Speech emotion recognition: feature extraction and classification

Alessandro Bonadeo
Politecnico di Torino
Student id: s302157
s302147@studenti.polito.it

***Abstract*—In this report we propose a possible solution to the Speech Emotion Recognition (SER) classification problem. In particular, our approach consists in splitting the audio signals into short-term windows called frames, extracting different features from each of them and then computing global statistics on the results. Two traditional machine learning methods have been selected to perform the classification task.**

I. Problem overview

Speech Emotion Recognition (SER) is the task of recognizing the emotional aspects of audio data distinguishing between a set of different emotion. The given dataset consists of a collection of audio recording of phrases pronounced by different person with different intonation and representing one of the possible emotion considered. In this problem the total spectrum of the emotion is represented by seven different ones:

- *angry*
- *disgusted*
- *fearful*
- *happy*
- *sad*
- *surprised*
- *neutral*

We have two different dataset:

- the *development* set: it contains 9597 labeled audio records. This portion of the given data will be used for training our algorithm.
- the *evaluation* set: containing 3201 unlabeled records, is going to be used to evaluate our classifier.

From an high level exploration of the development set we observe that the problem is not well balanced, with the class *surprised* that is three to four time less numerous than the other six. Looking more specifically to the single data points we can observe that, for each one, the sampling rate is 8 kHz and the average length is 2.72 seconds. The fact that we have such a various numbers of durations (Fig.1) suggest us that we must find a vector representation of the same length for every audio signal as input in order to train our models.

For intuition purposes we can take a glance at two different plot representation of an audio signal: one in time domain (Fig 2) and one the frequency domain (Fig 3). This differentiation is going to guide us also in the process of feature extraction. The waveform presents a null amplitude interval at the end and the beginning corresponding to initial and final silence in the recording. The spectrum instead suggest us that most of the energy of the wave is concentrated at the low frequencies.

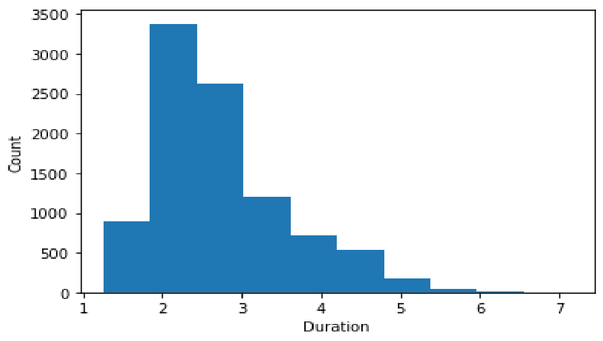

Fig. 1. Distribution of the durations of the recordings

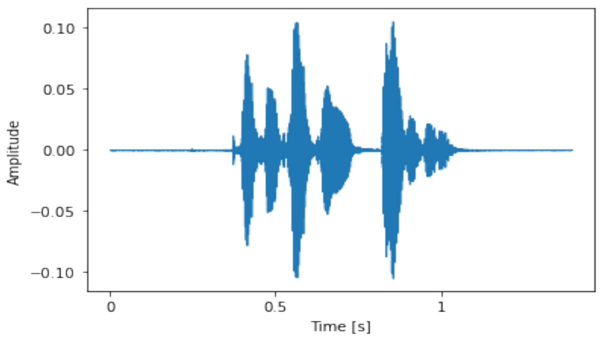

Fig. 2. Waveform

II. Proposed approach

A. Preprocessing

The preprocessing phase consists mostly in extracting relevant features from each recording. First we divide the signal into short-term windows called frames. The motivation behind this, is that we want to overcome the non-stationary nature of speech signal and get a good approximation of the frequency contour. By doing that we assume that frequencies in a signal are constant over a very short period of time and we apply the discrete time Fourier transform to each of this frame. Then we multiply each frame by the Hamming window function in order to reduce spectral leakage and improve the signal's

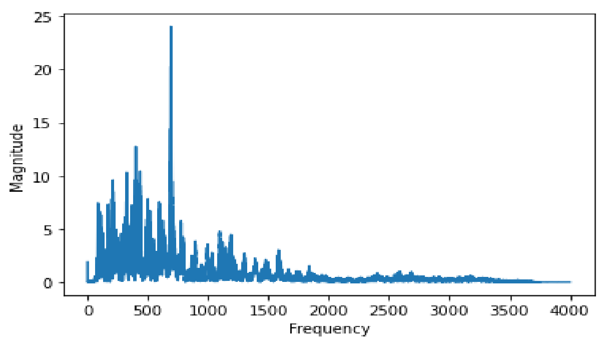

Fig. 3. Power spectrum

clarity.
As we mentioned above, we're selecting the features we're going to use distinguishing between their original domain; in particular we have chosen: energy (Fig 4) and zero crossing

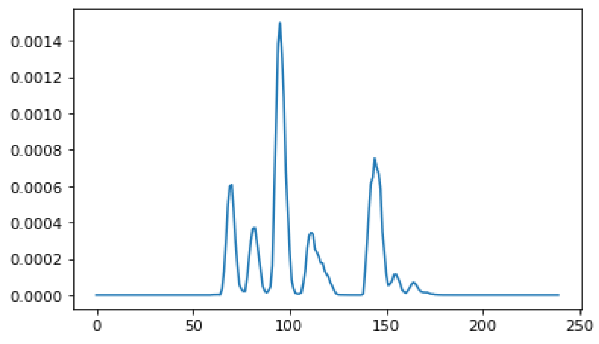

Fig. 4. Energy

rate (Fig 5) in time domain and spectral centroid (Fig 6) and

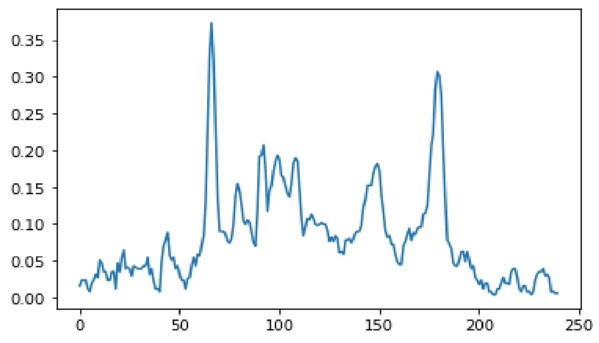

Fig. 5. Zero Crossing Rate

MFCCs (Mel Frequency Cepstral Coefficients) (Fig 7) in the frequency domain. A typical number of MFCCs for this type of task is 26 [1] so we end up with a total of 29 features. Since each feature is computed in each of the frame we extracted before, and since different audio files present different number of frames (for example in the figures above we have an audio signal with 250 frames, see x axis) we compute two different global statistics for each of this vector: mean and standard

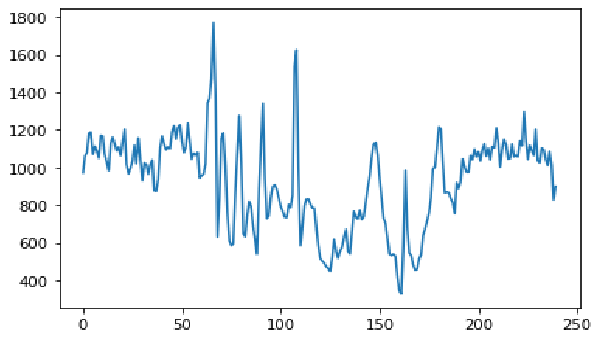

Fig. 6. Spectral centroid

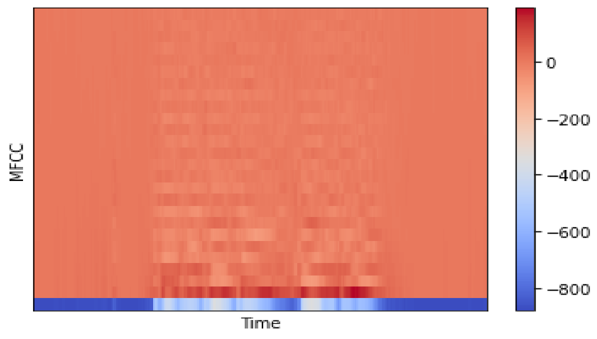

Fig. 7. MFCCs: 26 row matrix, one for each coefficient

deviation. At the end these statistics are going to be our final feature vector coefficients for a total of 58.

B. Model selection

We tested three algorithm:

- Random forests: it is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.
- SVM: the aim of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (with N being the dimension of the features vector) that separates the data records. When the internal structure of the data is non linear kernel methods are used to overcome the linear nature of the SVM.
- K-Nearest Neighbor: this is a more simple algorithm than the previous two and it assign a point to a specific class looking at its K nearest training points (usually using a majority voting scheme). This is going to be used as a baseline method.

C. Hyperparameters tuning

After a first run of the three algorithm with the *scikit learn* default hyperparametrs, using a simple 80/20 train/test split, due to the lack of performance of KNN, we decided to focus just on random forest and SVM. The latter seems already to slightly outperform the former. To determine in a precise and

Model	Parameters	Values
Random Forest	*max_dept*	{None, 10, 50, 100}
	n_estimator	{100, 250, 500}
SVM	*kernel*	{rbf, sigmoid}
	C	{1, 5, 10, 50, 100, 500, 1000}

TABLE I
HYPERARAMETERS

robust way which one is better we performed a K-Fold cross validation over two different set of hyperparameters using a grid search approach.
The different combinations for both algorithm are generated from the set of parameters defined in Table 1.

III. RESULTS

After the first run of the K-fold cross validation (with K=5) we observe that SVM, with a f1 score of around 0.71 still outperform random forest which reached a score of around 0.67. SVM has {'*kernel*': rgb, 'C': 10} as optimal configuration while random forest perform at best with {'*max_depth*': None, '*n_estimator*':500}. After these results, looking closer at how different SVM hyperparameters configurations ranked against each other, we notice that the *rbf* kernel performed better than the *sigmoid* one for each value of *C*. We decided then to run again a cross validation just for the *C* parameter expanding its values range around the optimum that we just found. We get a slightly better performance with *C* equal to 12 and f1 score of around 0.72 on the remaining test set. Figure 8 shows the f1 score for the different values of *C* in the grid. We can

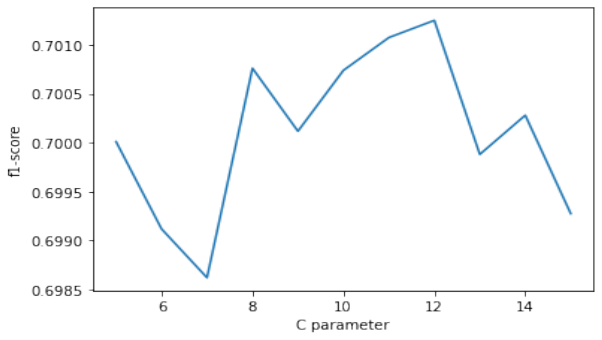

Fig. 8. C hypeparameter

also visualize for each class the different correct and wrong predictions thanks to the confusion matrix representation (Fig 9), where the following is the dictionary corresponding to the used label encoding: {0:'Angry', 1:'Disgusted', 2:'Fearful', 3:'Happy', 4:'Neutral', 5:'Sad', 6:'Surprised'}

IV. DISCUSSION

After having selected the best model based on the evaluation process we try it on the evaluation dataset obtaining a f1 public score of 0.721. This is fairly better than the given

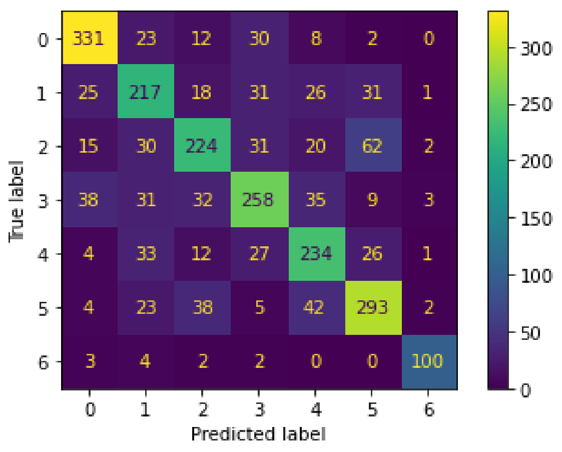

Fig. 9. Confusion matrix

baseline. We think that the good performances of the classifier are due to the fact that it has been done a careful work in the feature extraction phase selecting empirically the most effective an representative of the human voice. Furthermore we have chosen to use two of the most performing algorithm for this type of task considering traditional machine learning methods.
A further step to achieve better results would be following the deep learning route: Long Short Term Memory (LSTM), Convolutional Neural Networks (CNNs), Hidden Markov Models (HMMs) and Deep Neural Networks (DNNs) [2]. These methods don't need any feature summary statistics and MFCCs can be directly used as input of the algorithm.

REFERENCES

[1] M. S. Likitha, S. R. R. Gupta, K. Hasitha, and A. U. Raju, "Speech based human emotion recognition using mfcc," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 2257–2260, 2017.
[2] K. Venkataramanan and H. R. Rajamohan, "Emotion recognition from speech," *CoRR*, vol. abs/1912.10458, 2019.