 0; Enbfinal = 0;
        Pf = 0; Col = 0; Deo = 0;
        Errlda = []; Errnb = [];
        for jj = 1:100
            floydtrain = []; coltrain = []; deotrain = [];
            ftest = []; coltest = []; deotest = [];
            fsampl = randsample(15,trainp); ft = setdiff(1:15, fsampl)';
            colsampl = randsample(15,trainp) + 15; colt = setdiff(16:30, colsampl)';
            deosampl = randsample(15,trainp) + 30; deot = setdiff(31:45, deosampl)';

            for kk = 1:trainp
                ff = v(fsampl(kk), rowbeg:rowend);
                cc = v(colsampl(kk), rowbeg:rowend);
                dd = v(deosampl(kk), rowbeg:rowend);
                floydtrain = [floydtrain ; ff];
                coltrain = [coltrain ; cc];
                deotrain = [deotrain ; dd];
            end
            for kk = 1:testp
                ff = v(ft(kk), rowbeg:rowend);
                cc = v(colt(kk), rowbeg:rowend);
                dd = v(deot(kk), rowbeg:rowend);
                ftest = [ftest ; ff];
                coltest = [coltest ; cc];
                deotest = [deotest ; dd];
            end
            train = [floydtrain; coltrain; deotrain];
            tests = [ftest; coltest; deotest];
            correct = [allanswers(11:15); allanswers(26:30); allanswers(41:45)];
            answer = [ones(trainp,1); 2*ones(trainp,1); 3*ones(trainp, 1)];


            [ind err] = classify(tests, train, answer);

            Err = 0;
            for ii = 1:length(ind)
                if ind(ii) == correct(ii)
                    Err = Err + 1;
                    if ii <=5
                        Pf = Pf +1;
                    elseif ii <=10
                        Col = Col + 1;
                    else
                        Deo = Deo + 1;
                    end
                end

            end
            Err = Err/15*100;

            Errlda = [Errlda Err];


            figure(2)
            nb = fitNaiveBayes(train, answer);
            prednb = nb.predict(tests);

            figure(2)
            bar(ind);
            set(gca, 'FontSize',16);
            xlabel('Test Songs'); ylabel('Classification');

            Err = 0;
            for ii = 1:length(prednb)
                if prednb(ii) == correct(ii)
                    Err = Err + 1;
                end

            end
            Err = Err/15*100;
            Errnb = [Errnb Err];

        end

        mean(Errnb)
        std(Errnb)

        mean(Errlda)
        std(Errlda)


        Pf = Pf/(testp*jj)*100;
        Col = Col/(testp*jj)*100;
        Deo = Deo/(testp*jj)*100;
        Banderr = [Pf Col Deo];
        Bands = ['Floyd', 'COl', 'ddd'];


        nberr = [nberr mean(Errnb)];
        nbstd = [nbstd std(Errnb)];
        ldaerr = [ldaerr mean(Errlda)];
        ldastd = [ldastd std(Errlda)];


end
```

Published with MATLAB® R2014b