Iterative Feature Normalization: A case study on the RAVDESS dataset

Aditya Chetan
Computer Science and Applied Mathematics
Indraprastha Institute of Information Technology, Delhi
aditya16217@iiitd.ac.in

Abstract

In this report, I have summarised my implementation of the Iterative Feature Normalisation (IFN) method given by Busso et al. [4]. IFN was proposed as a method for normalising acoustic features in datasets where the speaker is known, for improving performance on Affect Recognition. In this report, I have discussed my methodology for the implementation of the paper, the design decisions and assumptuions that I made. Since the original dataset used by the authors of IFN was not publicly available, I have generated results of the publicly available RAVDESS dataset created by Livingstone and Russo [6], and available publicly. This report was created as a part of an assignment for CSE5XX Affective Computing taught at IIIT Delhi during Winter 2020.

1 IFN Algorithm

In this section, I have discussed the exact steps that I followed for implementing the Iterative Feature Normalisation procedure. For studying this procedure, I referred to Busso et al. [4]. However, a lot of the specifics of the algorithm were not described in detail in this paper. Hence I also had to read up Busso et al. [3] and Busso et al. [3] for obtaining more concrete details about the implementation of the framework. The steps I followed were as follows:

1. **Creating a reference corpus**: Busso et al. [4] have used a reference corpus for training reference GMMs. The likelihoods obtained from these GMMs are used for further training an LDC classifier. This reference corpus is also used for estimation of the normalisation constants (S^s_{F0}). However, the exact corpus used in the paper was not publicly available. Hence, to create a reference corpus, I randomly selected one audio sample from the *neutral* audio samples of each speaker in the dataset. Hence, I ended up with 24 neutral audio samples, one for each unique speaker in my dataset. This became my reference corpus. The leftover dataset became my emotional corpus.
2. **Pre-processing the dataset**: The variant of IFN discussed in Busso et al. [4] requires the F0 contour of every audio sample in the dataset. Busso et al. [3] detailed that in order to extract F0 contour, they had used the PRAAT software developed by Boersma and Weenink [1]. PRAAT's API exposes the auto-correlation method used for computing F0 contours in audio files. Since I was conding in PYTHON, I made use of the PARSELMOUTH library [5], as they have PYTHON bindings for PRAAT functions. The output was the F0 contour for each audio file, with a window of 40ms, hop size of 10ms and overlap of 30ms, just as described by Busso et al. [3]. Both the reference and emotions corpus were preprocessed using this method.
3. **Feature extraction**: The F0 contours would have different dimensions for different files. Busso et al. [4] relied on statistics computed from F0 contours for training their classifiers. These statistics have been detailed by Busso et al. [3]. I referred to their work for writing

PYTHON implementations of the different statistics described there. For my main results, I picked the same features as used by Busso et al. [4].

4. **Training Neutral GMM models**: Using the features extracted from the reference corpus, a GMM model was trained on each feature individually. The number of components for each GMM model was set to 1, as Busso et al. [3] clearly stated that the number of components had no effect on the performance of the model. At the end of this step, I had as many GMM models as the number of features.
5. **A single IFN run**:
   (a) In each IFN run, first I sub-sample instances from the emotional class, to obtain the same number of instances as the neutral class, to prevent class imbalance.
   (b) Then, I split this under-sampled dataset into train and test set. Features extracted from the train set are used to extract likelihood scores from the trained GMMs. These likelihood scores are then used as features to train the LDC classifier.
   (c) The LDC classifier is used to predict the outcome on the test set.
   (d) The likelihood of the LDC is subjected to a threshold (henceforth referred to as `neu_threshold`), and the labels obtained in this way are used for deciding labels for the training set. These labels are only used for calculation of normalisation constant as described by Busso et al. [4], and not for training anything.
   (e) The number of files whose labels have changed frmo previous iteration are calculated. If this number if larger than 5%, the IFN run is terminated.
6. **Multiple runs**: IFN is run in this manner for 400 times, and results calculated are averaged over all runs.

Note: The Assignment suggested that we take the first 18 speakers in RAVDESS as training set and last 6 speakers as testing set. However, I do not think that this was possible in any way. IFN makes the assumption that not only do we have a healthy balance of emotional labels in the training set, but also expects the presence of a minimum number of samples from every speaker. In fact, if the LDC ever predicts no neutral samples, Busso et al. [4] pick a minimum number of the highest likelihood samples as neutral. Hence, I have followed the split mechanism described above.

2 Results & Observations

In this section, I have discussed the results that I obtained, by replicating experiments from Busso et al. [4] wherever possible. I have also made observations about the differences and have given a discussion about possible reasons for it. Additionally, hyperparameter selection is described here.

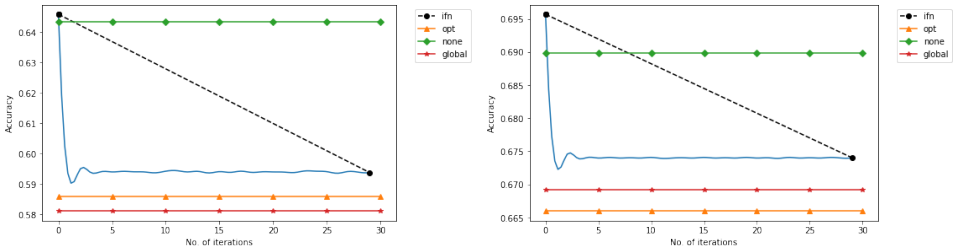
2.1 Training accuracy over iterations

I compared the training accuracy of IFN method across different iterations. Result is shown in Figure 1. This plot has been generated by averaging the accuracy in each iteration over all the 400 runs of IFN. Since each run of IFN can terminate in different number of iterations, the accuracy obtained in the last iteration of an IFN run is used to pad it to make all the IFN runs to be of equal length. Then the average was calculated for each iteration. In both training schemes, we observe more or less the same trends.

If we compare this plot with the Busso et al. [4, Fig. 3], two observations can be made:

- The trend between different normalisations does not match the results of Busso et al. [4].
- Accuracy does not seem to improve over iterations of the IFN. In fact, it dips in the earlier iterations, and then oscillates at a lower value.

In my opinion, these differences in results are because of a smaller dataset and a smaller number of samples per speaker in the training and test set. The reason I state this is due to similarity of the accuracy curve to the curve plotted for EMO-DB dataset [2] in Busso et al. [4]. The authors have reported that a possible reason for this could be lower average number of samples per speaker (~53). In our case, this number at the time training is only 4-5, as at the time of training, we sub-sample from the emotional dataset to maintain a class balance between neutral and emotion classes. The number



(a) Neutral Scheme (b) Conventional Scheme

Figure 1: Accuracy over training iterations for different normalisation schemes.

of neutral samples per speaker is 3 for the emotional corpus (total neutral samples in RAVDESS per speaker are 4 and one is taken for reference corpus).

2.2 Performance comparison of various normalisation schemes

Table 1 compares the performance of different normalisation schemes described in Busso et al. [4], across the training schemes described by them. These results have been generated by averaging the results on the test sets obtained during 400 runs of the IFN.

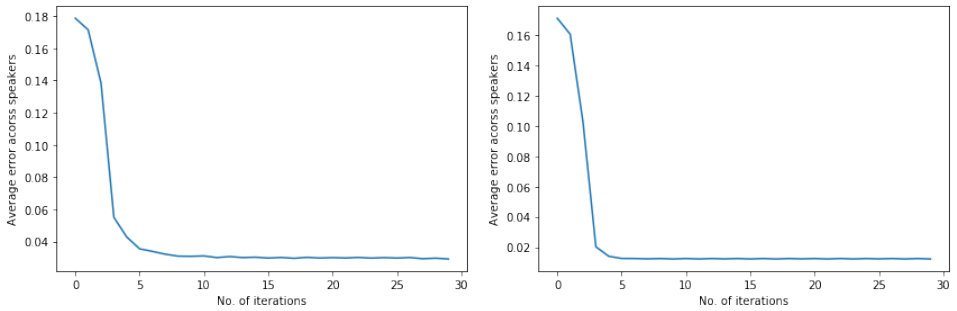
Here we can clearly see that the trend given in Busso et al. [4] is not followed with respect to the performance of various normalisation schemes. No normalisation performs better than IFN. I could not find any concrete reson for this, except for the fact that the size of our dataset is extremely limited and or referece corpus is also sampled from our emotional corpus, both of which is different setting from Busso et al. [4].

	neutral				conventional			
	P	R	F1	A	P	R	F1	A
ifn	0.721	0.594	0.522	0.594	0.731	0.674	0.652	0.674
global	0.702	0.581	0.503	0.581	0.729	0.669	0.644	0.669
opt	0.707	0.586	0.513	0.586	0.726	0.666	0.642	0.666
none	0.751	0.643	0.597	0.643	0.734	0.69	0.674	0.69

Table 1: Average results for different normalisation schemes

2.3 Quality of normalisation constants

Busso et al. [4] have shared that the error (average absolute difference) in the prediction of their normalisation constants, increases as the iterations proceed. I performed a similar analysis for the both schemes of training, visualised in Fig. 2.



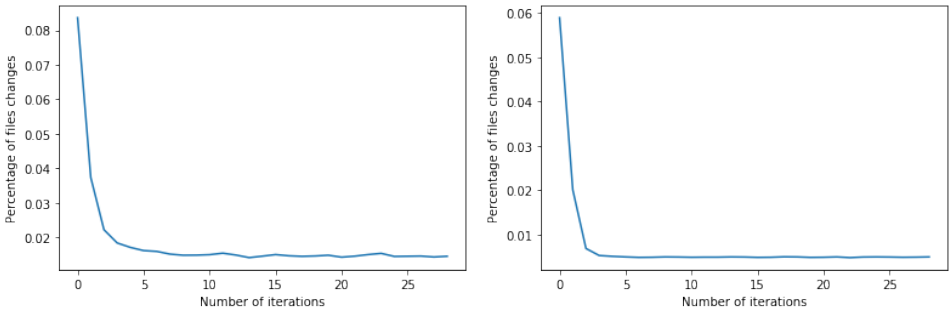
(a) Neutral Scheme (b) Conventional Scheme

Figure 2: Error in estimation of normalisation constants across training iterations

However, in my case, I could see a reduction in error. This may be because the trained model is overfitting on the training set, and therefore the value of optimal and computed values of the constants come out to be equal as most of the neutral samples in the training set are classified correctly.

2.4 Percentages of files changed in every iteration

Lastly, Busso et al. [4] have analysed the percentage of files that undergo a change in each iteration. This is also used as a convergence criteria by them. It is expected that the number of files that switch their labels accoring to the LDC's predictions, will decrease as we approach optimal normalisation constants.

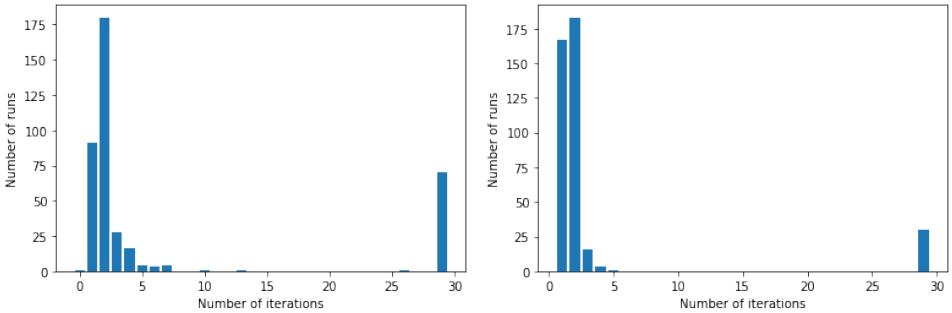


(a) Neutral Scheme (b) Conventional Scheme

Figure 3: Percentage file change along IFN iterations

Results are shown in Figure 3. For completeness, I have performed this analysis by averaging the percentage file change across all the 400 runs.

Additionally, I also analysed the distribution of the number of iterations that the IFN takes to converge across the 400 runs. I set a maximum limit of 30 iterations for this, i.e., if IFN does not converge in 30 runs, I terminated that particular run. Results have been visualised in Figure 4.



(a) Neutral Scheme (b) Conventional Scheme

Figure 4: Distribution of number of iterations taken for convergence by IFN

The results show that IFN takes varying number of ierations to converge, but for the most part is able to converge within 30 iterations. However, there are some cases where it has not converged too, especially in the neutral scheme. This is contrary to the results obtrained by Busso et al. [4], where it was convering in all cases. I could not pinpoint a reason for this, but mostly I felt it was due to the smaller size of the training data, and choice of features. For this small dataset, IFN seemed to be particularly sensitive to the features chosen and the selection of hyperparameters.

Selection of hyperparameters

For the most part, I tried to follow the same settings for hyperparaters as described by Busso et al. [4], however, in some cases, because of completely senseless results, I have made changes to a few hyper-parameters.

- **Likelihood threshold for neutral samples**: Busso et al. [4] have used a likelihood threshold of 0.7 (`neu_threshold`) over the likelihoods predicted by their LDC. However, they also mention that this threshold was decided by observation, to ensure that a decent number of neutral samples are predicted while maintaining decent accuracy. In our case, the dataset is smaller, the LDC learns on even fewer number of instances. Hence, to accommodate the changes because of our dataset, I have taken this threshold as 0.45.
- **Ratio of train-test split**: Busso et al. [4] used a train-test ratio of 7:3. However, with this ratio, it was not possible to perform a stratified split keeping both speakers and labels in mind. Hence, I took a 65:35 ratio.

2.5 Challenges & Assumptions

In this section, I have enumerated some of the challenges I faced because of ambiguities in Busso et al. [4] and the assumptions I had to make to overcome them.

1. **Voiced Segment features**: Busso et al. [4] refer to Busso et al. [3] for details about the features they have extracted. However, I could not find a description of voiced segments in Busso et al. [3]. Intuitively it seemed to me that the non-zero elements in the F0 contour are the voiced segment features. I also noticed that the number of voiced segments in the files according to PARSELMOUTH were equal to number of non-zero elements. I also read online that in an unvoiced region of audio, there is an unvoice phoneme, and F0 does not exist for that region. Hence, I assumed that voiced segments refers to non-zero elements and proceeded to extract features by considering only the non-zero elements as the F0 contour.
2. **Cumulative Normalisation or Original Normalisation**: Busso et al. [4] do not make it extremely clear whether in each iteration of IFN, the F0 contour of the previous iteration is to be scaled with the new constant or the original F0 contour is to be scaled. Initially, I was doing the former, but it later I realised that it did not make sense to me as at test time, the constant would have to be applied on the original F0 contour and not the cumulatively modified F0 contour. Additionally, this was leading to blowing up of constant values and leading to numerical errors. Hence, I switched to normalising the original F0 contour.
3. **Ambiguity in analysis of Busso et al. [4]**: The analysis and results generated in Busso et al. [4] seem very ambiguous. For instance, take the analysis of percentage of file changes across iterations of the IFN. I experienced that IFN takes different number of iterations to converge in different runs, depending on what sub-sample is chosen for training. However, from the plot of Busso et al. [4], it is not clear whether they have done an amortized analysis, or whether they have cherry-picked one particular run. This also made it difficult to benchmark and check the correctness of my own implementation. However, I have tried to take rational assumptions about these things and have elaborated on them in the Results & Observations section. (Section 2).

References

[1] Paul Boersma and David Weenink. Praat: doing phonetics by computer [Computer program]. Version 6.0.37, retrieved 3 February 2018 `http://www.praat.org/`, 2018.

[2] Felix Burkhardt, Astrid Paeschke, M. Rolfes, Walter Sendlmeier, and Benjamin Weiss. A database of german emotional speech. volume 5, 01 2005.

[3] C. Busso, S. Lee, and S. Narayanan. Analysis of emotionally salient aspects of fundamental frequency for emotion detection. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4), 2009.

[4] C. Busso, A. Metallinou, and S. S. Narayanan. Iterative feature normalization for emotional speech detection. In *ICASSP 2011*, 2011.

[5] Yannick Jadoul, Bill Thompson, and Bart de Boer. Introducing Parselmouth: A Python interface to Praat. *Journal of Phonetics*, 71, 2018. doi: https://doi.org/10.1016/j.wocn.2018.07.001.

[6] Steven R. Livingstone and Frank A. Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PLOS ONE*, 13(5), 05 2018. doi: 10.1371/journal.pone.0196391. URL `https://doi.org/10.1371/journal.pone.0196391`.