MP4: Gaussian Mixture Model

Group: Akhil Alapaty & Ojus Deshmukh

I. Introduction.

We had two different feature vectors in this MP - Cepstrum, and PCA. We calculated the Cepstrum feature vector for the data. This is done by converting the raw pixel feature vector signal into a vector of frames using the sig2frames function. Next, we took the Inverse Fourier Transform of the logarithm of the frames vector's magnitude spectrum. The Cepstrum feature vector is more effective than the Raw Pixel feature vector because it catches features such as pitches and frequencies in a speech signal, since they are part of the logarithmic representation. The cepstrum is generally characterized as containing information that depicts the rate of change in a signal's separate spectrum bands.

The first feature vector being called on the images is the Raw Pixel vector. As we know, the raw image vector contains minimally processed data from the original images. We reshaped and resized the original image into different vectors in order to compute various sizes of raw pixel feature vectors. Next, we implemented the PCA vector through the eigenvalue decomposition algorithm given to us. The PCA extraction method is better than the Raw Pixel method because as the name suggests, it only measures the data in terms of the principal components, which are defined as those with the most variance. Each eigenvector has a respective eigenvalue, which is the number that contains how much variance exists in the data in that given direction. The principal component would be the eigenvector that has the highest corresponding eigenvalue. This was carried out by applying the Gram matrix transformation on our data.

The two learning metrics we used for this MP were KNN and GMM. We used KNN mainly to help us calculate accuracy, by grouping the test data by person. It generally

incorporates a “leave one out” strategy, but in this case, the training and test sets were completely different. This cycles through all 40 images until accuracies are computed for each of them using the nearest neighbor algorithm with k = 10.

GMM generates multiple probability distributions for each training data set that it is given. It then uses those probabilities to find the most likely match for its test data.

II. Methods

Our first critical transform was calculating the cepstrum. In Lines 46 and 50 of mp4.m, we have two for loops that run through cepstrum for both the training and the test data. The cepstrum is calculated entirely through a cepstrum.m function, which does the following calculations:

$$c[n] = \mathcal{F}^{-1}\{\log|\mathcal{F}\{x[n]\}|\}$$

For each window, the cepstrum is mathematically calculated as such:

$$c[n] = \sum_{n=0}^{N-1} \log\left(\left|\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}\right|\right) e^{j\frac{2\pi}{N}kn}$$

Our second transform was calculating PCA. After calculating the gram matrix of the image data, we found the top N principal components of the data. We calculated the minimum number of eigenvectors required for keeping 95% on the energy. We found that we needed 68 eigenvectors to keep 95% of the energy, then selected the eigenvectors corresponding to the 68 largest eigenvalues, and projected our image data onto them. See lines 12-15 of pca.m. To project the data, we multiplied the gram matrix by the selected eigenvectors and then multiplying by the square root of the eigenvalue matrix. See lines 30 - 34 of pca.m.

$$: \ U = Z * V * S^{-1}$$

Our third critical transformation was creating a GMM for the input data. We created the GMM using the train data entirely within gmm_train.m. We first initialized Mu, Sigma, and the weight of the matrix. Gaussian Mixture Models are usually used for data clustering. Fitted GMM’s cluster by assigning query data points to the normal components in order to maximize

the posterior probability. The GMM implementation can be found in the gmm_train and gmm_eval Matlab files.

$$p(x_i;\Lambda) = \sum_{k=1}^{K} p(k|\Lambda)p(x_i|k;\Lambda) \;=\; \sum_{k=1}^{K} w_k p(x_i|k;\Lambda)$$

*This equation image was taken from the MP4 walkthrough by Amit Das

III. Results

```
--------- Person ID Accuracy: Visual Only -----------

ans =

  5×1 table

          Acc
          ___

    A      90
    B      80
    C      30
    D      80
    Avg    70

--------------------

--------- Person ID Accuracy: Audio Only -----------

ans =

  5×1 table

          Acc
          ____

    A       50
    B       70
    C       60
    D       90
    Avg   67.5

--------------------
```

```
--------- Person ID Accuracy: Audio + Visual -----------

ans =

  5×9 table
```

	w_01	w_02	w_03	w_04	w_05	w_06	w_07	w_08	w_09
A	90	90	90	90	91	91	93	95	97
B	80	80	80	80	80	80	80	81	85
C	30	30	30	34	36	45	54	58	62
D	80	80	80	81	82	87	90	99	99
Avg	70	70	70	71.25	72.25	75.75	79.25	83.25	85.75

```
--------------------
```

IV. Discussion

As we look at our Audio + Visual table, we notice that as w increases, the Average steadily increases as well (70 for w =.01 while 85.75 for w = .09). At first, we couldn't identify a reason for this occurrence, but a hypothesis is that as w increases, the GMM likelihood's impact also increases because it is raised to a higher power while the k-NN posterior's impact decreases. Since the averages increase as GMM's impact increases, we believe that GMM likelihood has a more powerful role than k-NN posterior when it comes to identifying the Fusion data.