Detecting Audio/Video Asynchrony

Alex Perelygin and Michael R. Jones

December 16, 2011

1 Introduction

Asynchrony between the audio and video track in streaming video can cause user frustration with the service but often goes unreported. Providers of on-demand internet streaming media such as Netflix reach millions of users and thus are concerned with supplying correctly aligned audio/video content. Our project will attempt to detect A/V misalignment in standard (non-streaming) movie clips. We expect that if certain time-dependent video and audio features are correlated, we should be able to predict the misalignment of a video and audio track by comparing the progression of these audio and video features over a segment of the movie clip.

2 Assumptions and Restrictions

Misalignment in audio is perceivable when audio leads video by more than 15 milliseconds and lags video by more than 45 milliseconds [1]. Based on this find we formulate the duration of our time-window during data collection to capture the variation in A/V features over time.

We will restrict misalignment of the audio track to constant values over the duration of the movie clip – i.e. not time-varying of progressive. We will also limit our dataset to movie clips in which the audio track has been delayed.

3 Features

We selected audio and video features based on their ability to reflect continuous states over time. Each feature was computed over a 1 second segment, consisting of 25 frames. The value and first derivative value of each feature was used, since in many cases the change from the last value of the feature is the most important point. A total of 1680 features per 10 second window were collected.

3.1 Video Features

We used a 10-bin image intensity histogram and image intensity statistics (mean and standard deviation), since the change of these over time corresponds to how much the image is changing. We reasoned this could correspond to audio shifts, and is used in other research as a synchrony measure which is the video equivalent of audio energy [2]. The correlation coefficient between video frames was picked for a similar reason, as low correlations would imply major changes in the scene. The binary image area and binary image centroid were used to detect the proportion of the scene that was "bright" versus "dark", and where the center of the bright area was. It was done by converting the frame image into a binary image using a threshold calculated by the **graythresh** function in Matlab. This was reasoned to be a more precise formulation of a specific scene characteristic, brightness. The change in this value was reasoned to also correspond to audio shifts over time.

Also, face detection was performed using the OpenCV Viola-Jones algorithm [3] on each frame. The feature value was 1 if any faces were detected in the image, and 0 otherwise. This was reasoned to correspond with the speech track.

3.2 Audio Features

The root mean square and the max/min range of the audio signal over a frame of video was used as a general audio feature, which we reasoned would correspond to a rough approximation of the audio track for each second segment.

The standard deviation of the waveform, the zero crossing rate, and the Mel-frequency Cepstral Coefficients (MFCCs) were all used to identify similar segments of audio, as described by Gillet, Essid, et al [4]. The MFCC's were calculated using Matlab code by Daniel Ellis [5]. The standard deviation and zero crossing rate should correspond to periodic audio elements and noise, respectively. The MFCCs provide a parameterization of the human speech signal [6], so they are able to provide segmentation of speech in an audio track. Since human detection of audio-video synchrony is often based on the alignment of speech with visual cues such as lip-movement, we reasoned these coefficients could provide a rough indicator of certain common video contexts. Typical speech-recognition research combines the standard 13 MFCCs with 13 delta (first derivative) and 13 delta-delta (second derivative) MFCCs [2,7]. These correspond to dynamic features of the speech signal, used both for speech recognition and speaker verification. The mean, standard deviation, and max/min range of these 39 coefficients were used.

A 5-bin average power of the audio waveform was computed from its periodogram. The power of audio signal and the change in power should similarly correspond to shifts in the video scene, as is noted by Chollet et al. [6]

4 Method and Implementation

Our approach to this project consisted of using the feature set we developed to train a supervised learning algorithm to predict misalignment. The basic premise was that the 10 second window used would contain a correlation of the changing video features over time to the changing audio features, and this would constitute a detectable pattern. The goal of training would then be for the learning algorithm to find this pattern, match it with a set of known alignment delays, and learn a prediction of the correct classification based on this.

4.1 Feature Extraction

We implemented the data set generation (extraction of the features to be used) using the VideoFileReader class of the Vision toolkit in Matlab. This was done by stepping through each frame in the videos used, using the frame rate to collect the per-second feature values, and storing every 10 second window as a column in a features matrix. Misalignment was artificially introduced by stepping through a fixed number of frames in the audio track prior to beginning the feature collection, introducing a delay equal to the time spanned by the number of frames skipped.

4.2 Training and Testing

We trained and tested using features extracted from 2 movies: "Pirates of the Caribbean" and "Imagine Me & You," standard movies with regular characteristics. These movies both involved significant amounts of dialogue, leading us to believe our speech recognition features could come into play. "Pirates of the Caribbean" involved significantly more action sequences than the other movie, so we wished to compare the effects of this on our learning. We trained and tested on both movies separately and together, in order to compare the performance of our learning in both a specific and a more general setting.

4.3 Linear Classifier

Initially, we attempted to use linear regression to create a linear prediction of alignment delay based on the feature values. We implemented it using stochastic gradient descent, since the large data set would have made batch gradient descent very slow. Our first attempts to run the algorithm resulted in exploding weight values that clearly never reached the cost-minimizing value. We were able to fix this by using a much lower learning rate (around 0.0001 instead of 0.1 as we picked originally), and by dividing the feature values by their maximum values, thereby standardizing them. With these practical considerations, we were able to minimize the cost function and converge to the optimal set of weights. However, the alignment delay values predicted by the classifier were extremely inaccurate, returning results that appeared random and were orders of magnitude away from the correct value.

4.4 Neural Network

Based on the poor results offered by the linear classifier, we deduced that there was no linear relationship between the feature values and the true alignment delay in the 10 second windows we were testing over. Since we did not have any deeper knowledge about what form of relationship there could be, we decided to use a neural network to automatically find and learn from the correlation. Thus, we implemented a recursive backpropagation neural network using a single hidden layer, which predicted a single output: 1 for misaligned, 0 for aligned. We used a sigmoid activation function at all layers, to create a clear boundary between aligned and misaligned activations. We initially did this using batch gradient descent, with a learning rate of 0.1 and a regularization factor (lambda) of 0.1. However, testing on the Iris flower data set [8] revealed this was underfitting the data, so we reduced the regularization factor to 0.001 and were able to classify the data well. Convergence was still slow on our audio/video feature set, however, so we decided to use minFunc with the L-BFGS method [9] to find the weight values rather than using gradient descent.

4.5 Whitening and PCA

This algorithm was still not able to learn from the data very well, as it would not be able to decrease the generalization error on the training set. We deduced that this was due to the feature values being of differing signs and magnitudes, which resulted in neural network weights that were also very skewed. Therefore, we applied principal components analysis with whitening to our data prior to training the neural network on it. The main idea here was to reshape the data into a fairly uniformly scaled hypersphere so that it would be simpler to learn from. We initially used 100 whitened principle components with a neural network consisting of 50 hidden neurons to train, as we found this could find the lowest generalization error with appropriate regularization factors, but we varied this to test different possibilities. Finally, we used the liblinear SVM with L2-regularization L2-loss [10] to classify the data as a comparison to the neural network that was our main learning algorithm.

5 Results

The first result that we noticed was the large effect that the regularization factor had on the generalization error. As the parameter value approached 0 (or indeed reached), the accuracy of the trained neural network predictions on the training set approached 100%. However, with larger values, the accuracy of the predictions on the training set decreases before reaching a steady level around 60%, as the chart below shows. However, the accuracy on the test set did not change significantly as the generalization error changed. The testing accuracy remained around 50%, or equivalent to random guessing. This implied that, for a training set size of

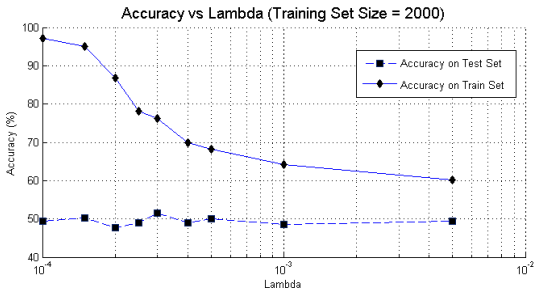

Figure 1: Regularization factor vs Accuracy

2000, the neural network could determine an appropriate pattern in the data, but this was not useful to predicting previously unseen examples. Next, we chose a specific regularization factor value (0.0005) and varied the training set size to determine if the poor performance was due to overfitting. As the chart below shows, the generalization error does increase as training set size increases, but there is no corresponding change in testing accuracy.

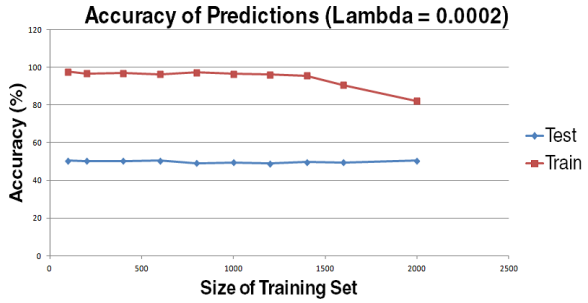

Figure 2: Training Size vs Accuracy

This led to the same conclusion as from the first result: the neural network was able to do an excellent job of finding a non-linear pattern in the data and match it to the output, but this was not useful for making future predictions. Finally, we compared our neural network's performance to that of an SVM, using several different principle component numbers (100, 200, 400, 600 and 800). Our neural network was better at finding patterns in the data and classifying the training set than the SVM was, but they were equally poor at making predictions on unseen examples. We also noticed that the generalization error of the neural network on the training set increased as the number of principal components increased; meanwhile, the generalization error of SVM increased as the number of principal components increased. We believe this can be attributed to the regularization factor having a greater weight on cost function as the size of the weight matrix increases. This conclusion is reinforced by the fact that for lambda equal to zero, the accuracy of the neural network remains near 100% for any number of principal components. Conversely, increasing the input dimension provides the SVM with greater facility to linearly separate the training examples.

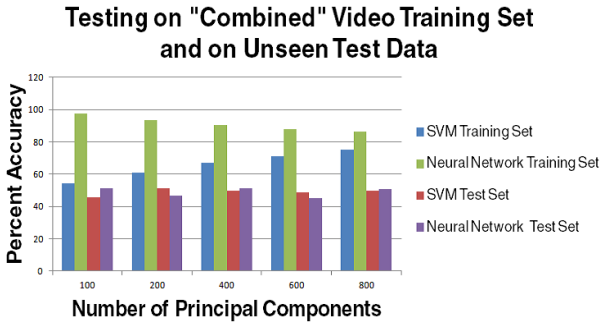

Figure 3: SVM vs Neural Network

6 Conclusion and Future Work

We were not able to find any useful correlation between the features used and the alignment of the video and audio tracks. The neural network was able to find a pattern, but this pattern was not useful for future prediction. We believe this was due to the inapplicability of general audio-video correlation measures to alignment, due to the extremely wide range of scenarios in videos. A way to mitigate this inapplicability would be to introduce specific features for specific scenarios. An example would be a feature to detect an action such as a door slamming or an explosion, paired with an audio feature to detect

the expected audio response. The data set would need to be restricted to include these scenarios. Then, a bank of specific feature-scenario pairs could be constructed to create more specific cases where a clear expected video-audio pair occurs.

7 Acknowledgments

We would like to thank Andrew Maas for generously offering advice and for providing the Matlab code for feature whitening and PCA.

References

[1] "Relative Timing of Sound and Vision for Broadcast Operations". Advanced Television Systems Committee report, IS-191, 2003.

[2] Hervé Bredin and Gérard Chollet. 2007. Audiovisual speech synchrony measure: application to biometrics. EURASIP J. Appl. Signal Process. 2007, 1 (January 2007), 179-179.

[3] Dirk-Jan Kroon. Retrieved from "http://www.mathworks.com/matlabcentral/fileexchange/29437-viola-jones-object-detection"

[4] Gillet, O.; Essid, S.; Richard, G.; , "On the Correlation of Automatic Audio and Visual Segmentations of Music Videos," Circuits and Systems for Video Technology, IEEE Transactions on , vol.17, no.3, pp.347-355, March 2007

[5] Ellis, D. "PLP and RASTA (and MFCC, and inversion) in Matlab," "http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/"

[6] H. Bredin and G. Chollet. Audiovisual speech synchrony measure: Application to biometrics. EURASIP Journal on Advances in Signal Processing, 2007.

[7] Hossan, M.A.; Memon, S.; Gregory, M.A.; , "A novel approach for MFCC feature extraction," Signal Processing and Communication Systems (ICSPCS), 2010 4th International Conference on , vol., no., pp.1-5, 13-15 Dec. 2010

[8] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science

[9] Mark Schmidt. Retrieved from "http://www.di.ens.fr/~mschmidt/Software/minFunc.html"

[10] Machine Learning Group at National Taiwan University. Retrieved from "http://www.csie.ntu.edu.tw/~cjlin/liblinear/"