Machine Learning approaches to Digit Vocal Classification

Sara Montese
Politecnico di Torino
S308482@studenti.polito.it

***Abstract*—In this paper, we introduce two possible Machine Learning approaches to the digit vocal classification problem. The solution compares different classification models, Random Forests and Support Vector Maching, over the Free Spoken Digit Dataset. By leveraging both time and frequency-based features of the audio files, both models achieve promising results.**

***Index Terms*—Machine Learning, Random Forest, SVM, Vocal Classification**

I. Problem overview

The proposed task is a classification problem on a dataset inspired by the Free Spoken Digit Dataset[1]. The dataset is composed of 2,000 recordings of numbers from 0 to 9 with English pronunciation. Each recording is a mono wav file and the sampling rate is 8 kHz. The objective of this study is to correctly identify the digit being uttered in each recording. The data have been distributed uniformly in two separate collections:

- a *development* set, containing 1500 recordings with ground-truth labels;
- a *evaluation* set, containing 500 recordings without labels.

The problem is well balanced: for each of the classes there are 150 samples. We used the development set to build a classification model capable of predicting the digit number that would be assigned to each record of the evaluation set. An important step was to carry out data exploration to derive an overview of the data. First of all, we started by visualizing the distribution of the audio lengths. As shown in Figure 1, we can notice that audio signals have different lengths, some of them have leading and trailing silence intervals.

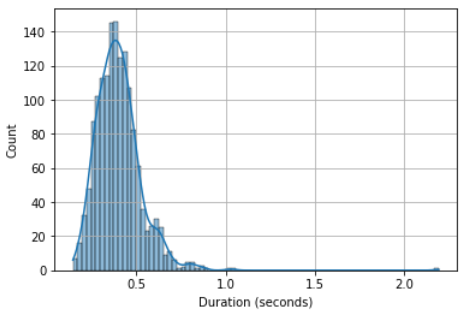

Fig. 1. Length distribution of audio signals

[1]Dataset website: https://github.com/Jakobovski/free-spoken-digit-dataset

II. Proposed approach

A. Data preprocessing

The training data in *.wav* format cannot be input directly into the model. We have to load the audio data from the files and process them in order to reach the quality and the format the model expects. By analysing the length distribution of the signals, we observe that the 95th percentile of audio lengths is around 0.62 seconds (Figure 2); we consider as outliers observations whose length lies outside the interval (around 75 observations).

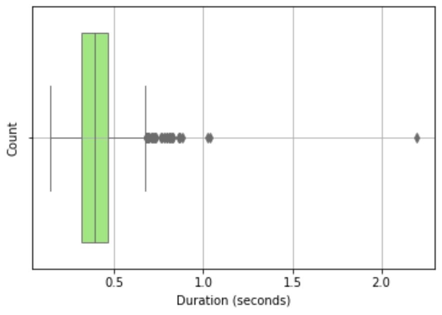

Fig. 2. Boxplot of audio signal lengths

We proceed by considering anything below 10 db as silence, and so removing leading and trailing silence from all signals. As a result, we got that the 95th percentile of the lengths was around 0.51 seconds. Figure 3 shows the new shape of the distribution.

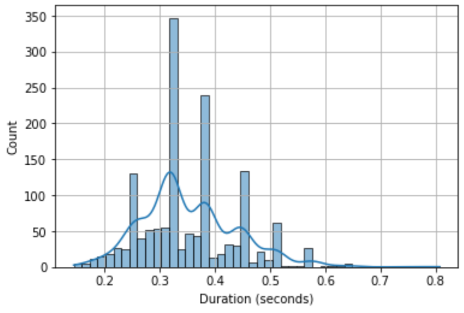

Fig. 3. Length distribution of audio signals after silence trimming

Since the audio data provided cannot be understood by models directly, we have to convert it into an understandable format by building a vector representation of the data. This mainly involves extracting several features out of our initial representation. Since we are dealing with audio signals, we can choose to work:

- On the *time* domain. The features are obtained by splitting the signal into chunks and characterize them by means of statistical measures.
- On the *frequency* domain. The features depend on the frequencies contained in the signal. This involves reshaping the signal using a transformation function (e.g. Fourier transform) and work on its spectrum of frequencies.

Since both time and frequency domains contain useful information regarding the recordings, we can leverage both by using the *spectrogram* of each signal. In a spectoral representation of audio signals, we get time on x-axis and different frequencies on y-axis. Values in the matrix represent different properties of audio signal related to particular time and frequency (e.g.. amplitude, power, etc)

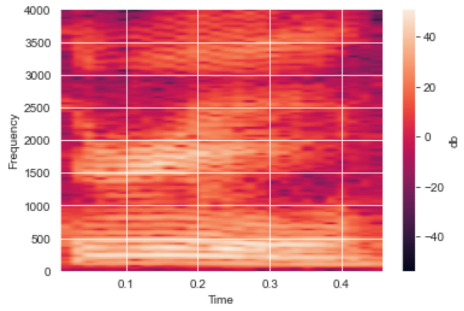

Fig. 4. Spectrogram of a signal of the datase (on logarithmic scale)

To extract features from a spectogram of given signal, we divide it into N x N sub-matrices of nearly identical shape. Later, we compute mean and standard deviation of these sub-matrices and consider them as features sets. The number of sub-matrices is considered an *hyperparameter* for classifier. Then, we carry out a grid search to check the optimal configuration of this hyperparameter that results in better performance of the two models. As shown in Figure 5, both models are stable in the neighborhood of 13, so we can select 13 as initial number N of bins.

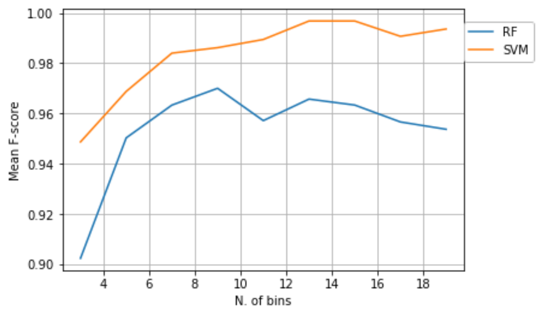

Fig. 5. Model evaluation on different numbers of bins

B. Model selection

We will now discuss more in depth the two models proposed for the classification task:

- *Support Vector Machine*: this algorithm applies a transformation to the data using a kernel function. It tries to find the hyperplane which maximizes the margin of errors between predicted and actual values that are tolerated. This model assumes that the data are in a standard range, so it requires feature vectors normalization.
- *Random Forests*: this algorithm belongs to the category of *ensemble* techniques because it combines multiple decision trees (trained on different portions of the training set and subsets of features) to classify data. This is generally done to enhance the performance and to avoid overfitting. Since RFs work on one feature at a time, there is no need to normalize the dataset.

C. Hyperparameters tuning

To find the best configuration of hyperparameters for our models, we have performed a grid search considering different combinations of parameters. The search space is summarized in tables I and II. To tune our models, we first need to split the development dataset: we decide to keep 80% of the observations in the training set, and use the remaining 20% for the test set.

TABLE I
GRID SEARCH PARAMETERS FOR RANDOM FOREST

Parameter	Values
n_estimators	{100,150,200}
criterion	{gini, entropy}
min_impurity_decrease	{0.0,0.05,0.1}
min_samples_split	{2, 4}
min_samples_leaf	{1, 2}

TABLE II
GRID SEARCH PARAMETERS FOR SVM CLASSIFIER

Parameter	Values
C	{0.1,1, 10, 100}
gamma	{1,0.1,0.01,0.001}
kernel	{rbf, poly, sigmoid}

Having set these values, we run the grid search for the classifiers and we evaluate each configuration on the test set using the F1-score, both for the RF and the SVM classifier. We identified the following best configurations:

- SVM: {*C*: 10, *gamma*: 0.001, *kernel*: 'rbf'}. With this configuration, we obtain F1-score = 0.99 on the test set.
- RF: {*criterion*: 'entropy', *min_impurity_decrease*: 0.0, *min_samples_leaf*: 1, *min_samples_split*: 2, *n_estimators*: 160}. With this configuration, we obtain F1-score = 0.95 on the test set

The hyperparameter tuning of the Random Forest is the most computationally expensive step in the entire process, with a total time of 9 minutes (1 minute for SVM grid search).

III. Results

In this section are reported the main results of the analysis. The F1-scores reported in Table III are the mean of the five F1-scores obtained by each model (with the best configuration among the ones reported in Tables I and II) during the fine tuning process.

TABLE III
Results

Model	**Best parameters**	**F1-score (5 folds)**
RF	*criterion*: 'entropy' *min_impurity_decrease*: 0.0 *min_samples_leaf*: 1 *min_samples_split*: 2 *n_estimators*: 16	0.95
SVM	*C*: 10 *gamma*: 0.001 *kernel*: 'rbf'	0.98

We have empirically shown that the selected classifiers achieve satisfactory results in terms of macro F1-score. In particular, not only the Support Vector Machine outperforms the Random Forest approach, but it requires a smaller effort in the hyperparameters tuning step as well.

IV. Discussion

Although the results obtained are already very promising, there is still space for improvement. The following are some aspects that might be worth examine to further improve the obtained results:

- Consider other feature extraction approaches, for example using *LibRosa* python package to extract Mel Frequency Cepstral Coefficients (MFCC), often used for speech recognition tasks).
- Run a more thorough grid search process, since only a limited set of hyperparameters has currently been studied.

References

[1] Chen, Lei-Ting Wang, Ming-Jen Wang, Chia-Jiu Tai, Heng-Ming. (2006). *Audio Signal Classification Using Support Vector Machines*. 188-193.

[2] Delgado Contreras, Juan Ruben García-Vázquez, Juan Brena, Ramon. (2014). *Classification of Environmental Audio Signals Using Statistical Time and Frequency Features.*

[3] Inam Ur Rehman. (2020). *Classification on FSDD using Spectograms*. url: https://www.kaggle.com/code/iinaam/classification-on-fsdd-using-spectograms/notebook