EC:447 Pattern Recognition and Machine Learning

Vowel-only versus consonant sound classification using EEG data corresponding to speech prompts

Submitted by:
Group K
Akarsh P
Sandeep Kotta
D R V Ramesh
Aditya Nishtala
Keerthan Shagrithaya

Department of Electronics and Communication Engineering
National Institute of Technology Karnataka
Surathkal,Karnataka, India

Contents

1 Data **1**

2 Dimensionality Reduction **1**

3 Gaussian Mixture Model **2**

3.1 Introduction 2

3.2 Code 2

3.3 Results 3

1 Data

The EEG data for this project was taken from the Kara One Database provided by the Department of Computer Science, University of Toronto. This database had mutlimodal data from 14 people, each subjected to approximately 165 phonemic/syllabic/word prompts. The multimodal data included EEG signals, audio signals and frames from Microsoft Kinect providing visual data. The different prompts present in the database were /iy/, /uw/, /piy/, /diy/, /tiy/, /m/, /n/, pat , pot, knew and gnaw.

For the problem of classifying whether a given prompt has a vowel-only sound or a consonant sound as well, we used prompts corresponding to /iy/ and /uw/ as vowel-only and those corresponding to /piy/, /diy/ and /tiy/ as consonants. Furthermore, we only used features extracted from the EEG data as input to our classifier. Along with the above mentioned data, the database had files with extracted features. In the feature file that we used, every prompt had 4 x 62 x 45 (4 segments x 62 channels x 45features/segment and channel) = 11160 features. The total number of trials corresponding to vowel-only sounds was 347 and that corresponding to consonant sounds was 522. Thus, our data matrix had dimensions 869 x 11160.

We divided the data matrix into training and testing in the ratio of 70%:30% respectively. Therefore, the training data had 247 vowel-only points and 372 consonant points and the test data had 100 vowel-only points and 150 consonant points.

2 Dimensionality Reduction

Since we had more features than data points for training, we had to either do feature selection or dimensionality reduction. We employed dimensionality reduction techniques.

Upon performing PCA, we observed that the variances corresponding to the directions of the principal components were very large in magnitude, $10^{56}, 10^{47}, 10^{46}, 10^{45}$ and so on in the decreasing order. This meant that most of the energy was concentrated in the first component alone and that the first component would be sufficient for classification. The values of the first component were analysed for vowel-only and consonant sounds and was found to be exactly the same for all data points irrespective of the class to which it actually belonged to. Thus, PCA wasn't very helpful in view of classification.

We then performed LDA on our training data matrix. Since LDA can output a maximum of number of classes -1 directions, we will get atmost 1 direction as output (because this is a binary classification problem). The figure on the next page shows the 1 dimensional distribution of the training data points after LDA. One can clearly identify two clusters. A smaller one at the left bottom and a larger one at the right top. The smaller one corresponds to the first 247 data points which are all vowel-only sounds. The larger cluster corresponds to the next 372 data points which are consonant sounds. This shows that this 1 dimension is sufficient for classification. Having obtained the vector mapping the training data to a single point, we use the vector to project the test data and convert the test data points to their equivalent 1 dimensional representations. The lower dimension training data is stored in .npz format as train.npz and the lower dimension test data is stored in .npy format as test.npy

Scikit-learn was used for doing PCA and LDA.

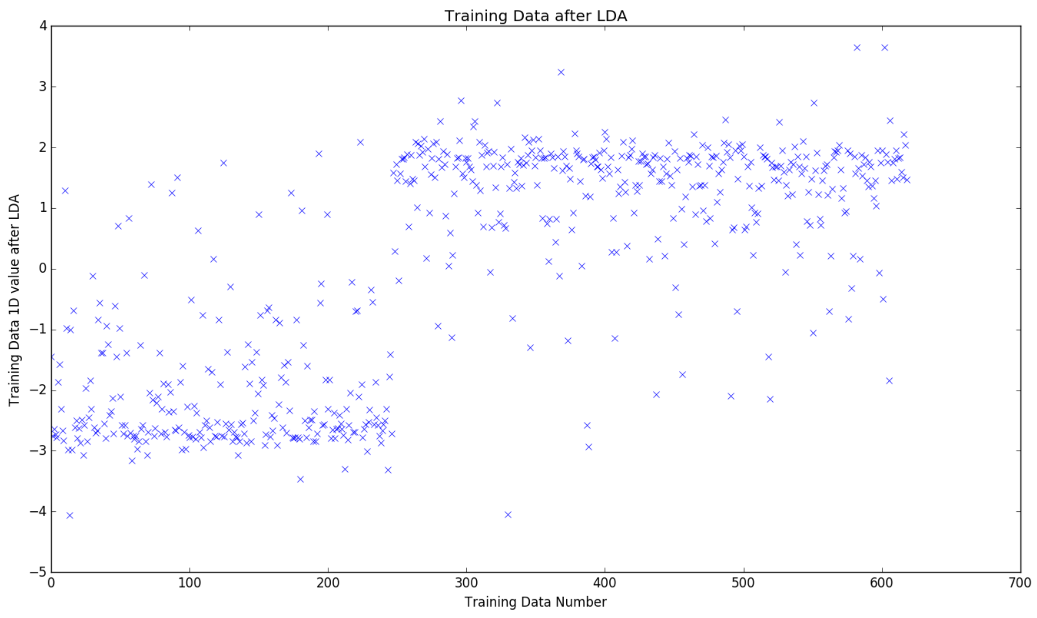


Figure 1: Training data after LDA

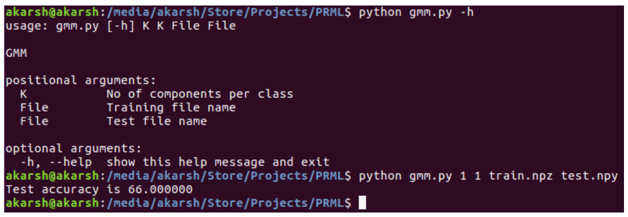
3 Gaussian Mixture Model

3.1 Introduction

Gaussian Mixture Model is popularly used for clustering data. The model states that the data is distributed according to the linear combination of multiple Gaussians with different means and covariances. Expectation-Maximization algorithm with soft assignment is used for estimating the weights, means, and covariances of the training data. Two GMMs are trained, one for each class. The only varaible component is the number of Gaussian components. This can vary from class to class. With the obtained parameters, the likelihood of the test data is found out for GMM corresponding to each class and multiplied with respective class priors to obtain value proportional to the posterior probability. We then compare the values obtained from each class and declare the output. The output is compared with the desired output and accuracy is computed.

3.2 Code

We have implemented the Expectation-Maximization algorithm with soft assignment in Python. The inputs to the program are the number of components required for consonant sound class, for vowel-only sound class, file name of training data and file name of test data. These should be given as command line arguments. For the training data, a .npz file is accepted and for test data a .npy file is accepted. The only output of the program is the test accuracy.



```
akarsh@akarsh:/media/akarsh/Store/Projects/PRML$ python gmm.py -h
usage: gmm.py [-h] K K File File

GMM

positional arguments:
  K           No of components per class
  File        Training file name
  File        Test file name

optional arguments:
  -h, --help  show this help message and exit
akarsh@akarsh:/media/akarsh/Store/Projects/PRML$ python gmm.py 1 1 train.npz test.npy
Test accuracy is 66.000000
akarsh@akarsh:/media/akarsh/Store/Projects/PRML$ 
```

Figure 2: Code input and Result

3.3 Results

This figure shows the accuracy of the GMM based classifier as a function of number of Gaussian components per class. We see that the accuracy doesn't increase by a lot for increase in the number of components.

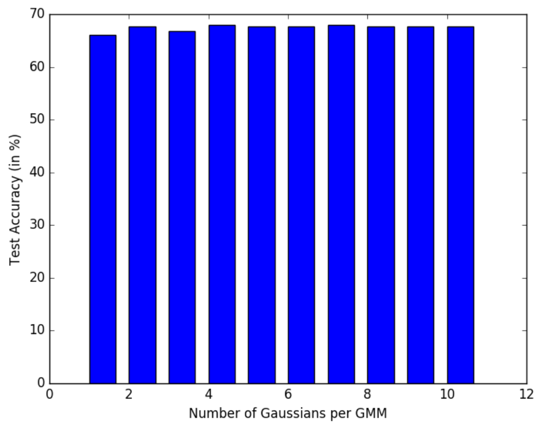


Figure 3: Test Accuracy vs Number of Gaussians per GMM

References

[1] Duda, Richard O., Peter E. Hart, and David G. Stork. Pattern classification. Vol. 2. New York: Wiley, 1973.

[2] Walt, Stéfan van der, S. Chris Colbert, and Gael Varoquaux. ”The NumPy array: a structure for efficient numerical computation.” Computing in Science & Engineering 13.2 (2011): 22-30.

[3] Jones, Eric, Travis Oliphant, and Pearu Peterson. ”SciPy: open source scientific tools for Python.” (2014).

[4] Pedregosa, Fabian, et al. ”Scikit-learn: Machine learning in Python.” Journal of Machine Learning Research 12.Oct (2011): 2825-2830.