Automated Semantic Tagging of Speech and Audio

B.Tech Project submitted in partial fulfillment
of the requirements for the degree of
Bachelors of Technology
in
CSE

by *Yashasvi Girdhar*
201101146
mohit.aggarwal@students.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2014

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this project,
titled "*Automated Semantic Tagging of Audio and Speech*"
by *Mohit Aggarwal*
with Roll No *201101164*,
has been carried out under my supervision for a partial fulfillment of degree.

November 15, 2014

Signature
Prof Kishore S. Prahallad
(Adviser)

Acknowledgements

We wish to express our sincere gratitude to **Dr Kishore S. Prahallad**, for providing us the opportunity to do our B.Tech project under his guidance. We sincerely thank him for his guidance and encouragement in carrying out this project work. He has been a constant source of motivation and encouragement for us. We thank him for all the initiative and zeal he filled us with throughout the project work.

We also wish to express my deep gratitude to **Professor Shailesh Kumar** for guiding us in this endeavor. We are thankful for his invaluable support, supervision and useful suggestions throughout this project work.

We would also like to thank **Professor Vikram Pudi** for his guidance and valuable feedback. His suggestions made our work easy and proficient.

Last but not the least, we're thankful and indebted to all those who helped us directly or indirectly in completion of this project work.

Abstract

The aim of the project is to use a machine learning algorithm to train a machine learning model to automatically tag the unknown speech and audio data into one of the semantic categories.

To get a thorough understanding of what is needed by the project, we are following the approach suggested by the research paper *"On using nearly-independent feature families for high precision and confidence "* by *Omid Madani, Manfred Georg and David Ross.*

We begin by showing that when very different sources of evidence such as text, audio, and video features are available, combining the outputs of base classifiers trained on each feature type separately, aka *late fusion*, can substantially increase the accuracy, compared to the performance of a single classifier trained on all the feature types, i.e., *early fusion*, or compared to the individual base classifiers.

We have used the *YouTube Multiview Video Games Dataset Data Set* released by Google, which consists of over 100k feature vectors extracted from public YouTube videos. These videos are labeled by one of 30 classes, each class corresponding to a video game (with some amount of class noise). We run different algorithms on this dataset and compare their results with the base result mentioned in the paper.

We then move on to *feature extraction* part wherein we research on the audio features and see if we can use them in our original problem. We mainly focus on understanding the 2 audio features - mfcc and spectrograph stream.

Finally we shift our focus to *feature selection*. We use spectral clustering technique to find better ways of fusion, i.e grouping the correlated features into some clusters and then, apply the classifier algorithm on each of these clusters, and finally combine the result from each classifier. We also apply some advanced Machine learning algorithms to cluster correlated features. We finally come up with algorithms that give up better accuracy as compared to the base results in the research paper.

Contents

Chapter 1

INTRODUCTION

The aim of the project is to devise a machine learning algorithm to train a machine learning model to automatically tag the unknown speech and audio data into one of the semantic categories.

In order to come up with this algorithm, we had the following initial requirements:

- We needed a labelled speech dataset as our training set.
- We needed some already established results on this dataset, so as to verify the correctness of the algorithm, that we would be using.

Therefore, we follow the approach suggested by the research paper “On Using nearly independent feature families for high precision and confidence” by Omid Madani, Manfred Georg and David Ross.

We divided our project into 4 phases :

Phase 1
We implement the approach suggested by the research paper to get a base understanding of our own problem. We implement 2 early fusion techniques on the Youtube Multiview Video Games dataset.

Phase 2
We implement 2 late fusion techniques and compare our results with the one mentioned in the research paper. We finally aim at proving that late fusion is better than early fusion.

Phase 3
We research on algorithms to extract audio features from the videos. We study the feature vectors used by the Youtube dataset and see if we can use them in our original problem.

Phase 4
We work on feature selection to find better ways of late fusion, and come up with algorithms that give better results.

We implemented the 4 phases and came up with a good solution to our original problem. Following sections describe in detail the work we have done in each phase.

Chapter 2

DATASET

We have used the *YouTube Multiview Video Games Dataset* (12 GB) released by Google. We chose 30 game titles at random, from amongst the most popular recent games. We treat each game classification as a binary 1-vs-rest problem. For each game, we collected 3000 videos that had the game title in their video title. Manually examining a random subset of such videos showed that about 90% of the videos are truly positive (the rest are irrelevant or do not contain gameplay). For each game, videos from other game titles constitute the negative videos, but to further diversify the negative set, we also added 30,000 videos to serve as negatives from other game titles. **The data, of 120,000 instances was split into 80% training, 10% validation, and 10% test.**

In Brief

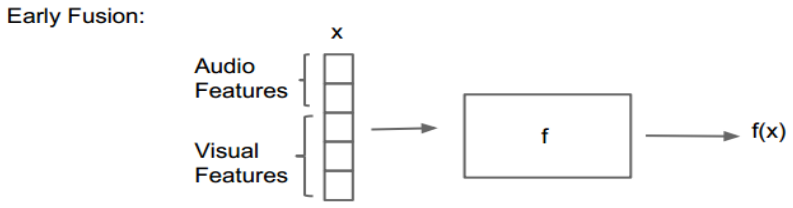
- The dataset consists of over 100k instances of the public YouTube videos.
- These videos are labeled by one of 30 classes, each class corresponding to a video game (with some amount of class noise).
- Each instance (video) is described by **three feature families** (textual, visual, and auditory), and each family is broken into subfamilies yielding up to 13 feature types per instance.
- Text feature family has 3 features:
    - text_description_unigrams
    - text_game_lda
    - text_tag_unigrams
- Audio feature family has 5 features:
    - audio_mfcc *(Dimensions-2000)*
    - audio_sai_boxes *(Dimensions-4230)*
    - audio_sai_intervalgrams *(Dimensions-7168)*
    - audio_spectrogram_stream *(Dimensions-1024)*
    - audio_volume_stream *(Dimensions-64)*
- Visual feature family has 3 features:
    - vision_hog_features (*Dimensions - 647*)
    - vision_cuboids_histogram (*Dimensions - 512*)
    - vision_hs_hist_stream (*Dimensions - 1024*)
    - vision_hist_motion_estimate (*Dimensions - 64*)
    - vision_misc (*Dimensions - 838*)
- The attributes in the feature vectors are real integer values.

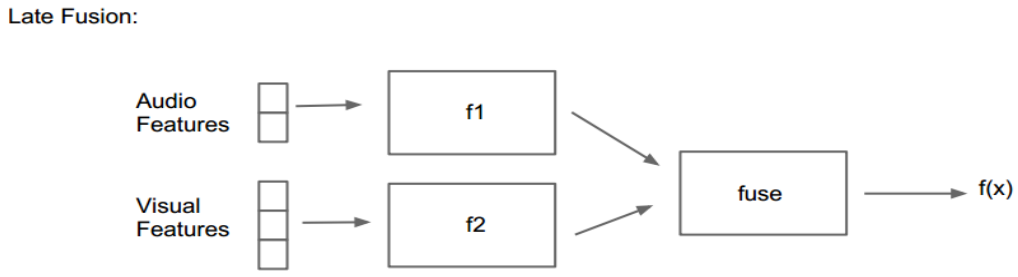
Chapter 3

EARLY FUSION AND LATE FUSION

Many applications, for example, in multimedia, provide diverse sets of feature families and distinct ways of processing the different signals. Given access to a number of different feature families, a basic question is how to use them effectively. Consider two extremes: training one classifier on all the features, aka *early fusion* or fusion in the feature space, versus training separate classifiers on each family then combining their output, aka *late fusion* or fusion in classier/semantic space. Training a single classier on all the families has the advantage of simplicity. Furthermore, the learner can potentially capture interactions among the different features. However, there are complications: one feature family can be relatively dense and low dimensional, while another very high dimensional and sparse. Creating a single feature vector out of all may amount to mixing apples and oranges. This can require considerable experimentation for scaling individual feature values and whole feature families (and/or designing special kernels), and yet, learning algorithms that can effectively integrate all the features' predictiveness may not exist. Furthermore, for a significant portion of the instances, whole feature families can be missing, such as absent audio or speech signals in a video. Training separate classifiers then combining the outputs, may lose the potential of learning from feature interactions across different modalities, but it offers advantages: one can choose appropriate learning algorithms for each feature family separately, and then combine them for best results.

In this section, we find that training distinct base classifiers offers an important benefit with respect to high precision classification. Feature families based on very different signals, for example, text, audio, and video features, can complement one another and yield independent sources of evidence.





Early Fusion

Methodology

In early fusion approach, we build a classifier that is trained on all the features appended together. We call this classier *Append*. We use the passive-aggressive online algorithm as the learner. This algorithm is in the perceptron linear classier family. We used efficient online learning because the feature vectors contain tens of thousands of dense features, and even for our relatively small problem subset, requiring all instances to fit in memory (as batch algorithms do) is prohibitive.

- There are total 13 features - 5 Audio, 5 Visual and 3 Textual.
- We train the base classifier on each of the feature families (with all its features appended together), and then use the test data to obtain the performance characteristics such as the precision.
- Total dataset size: 1, 20, 000 video instances
    - Train set (80%)
    - Validation set (10%)
    - Test set (10%)

Results

Visual Features

Feature	**True Positives**	**False Positives**	**Precision**
vision_cuboids_histogram.txt	6870	5060	0.58
vision_hog_features.txt	5400	6530	0.45
vision_hist_motion_extimate.txt	2964	8966	0.25
vision_hs_hist_stream.txt	4361	7569	0.37
vision_misc.txt	6182	5748	0.52

Audio Features

Feature	**True Positives**	**False Positives**	**Precision**
audio_mfcc.txt	3579	8351	0.30
audio_sai_boxes.txt	6920	5010	0.58
audio_sai_intervalgrams.txt	2863	9067	0.24
audio_spectogram_stream.txt	3220	8710	0.26

audio_volume_stream.txt	1232	10698	0.10

Textual Features

Feature	**True Positives**	**False Positives**	**Precision**
text_description_unigrams.txt	8112	3818	0.68
text_game_lda_1000.txt	10925	818	0.93
text_tag_unigrams.txt	10750	1180	0.90

Combined Features

Feature	**True Positives**	**False Positives**	**Precision**
Combined Audio Features	3232	8698	0.27
Combined Visual Features	4633	7297	0.38
Combined Audio and Visual Features	5120	6810	0.42

Observations

- Textual features gave better precision as compared to auditory or visual features.
- Combined Audio or Visual features gave a slightly better precision.
- Early fusion techniques gave average performance results. Textual features individually outperformed the combined features.
- We need to analyse the features individually in order to be able to understand the results of the experiments.

Late Fusion

Methodology

In *early fusion*, we were calculating the score for each classifier individually, and if it was greater than our threshold, we declared the instance to be belonging to that class (Binary Approach). In *Late Fusion*, we are not comparing the individual scores of the classifiers with our threshold. Instead, we first normalize the scores to get the probabilities of each classifier,

then we combine the probability scores of all the classifiers using the late fusion techniques and then compare it with the threshold to decide the class of an instance.

We implement 2 late fusion techniques -

- *NoisyOR* - False Positive Probability is simply the product of the false positive probabilities of base classifiers.
- *AVG* - Fusion using average of base classifier probability scores.

Results

	At a High Threshold	**At a Moderate Threshold**
Audio	0.046	0.093
Visual	0.12	0.47
NoisyOR	0.34	0.56
AVG (SUM)	0.49	0.71

Observations

- We can clearly see that in both the cases (audio and visual feature family), late fusion using NoisyOR or AVG gives far better results as compared to early fusion technique Append.
- Therefore we can conclude that the late fusion techniques give better results as compared to early fusion techniques.

Chapter 4

FEATURE EXTRACTION

In this section, we research on different features of the audio, and see if we can apply them in our original problem.

We mainly focused on 5 audio features:

- audio_mfcc
- audio_sai_boxes
- audio_sai_intervalgrams
- audio_spectrogram_stream
- audio_volume_stream

Each of these features required good amount of research in the field of signal analysis. However since our project is more focused on finding a machine learning algorithm to classify unknown speech and audio, we narrowed down our research to one audio feature - mfcc. This is the most widely used feature in audio feature extraction.

Mfcc representation

The *mel-frequency cepstrum (MFC)* is a representation of the short-term power spectrum of a sound. Sounds generated by a human are filtered by the shape of the vocal tract. This shape determines what sound comes out. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

Given below is the algorithm that we used to compute the mfcc representation of an audio signal.

- Frame the signal into short frames
- For each frame calculate the periodogram estimate of the power spectrum.
- Apply the mel filterbank to the power spectra, sum the energy in each filter.
- Take the logarithm of all filterbank energies.
- Take the DCT of the log filterbank energies.
- Keep DCT coefficients 2-13, discard the rest.

Algorithm in Detail

An audio signal is constantly changing, so to simplify things we assume that on short timescales the audio signal doesn't change much. This is why we frame the signal into 20-40ms frames. If the frame is much shorter we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.

The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates, different nerves fire informing the brain that certain frequencies are present. Our periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame.

The periodogram spectral estimate still contains a lot of information not required for Automatic Speech Recognition (ASR). In particular the cochlea can not discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason we take clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions. This is performed by our Mel filterbank: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations.

Once we have the filterbank energies, we take the logarithm of them. This is also motivated by human hearing: we don't hear loudness on a linear scale. Generally to double the perceived volume of a sound we need to put 8 times as much energy into it. This means that large variations in energy may not sound all that different if the sound is loud to begin with. This compression operation makes our features match more closely what humans actually hear.

The final step is to compute the DCT of the log filterbank energies.

Implementation
We implemented this algorithm in python. However since we didn't had the links for the youtube videos, whose features were present in the dataset, we could not verify this algorithm. This algorithm however can be used to extract the mfcc features for a given audio.

Chapter 5

FEATURE SELECTION

In this section, we focus on finding better ways of fusing selected features (aka late fusion) such that it improves our accuracy on the test set.

We use *Spectral clustering* technique to find better ways of fusion. We group the correlated features into some clusters and then, apply the classifier algorithm on each of these clusters, and finally combine the result from each classifier.

Spectral Clustering
Spectral clustering techniques make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. The similarity matrix is provided as an input and consists of a quantitative assessment of the relative similarity of each pair of points in the dataset.

Here we first compute similarity matrix of the features of the dataset which gives a similarity coefficient between any 2 features, i.e how correlated any 2 features are. We then perform dimensionality reduction by taking top k eigenvectors, and finally perform clustering in lower dimensions to get sets of correlated features.

How this applies to our problem
We have 5 audio feature families. We combine all the families to get a combined audio feature vector of ~14000 dimension for each data point.

Train set : 90,000 * 14,000

Test set : 10,000 * 14,000

We perform Spectral Clustering on the 14,000 audio feature vector to get clusters of correlated features.
Once we get clusters of correlated features, we segregate the data according to these clusters, build a model on each of these clusters and finally combine the output of each classifier via late fusion to get accuracy on the whole test set.

Algorithm for Spectral Clustering

Given a set of points $S = \{ s_1,....,s_n \}$ in $\mathbb{R}^l$ that we want to cluster into *k* subsets:

1. Form the affinity matrix $A \in \mathbb{R}^n$.
2. Define *D* to be diagonal matrix whose *(i,i)*-element is the sum of *A*'s *i*-th row, and construct the matrix $L = D^{-1/2}AD^{-1/2}$.
3. Find $x_1,x_2,....x_k$, the *k* largest eigen vectors of *L* (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [x_1x_2...x_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors in columns.
4. Form the matrix *Y* from *X* by renormalizing each of *X*'s rows to have unit length.
5. Treating each row of *Y* as a point in $\mathbb{R}^k$, cluster them into *k* clusters via K-means or any other algorithm (that attempts to minimize distortion).
6. Finally assign the original point s_i to cluster *j* if and only if row *i* of the matrix Y was assigned to cluster *j*.

Problems faced

- Generating full feature vectors from sparse vectors required huge memory. (~1 lakh instances with 14,000 dimension.)
- 90,000 instances with all audio featured combined took 8GB of space (in sparse form).
- Given such huge dimensions, we could not run this algorithm on normal systems due to memory constraints (4GB RAM).

Solution
We had to execute this algorithm on one of the audio feature families. We tested our algorithm on 2 audio feature families and came up with results that were better than the base results.

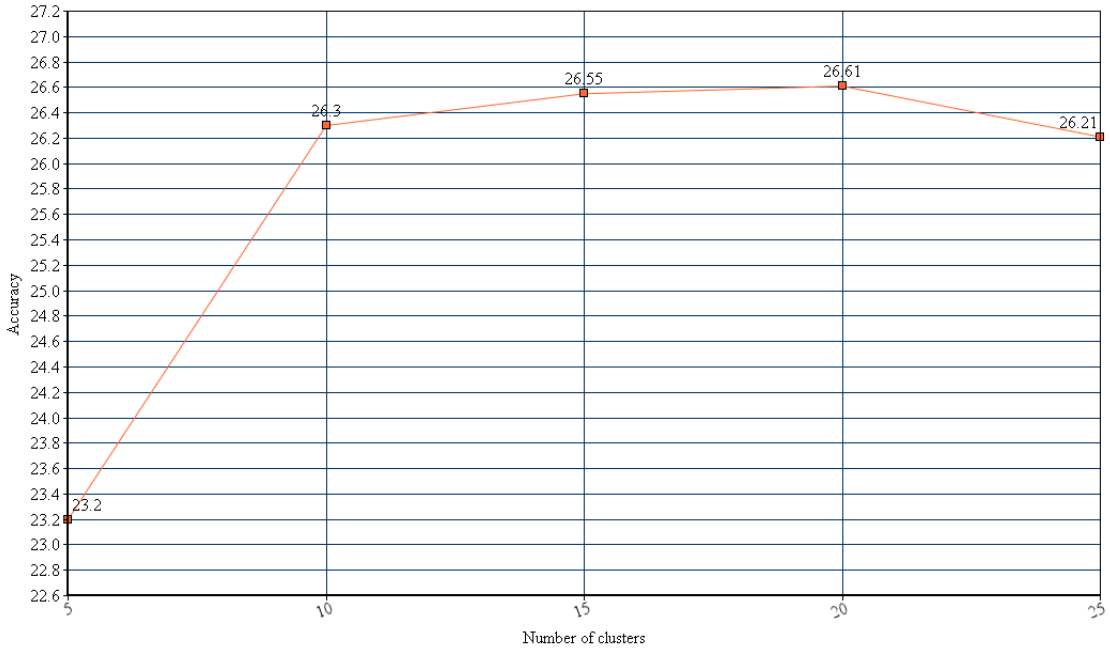
We ran our algorithm on a server with 16GB RAM so as to get the results for this huge dataset.

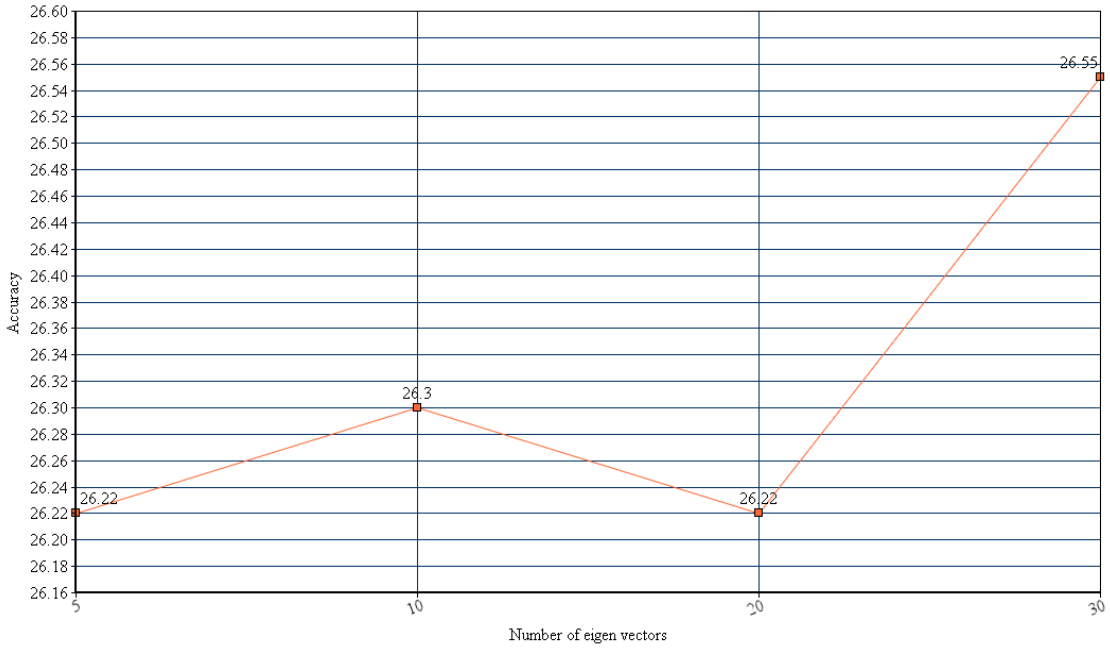
We had 2 free parameters -

1. Number of eigenvectors *#k*
2. Number of clusters we group the features into *#d*

Increasing both these parameters increased our accuracy.

Audio_volume_stream
Train set : 1,00,000 * 64
Test set : 11,930 * 64





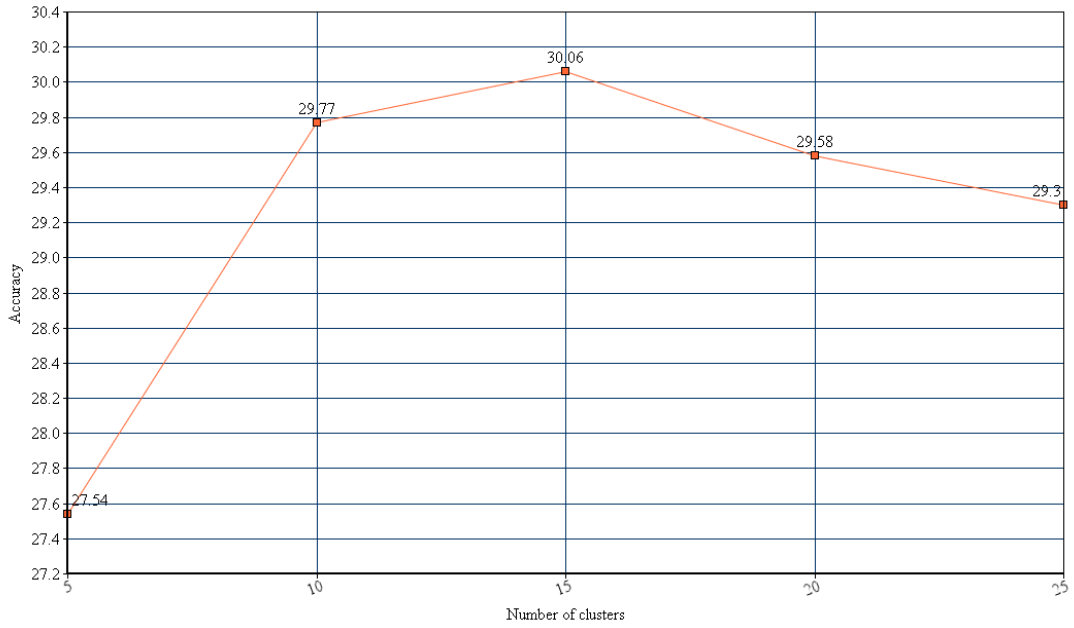
Here we get an average accuracy of 26% compared to 10% in early fusion approach. This implies *late fusion after feature selection* can significantly improve the accuracy.

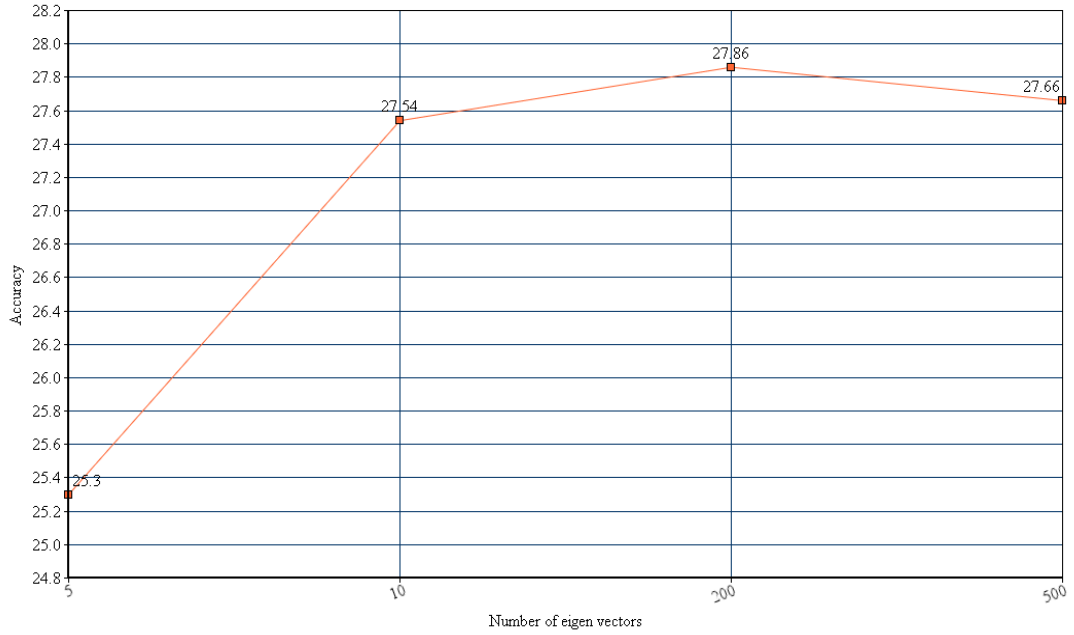
Audio_spectrogram_stream

Train set : 30,000 * 1024

(due to memory and time constraints, we had to reduce the dataset size)

Test set : 11,930 * 1024





Here we get an average accuracy of 29% compared to 26% in early fusion approach.

However we should notice that we got the 29% accuracy using train set, *which was 1/3rd the*

size of original train set (one used in early fusion approach). This implies that this approach performs fairly well, and would give far better results when used with the whole data set.

Conclusion

We can clearly see that using this approach increases our accuracy significantly. However this algorithm requires the entire dataset to be in main memory so that we can compute the affinity matrix for the features. This limits the usage of this algorithm to only small data sets due to time and memory constraints.

Chapter 6

FURTHER ANALYSIS

We have worked on different algorithms that can help classify unknown speech and audio with greater accuracy. We found the late fusion gives better results than the early fusion approach. We then worked on feature selection to find correlated features and perform late fusion on sets of data with similar features. This approach gave us better precision but required greater memory and time. Below, we explore 2 more approaches that can help us give even better results, keeping in mind the time and memory constraints.

Approach 1

We apply Fisher discriminant Analysis (FDA) on the initial dataset (1,00,000 * 14,000) to reduce it to a dataset with only dominant features.

We then use Spectral Clustering to group correlated features in this reduced dataset, and perform late fusion on them.

Approach 2 - Two level Clustering

We perform Spectral Clustering to find clusters of correlated features. Suppose we get 50 clusters.

Now for each cluster we take the PCA dimension within the cluster. Since they were highly correlated with each other we will have a minimal loss in PCA projection.

Now we have on Principal feature representing each of the 50 clusters.

We now do FDA on this 50 x 50 matrix and do second level clustering and build models on those.

We then use late fusion techniques to combine the output from each model, and accordingly classify the test instances.

Bibliography

1. *On using nearly-independent feature families for high precision and confidence* - Omid Madani, Manfred Georg, David Ross
2. *Improving Automatic Music Tag Annotation Using Stacked Generalization Of Probabilistic SVM Outputs* - S.R. Ness, L.G. Martins
3. *Combining feature kernels for Semantic Music Retrieval* - Luke Barrington, Mehrdad Yazdani, Douglas Turnbull, Gert Lanckriet
4. *Automatic Tagging of Audio - The State-of-the-Art* - Thierry Bertin, Douglas Eck, Michael Mandel
5. *Automatic Semantic Tagging of Speech Audio* - Yves Raimond, Chris Lowis, Jonathan Tweed
6. *Online Passive Aggressive Algorithms* - Koby Crammer, Ofer Dekel, Yoram Singer
7. *On Spectral Clustering: Analysis and an algorithm* - Andrew Y.Ng, Michael I. Jordan, Yair Weiss