MUSIC GENRE CLASSIFICATION –MULTINOMIAL LOGISTIC REGRESSION

MULTINOMIAL LOGISTIC REGRESSION CLASSIFIER INTRODUCTION:

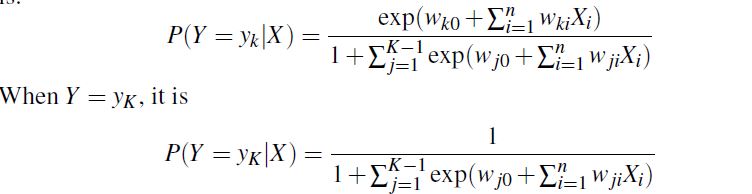
MULTINOMIAL LOGISTIC REGRESSION CLASSIFIER:

Logistic Regression is an approach to learning functions of the form f : X →Y, or P(Y|X) in the case where Y is discrete-valued, and X = hX1 ...Xni is any vector containing discrete or continuous variables.

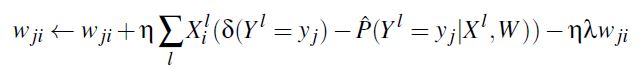
For this Project we are implementing Multinomial Logistic Regression is to classify music audio files into different Genre. The Weight updating during the training is done through Gradient Descent.

The formula below represents the mathematical representation of the Logistic regression and Gradient Descent Weight Update.

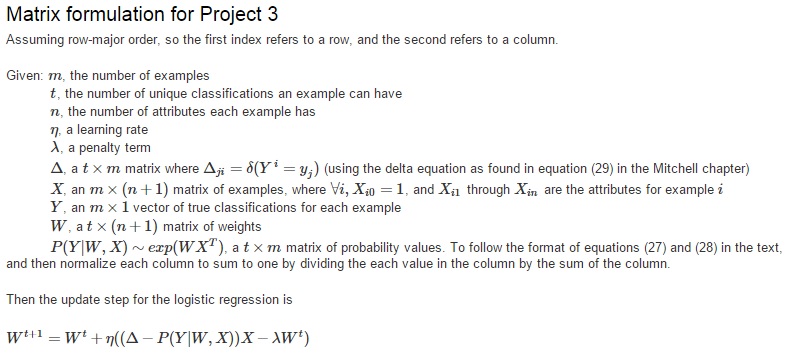
Logistic Regression Probability Matrix:



Weight Update:



The matrix manipulation way used in my code is as follows:



CODE DESCRIPTION:

I have 3 Python Codes for this project

**music\_genre\_fft.py** –Classifier by extracting FFT[First 1000 Features] Features of the Wav files

1. I extract the First Thousand FFT Features of each music file of different Genre by the following code

sample\_rate,x=scipy.io.wavfile.read(file\_path)  
fft\_features=abs(scipy.fft(x)[:1000])  
data.append(fft\_features)

genre\_list.append(1)

1. Finally with input matrix of size 600\*1001(+1 bias) and perform 10 fold cross validation to split 90 % of traing and 10% of testing.

kf = cross\_validation.KFold(600, n\_folds=ten,shuffle=True)

1. Initalizing weight matrix of size 6\*1001 and Delta matrix with 1 and 0s of size (genre calss,training data size)(here its 6\* 520(90 % of training)) perform Logistic regression on the trained data

**def** cal\_Prob\_init(data,weight\_matrix):  
 norm\_trans=data.transpose()  
 Prob\_trans=np.power(np.e, np.dot(weight\_matrix, norm\_trans))  
 Prob\_trans[len(Prob\_trans)-1]=1  
 Prob\_trans=Prob\_trans/Prob\_trans.sum(axis=0)**return** Prob\_trans

1. Using GRADIENT DESCENT concept update the weight for around 1000 epochs and then the trained weight is used for tesing dataset (10% of the data) and produce the confusion Matrix and accuracy

**def**training(weight\_matrix,new\_eta,Delta,norm\_train,prob\_train,lam\_init):  
 weight\_matrix=weight\_matrix+(new\_eta\*(np.dot((Delta-prob\_train),norm\_train)-(lam\_init\*weight\_matrix)))  
**return** weight\_matrix

1. Step 4 is repeated 10 time so that we have tested on the entire 600 files of data(every time 10% of unique test data which is not used in previous iteration)

**Music\_genre\_fft\_best20.py** -Top 20 features of each genre and perform Multinomial Logistic Regression:

Top 20 features of each Gnere is done through Standard Deviation and we get an Input Matrix of 600\*118 as the way i took only 2 features collide with other Genre.and Done the same logic as descriped in the previous python code description

**Mfcc\_music\_genre.py**-MFCC Feature extraction:

Feature Extraction is done through MFCC and the same Logistic Regression is applied on this data

CONFUSION MATRIX, ACCURACY AND BIAS EXPLANATION:

1. FFT FEATURE EXTRACTION : CONFUSION MATRIX AND ACCURACY

[[73 10 0 13 3 1]

[ 7 44 3 31 13 2]

[ 5 8 12 59 12 4]

[ 6 3 2 76 11 2]

[ 3 5 3 50 33 6]

[ 5 3 1 51 28 12]]

0.416666666667

As per the confusion matrix my Most of my music files are classified belonging to Genre 1(Classical) correctly and when compared diagonally nearly half of the music file are classified correctly but our output are biased mostly towards class1 as most of music files are classified to Genre 1 .Our classifier get confused between as their features appears to be similar and our predictions are not that accuracte.

1. FFT TOP 20 FEATURES: CONFUSION MATRIX AND ACCURACY

[[66 8 1 13 11 1]

[17 36 2 11 32 2]

[ 4 5 9 30 47 5]

[ 1 3 1 63 30 2]

[ 2 1 1 24 68 4]

[ 2 3 0 37 49 9]]

ACCURACY: 0.418333333333

As per the confusion matrix my Most of my music files are classified belonging to Genre 1(Classical) correctly and when compared diagonally less than half of the music file are classified correctly but our output are biased mostly towards class1 as most of music files are classified to Genre 1 since our feature extraction of top 20 FFT features are more similar toward classical. The FFT features are most similar between those classes.

1. MFCC FEATURE EXTRACTION: CONFUSION MATRIX AND ACCURACY

[[88 4 3 0 2 3]

[26 37 18 2 11 6]

[15 13 47 6 12 7]

[ 2 2 4 86 4 2]

[ 4 10 14 8 34 30]

[ 0 2 2 0 2 94]]

ACCURACY: 0.643333333333

MFCC Feature extraction classifier yields better accuracy when compared to the above 2 methods of Feature extraction. As most of the classification with respect to Genre 1 are correct and Diagnal matrix appears to more when compared to the other predictions

REFERNCES:

CODE:

1. http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

2. www.docs.spicy.org, for referencing the python syntaxes

3. Referenced Piazza codes, discussions and formula for implementing GRADIENT DESCENT and LOGISTIC REGRESSION.

DOCUMENT:

1. http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf
2. <http://dimacs.rutgers.edu/Research/MMS/authorID-me05-fixed.pdf>

APPENDIX:

RESULTS:

FFT:

C:\Python27\python.exe "C:/Users/Thilak/Desktop/ml/pjt 3/pjt3code/music\_genre\_fft.py"

[[ 9 0 0 0 0 0]

[ 0 5 1 2 0 0]

[ 1 0 0 11 2 0]

[ 1 0 0 9 1 0]

[ 0 0 0 9 0 0]

[ 0 0 0 9 0 0]]

[[14 0 0 0 1 0]

[ 2 7 1 10 0 0]

[ 1 0 2 19 4 0]

[ 1 0 1 18 1 0]

[ 0 0 1 15 3 0]

[ 0 2 0 15 2 0]]

[[18 2 0 0 1 0]

[ 4 10 3 12 3 0]

[ 1 0 6 28 4 1]

[ 1 0 1 23 2 0]

[ 0 0 1 21 7 1]

[ 0 2 0 22 5 1]]

[[25 2 0 2 1 0]

[ 6 16 3 12 5 0]

[ 2 0 6 33 5 2]

[ 2 1 2 31 4 0]

[ 0 2 1 27 10 2]

[ 1 2 0 24 10 1]]

[[38 3 0 3 2 0]

[ 6 22 3 14 6 0]

[ 2 1 7 36 5 2]

[ 2 1 2 40 5 0]

[ 0 2 1 34 12 4]

[ 2 2 0 30 11 2]]

[[43 7 0 5 2 0]

[ 6 29 3 16 7 0]

[ 2 1 8 39 5 2]

[ 2 3 2 48 7 1]

[ 2 2 2 38 17 4]

[ 2 2 0 31 15 7]]

[[47 9 0 8 3 1]

[ 6 31 3 22 11 1]

[ 2 1 8 43 8 2]

[ 2 3 2 53 10 1]

[ 2 3 3 38 23 4]

[ 2 2 0 35 24 7]]

[[57 10 0 11 3 1]

[ 6 35 3 25 12 1]

[ 2 6 8 49 11 2]

[ 2 3 2 59 10 1]

[ 2 3 3 41 26 4]

[ 2 2 0 44 26 8]]

[[64 10 0 13 3 1]

[ 6 39 3 29 13 2]

[ 2 6 11 55 12 4]

[ 2 3 2 70 10 2]

[ 2 4 3 45 31 5]

[ 2 2 1 46 27 10]]

[[73 10 0 13 3 1]

[ 7 44 3 31 13 2]

[ 5 8 12 59 12 4]

[ 6 3 2 76 11 2]

[ 3 5 3 50 33 6]

[ 5 3 1 51 28 12]]

[[73 10 0 13 3 1]

[ 7 44 3 31 13 2]

[ 5 8 12 59 12 4]

[ 6 3 2 76 11 2]

[ 3 5 3 50 33 6]

[ 5 3 1 51 28 12]]

0.416666666667

Process finished with exit code 0

MFCC:

C:\Python27\python.exe "C:/Users/Thilak/Desktop/ml/pjt 3/pjt3code/mfcc\_music\_genre.py"

C:\Python27\lib\site-packages\scikits\talkbox\features\mfcc.py:108: RuntimeWarning: divide by zero encountered in log10

mspec = np.log10(np.dot(spec, fbank.T))

[[10 0 0 1 0 0]

[ 3 3 2 0 0 1]

[ 5 1 6 0 0 0]

[ 0 0 1 6 0 0]

[ 1 0 1 0 4 5]

[ 0 0 0 0 0 10]]

[[18 2 0 1 1 0]

[ 6 9 5 0 0 1]

[ 6 2 9 0 0 0]

[ 0 0 2 13 0 0]

[ 1 0 3 0 11 6]

[ 1 2 0 0 1 20]]

[[25 3 0 1 1 0]

[ 9 14 6 1 0 1]

[ 7 3 15 0 2 0]

[ 0 0 2 22 0 0]

[ 1 1 5 0 20 6]

[ 1 2 0 0 1 31]]

[[32 3 0 1 1 1]

[10 16 8 2 4 1]

[ 7 5 17 0 3 1]

[ 0 0 2 31 0 0]

[ 2 1 6 1 27 10]

[ 1 2 1 0 1 43]]

[[41 3 1 1 1 1]

[14 16 10 2 6 2]

[10 5 18 0 4 2]

[ 0 0 3 43 1 0]

[ 2 1 6 2 30 15]

[ 1 2 1 0 1 55]]

[[54 3 1 1 2 1]

[15 23 12 3 6 2]

[11 5 26 1 7 3]

[ 0 1 4 49 1 0]

[ 2 2 8 3 31 16]

[ 1 2 1 0 1 62]]

[[62 3 2 1 2 1]

[18 25 15 3 6 4]

[13 5 31 2 7 5]

[ 1 1 6 55 2 1]

[ 4 2 9 3 33 25]

[ 1 2 1 0 1 68]]

[[69 4 2 1 2 1]

[19 30 18 3 6 4]

[15 5 36 2 9 5]

[ 2 2 6 63 3 2]

[ 4 3 11 4 38 27]

[ 2 2 1 0 2 77]]

[[74 4 2 1 4 1]

[22 36 20 3 6 4]

[15 8 39 5 13 6]

[ 2 2 7 75 4 2]

[ 5 4 11 4 41 29]

[ 2 2 1 0 2 84]]

[[85 5 3 1 4 2]

[25 41 20 3 7 4]

[16 10 47 6 14 7]

[ 2 2 9 81 4 2]

[ 5 5 11 6 42 31]

[ 2 2 1 0 2 93]]

[[85 5 3 1 4 2]

[25 41 20 3 7 4]

[16 10 47 6 14 7]

[ 2 2 9 81 4 2]

[ 5 5 11 6 42 31]

[ 2 2 1 0 2 93]]

0.648333333333

Process finished with exit code 0