AFC Coding Assignment 2

Vishaal Udandarao

March 2020

1 Pre-processing and F0 Contours Extraction

The given dataset contains 24 speakers in total. Each speaker has 60 speech samples each. The distribution of class samples is shown in the table below.

Class	Number of samples per speaker
Neutral	4
Calm	8
Happy	8
Sad	8
Angry	8
Fearful	8
Disgust	8
Surprise	8
Total	60

1.1 F0 contour extraction through windows

Firstly, all the features for all the 60*24=1440 samples are extracted. For every speech sample, the F0 contours are extracted by using the autocorrelation method. This is done by firstly splitting each speech sample into individual frames, and then retrieving the F0 contour value corresponding to every frame.

The splitting is done in the following manner. For every speech signal, it is ensured that we retrieve 100 frames per second. The window size and overlap windows are set according to the individual sampling rates of the signals.

1.2 F0 statistics extraction

Once, we have the F0 contours corresponding to every speech sample, we extract 7 F0 statistics from each F0 contour. They are as listed below:

- f1: F0 mean
- f2: F0 standard deviation

- f3: F0 range
- f4: F0 quantile 1
- f5: F0 quantile 3
- f6: F0 IQR
- f7: F0 median

Hence, each speech sample is represented by a 7 dimensional vector: [f1, f2, f3, f4, f5, f6, f7]

1.3 Sampling a reference neutral dataset

The original IFN paper makes use of a reference neutral dataset to train the neutral models (GMMs or HMMs). This is done to get the fitness scores of the emotional and neutral speech samples with respect to the neutrally trained model.

Since we cannot make use of the reference neutral dataset that is mentioned in the paper, we sample our own reference neutral dataset from the dataset given. For every speaker, there are 4 neutral speech samples. The 1st neutral speech sample of every speaker is taken separately to construct the reference neutral dataset. Therefore, after sampling, there are 24 samples in the reference neutral dataset and 72 neutral samples in the remaining training dataset.

1.4 Training, Validation and Test splits

Once we subsample a reference neutral dataset, we are left with 72 neutral samples in the entire dataset. Further, there are 56 emotional samples per speaker, therefore, the total number of emotional samples in the dataset is 56*24=1344.

To ensure that there is no bias with respect to speaker identities, the training, test and validation splits are done in the following manner:

1. Traning and Validation splits: We consider the first 18 speakers for training. Therefore, the total number of neutral samples in the training+validation split will be 54. To ensure no class imbalance, 54 samples are drawn randomly in an IID manner uniformly across all 18 speakers. Therefore, the training+validation set contains 108 samples (with equal neutral and emotional samples). Finally, the training set and validation set splits are divided into 72 and 36. This is again done ensuring uniformity per speaker.

2. Test split: We consider the last 6 speakers for testing. There are 18 neutral samples in the test set and 336 emotional samples.

2 Results and Analysis

For comparison, I have implemented three algorithms for neutral-emotion classification. A brief explanation of all three algorithms is given below:

- **Optimal feature normalization**: Once we have the training data with 7 F0 statistics as a feature vector, we scale/normalize each feature vector in a speaker-dependent manner. We first compute the reference neutral dataset normalizer $F0_{ref}$; it is simply the mean of all the F0 mean features across the reference neutral dataset. We then compute the per-speaker neutral factor $F0^{s}_{neu}$(for speaker s); it is the mean of all the F0 mean features across the neutral subset of the speaker s. Now, each speaker's feature arrays are scaled by a factor of $\frac{F0_{ref}}{F0^{s}_{neu}}$ to achieve optimal normalization.
- **No normalization**: The feature arrays are taken as is. No feature normalization is done.
- Iterative Feature Normalization: The IFN algorithm as described in the paper is implemented. First, we train 7 GMM models to produce fitness scores for every training sample. GMMs with 2 gaussians are considered. Once, we retrieve GMM likelihood/fitness scores for each training sample, we use these 7 fitness scores as features to classify into neutral or emotional classes. A simple binary LDA (linear discriminant analysis) classifier is implemented as the binary class classifier. In every iteration of the IFN, we only consider neutral predictions whose classification confidence is greater than 0.6. After every iteration, we recompute the normalization parameters on the basis of the thresholded neutral predictions. After every iteration, we check the percentage of files in the entire training dataset that change labels. The algorithm converges when the percentage of files that change labels in subsequent iterations is less than 5%.

2.1 Results

The following table depicts the accuracy, precision, recall and F1-score across all the three algorithms. All the results are averaged over 5 different runs (the paper averages results over 400 different runs, this was not possible due to time constraints).

Algorithm	Accuracy	Precision	Recall	F1-score
No normalization	0.58/**0.43**	0.59/**0.45**	0.55/**0.4**	0.57/**0.58**
Converged IFN	0.97/**0.95**	1/**0.95**	0.94/**1**	0.97/**0.97**
Optimal normalization	1/**0.98**	1/**1**	1/**0.97**	1/**0.98**

Table 1: Results of all algorithms. The first value is on the validation set, the second value is on the test set.

Based on the results, we can clearly see that the algorithm without any normalization performs the worst. The algorithm normalized with the optimized parameters performs the best on the validation as well as the test sets. This is expected since the scaling parameters constrain the neutral samples in the dataset to have more discriminative features due to the likelihood scores obtained from the trained GMMs. The converged IFN algorithm (convergence details and explanation given in next section) also performs very well and has almost matched the performance of the optimal normalization parameters. On further analysis, we see that this might be due to two major reasons:

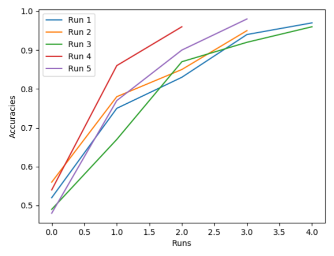
- The normalization parameters learnt by the IFN algorithm are almost equal to the optimal normalization parameters
- The size of the training, test and validation sets are very small and hence the algorithm was able to converge easily without any optimization issues.

2.2 Convergence Analysis

As mentioned above, every algorithm was run for 5 different times, for giving the averaged results. The IFN algorithm convergence condition was set to a threshold of 5% label change i.e. between consecutive IFN iterations if the percentage of files that changed label predictions was less than 5%, then the algorithm terminates and the convergence condition is met.

Across the 5 iterations, the convergence condition was achieved within 4 iterations. The details of the individual runs are mentioned below in Table 2.2

The convergence plots (runs vs percentage changes and runs vs accuracies) are also shown below:



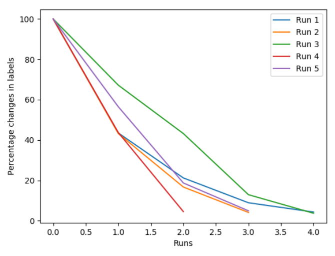


Figure 1: Left: Runs vs accuracies, Right: Runs vs percentage changes

On analyzing the plots and the convergence table, we clearly see that the IFN algorithm converges to the optimal normalization parameters (ap-

Runs	**Iteration**	**Percentage change**	**Accuracy**
	1	100	0.52
	2	43.5	0.75
1	3	21.2	0.83
	4	8.9	0.94
	5	4.3	0.97
	1	100	0.56
2	2	43.2	0.78
	3	16.7	0.85
	4	4.1	0.95
	1	100	0.49
	2	67.2	0.67
3	3	43.2	0.87
	4	12.9	0.92
	5	3.7	0.96
	1	100	0.54
4	2	43.7	0.86
	3	4.5	0.96
	1	100	0.48
5	2	56.5	0.77
	3	18.7	0.9
	4	4.9	0.98

proximately) on every run (at most requires 5 iterations for convergence). The accuracies monotonically increase with each iteration and the percentage change in the labels also monotonically decreases with iterations. Hence, we conclude that the algorithm given in the IFN paper seems to work well for the features considered and for the RAVDESS speech dataset.
.