ECE4200 - Cornell University
Kaggle Competition

Jorge Calvar Seco (jc2767)

December 12, 2021

1 Introduction

In this report I expose the work I have carried out to create successful models for the ECE4200 Fall 2021 Kaggle Competition at Cornell University.

The dataset consisted of a series of audio files (.wav), in which 2 numbers from 1 to 4 were recorded at the same time. The aim is to predict these numbers.

2 Data preprocessing

To process the data, I have developed a series of functions and a script. Using scipy the audio files are read, and then moved on to a pandas dataframe. Finally, this dataframe is converted to a numpy array to be used in the models. A 90/10% split has been used to separate train and validation sets. The number of samples used varied in the 10000 - 50000 range.

The audio data has also been converted to a spectrogram using the `scipy.signal.spectrogram` function. This creates a new dataset which has two dimensions per sample instead of one.

3 Model creation

3.1 First notebook: course models

Firstly, I have tried many of the different models studied in class with the `sklearn` package:

- **Decision Trees:** used maximum depths from 1 to 12 with both *gini* and *entropy* criterions.
- **Logistic Regression:** iterated over C parameter, as done in the course assignment.
- **Naive Bayes:** implemented gaussian naive bayes.
- **SVM:** iterated over degrees from 1 to 6 with the polynomial kernel, and tried also the rbf kernel. Tried several values for C.
- **Bagging:** tried several values for the number of estimators. Used a decision tree with entropy criterion as base estimator.
- **AdaBoost:** used both decision trees and random forests as base estimators. Iterated over the number of estimators and the maximum depth for the decision tree.

From the models above, the SVM worked clearly best, obtaining training and validation scores close to 100%. However, it took considerable time (several hours) to train. The performance of the other models are commented on the corresponding notebook.

3.2 Second notebook: PyTorch (final submission)

A second notebook was developed with the purpose of creating neural networks using `PyTorch`. This notebook was used for the final submission. The benefit of this new approach is the ability to create great performing models (similar to SVMs) while having a much smaller training time, which allows to try more possibilities. Many variations were tried including:

- **Batch Normalization:** it is vital to normalize the data before applying the model. Normalization between hidden layers also improved the performance slightly.
- **Dropout:** this randomly drops some samples between two layers. It avoids overfitting, but performance decreased slightly, so it was not used.
- **Convolution and pooling:** this technique could be applied to the spectrogram, treating it as an image. However, it did not improve results.
- **Number of hidden layers:** after trying several possibilities, 2 hidden layers were used.
- **Number of hidden neurons:** we tried several possibilities and found out that numbers near 500 were best.
- **Activation function:** LeakyReLU was used, as there was a slight improvement with respect to ReLU.
- **Epoch / Batch size:** several combinations were tried. The best result was obtained when batch size was the whole training set and epoch was near 500.

4 Missing label 43

During initial exploration, we noticed that the 43 label was not present in the training set. However, after listening some audios form the testing set, we discovered that it was present in the testing set. This explained the fact that accuracy obtained in the leaderboard after submitting a model was significantly lower than validation accuracy.

To solve this problem, we did the following:

- **Multioutput model:** a pytorch model was created with 4 neurons in the output layer. Each neuron explained the probability of a single number from 1 to 4 being present in the input audio. This solution increased notably the testing accuracy above 90.9%, which was the maximum obtainable before.
- **Labeling testing data:** with permission of the course instructor, we manually selected testing audios with the 43 label and incorporated them to the training set. To make the number significant, data augmentation techniques were used. However, this solution did not improve much the result.

5 Conclusion

With this approach, a testing accuracy of **96.517%** was obtained in the final leaderboard.