TÉCNICO LISBOA

INSTITUTO SUPERIOR TÉCNICO

SPEECH PROCESSING

LAB 3 - Speech Pattern Classification with Machine Learning

80966 - Francisco Azevedo

81036 - Marco Graça

14^{th} Group

LSDC1 - 14h00 Quinta-feira

Prof. Isabel Maria Martins Trancoso

May 9^{th} 2018

I. Introduction

In this work we approach the INTERSPEECH 2017 Cold Challenge in which the objective is to classify the speaker of an utterance as healthy or with a cold (or flue). In an effort to become more familiarized with the challenges presented by computational para-linguistics, we resorted to an extensive data set >3gb, divided in a training set, a development set and a testing set, to, with the help of features extracted with openSMILE software, recognize a pattern, namely cold (or flue) cues in speech. We used applications such as Keras and Tensorflow to train neural networks for that intent and scikit-learn for the other classifiers considered.

As a beforehand study of the data set, we tested it using a human baseline, with one of us choosing a recording to test the other, unaware of the health condition of the analyzed subject. While at first the data posed was very ambiguous, being in another language, unknown to either of us, and corresponding to text read by the subject in a fast way with no special intonation, we managed to get the results in Table 1. We'll try to improve this score with the help of our speech pattern classification and machine learning knowledge through this work.

Tab. 1: Overall human baseline accuracy (10 samples)

Volunteer/Metric	Accuracy
Marco	0.5
Francisco	0.5
Manel	0.6

II. Feature Selection

From the set of .config files presented by the openSMILE software at our disposal, we tested with MFCC, Gemaps and eGemaps with several models. We expected Gemaps results to be better than MFCC because they were conceived to be applied to voice research and affective computing. eGemaps turned out to be the most promising parameter set for the extraction of features (as we shall see in the next sections) expanding on the vast set of features extracted precisely for paralinguistic tasks, Gemaps, with 26 more features, focused on giving information about the voiced segments and more Low-Level Descriptors.[3]

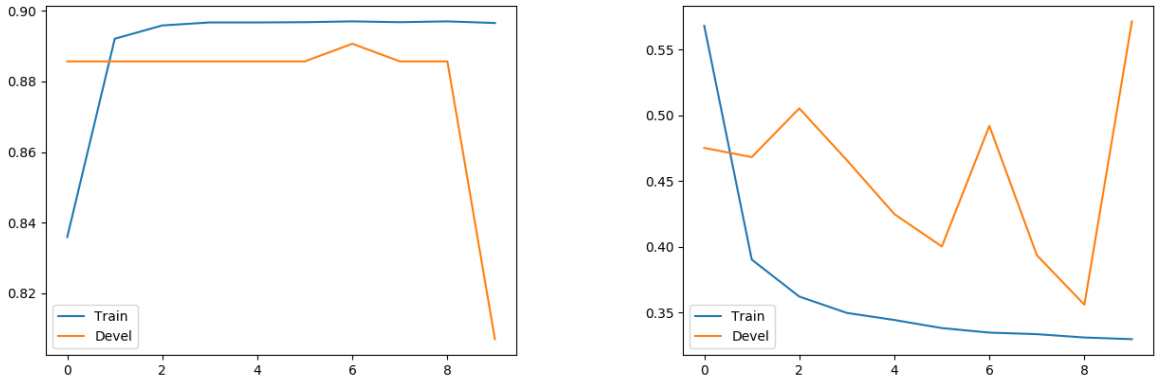
III. Applying Neural Networks

Our implementation improves on the suggested three-layers neural network. We maintain most aspects of the arquitecture: an activation function the RELU (rectifier) function, one of the most widely used for this purpose. All layers benefit from the Dropout regularization technique, that randomly drops units (along with their connections) from the neural network during training in an effort to prevent their co-adaptation and consequently an over-fitting [1].

The output of each layer unit but the output one is also subjected to Batch Normalization (a transformation is applied to it that brings its mean close to 0 and its standard deviation to 1), which reduces the amount by what the hidden unit values shift around (covariance shift) [2].

We implemented L2 ridge regularization with coefficient of 0.01 on the output of the layers (activation), which allow us to apply penalties on the complexity of the function that defines the output of that layer. These penalties are incorporated in the loss function that the network optimizes.

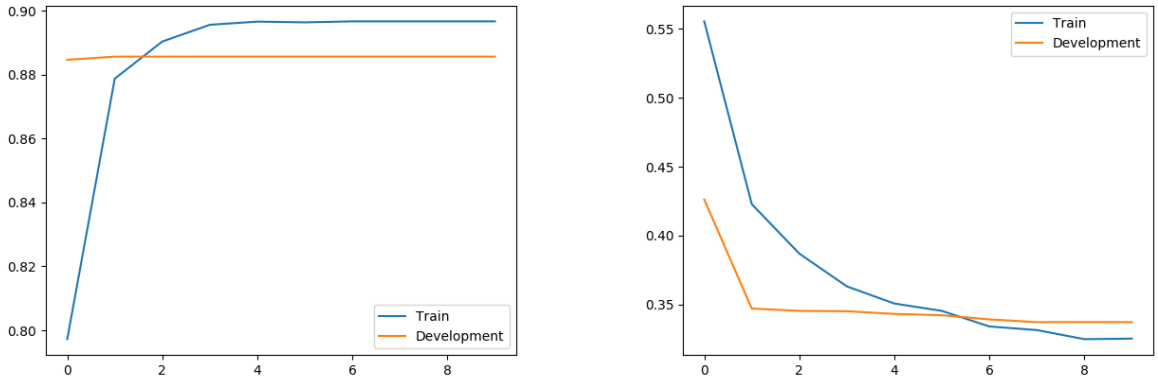
All 5 layers are fully connected, the first having 20 units (”neurons”), the hidden 10 and the output layer having 1, as this is a binary classifier. Since we wish for an output between 0 and 1 for that layer, the activation function for this unit will be the sigmoid function. The class predicted from the analyzed recording corresponds to the rounded value given by the neural network, 0 for a healthy speaker and 1 for one having a cold.



Accuracy Evolution Binary Cross Entropy Evolution

Fig. 1: Results for a Neural Network trained with the MFCC parameter set

We achieved, on test set, an UAR = 61.30 with this neural network and MFCC features which prooved better accuracy than Gemaps. We need to stress the fact that most changes, besides regularization, to the network had no effect whatsoever, this includes changing dropout values, activation layers, number of nodes in each layer or even number of layers. In most cases the results on both accuracy and UAR



Accuracy Evolution

Binary Cross Entropy Evolution

Fig. 2: Results for a Neural Network trained with the eGemaps parameter set

were equal or worse. Changing the learning rate and batch size could also change the accuracy due to having a longer, slower optimization process but the reality is that they did not. We did increase the networks size to 5 layers because we expected that this would be more adequate to the problem's complexity and available data (but the improvement was barely noticeable).

IV. Applying Support Vector Machines

Support Vector Machines (SVM's) was the second model to test. It is considered to be a good model for binary classification and especially for low amounts of data, however, this data set was medium size so other models surpassed it. We tried several kernels for this approach and using both Gemaps and eGemaps parameter sets, and settled for MFCC's with the Radial Basis Function Kernel (RBF), as it provided the most satisfying results, coupled with a tolerance of 0.001 as the stopping criterion and no iteration limit. We achieved better UAR (UAR = 0.56) than the neural network baseline but could not exceed by any means the UAR = 0.6 region even when trying different kernels, parameters and eGemaps. We were not able to understand why the accuracy was so low even when applying parameters that should yield better results.

V. Applying Decision Trees, Random Forests and Gradient Boosting

In this section we go through our first solutions with new models all based on decision trees with a twist. We decided to test decision trees despite knowing that Gemaps and MFCC had a lot of features

and therefore each feature could only have a very small relevance (something we did not have time to test). We decided to test because decision trees are a rule based system: if this happens, then make this decision; recursively. This lack of flexibility could be positive but can also lead to overfit because with this much features the tree splits a lot and coverages all possible data set cases. This leads to the tree ”memorizing”the dataset instead of generalizing. To battle this we used Random Forests and Gradient Boosting. [4] [5]

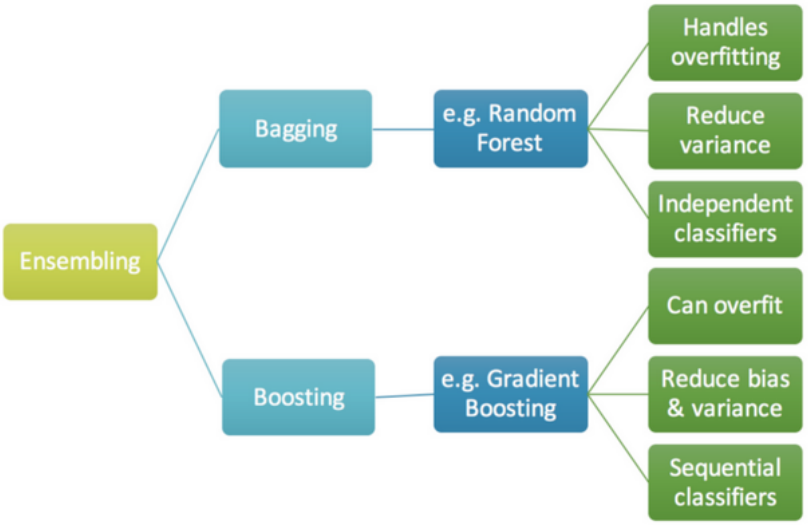


Fig. 3: Random Forests and Gradient Boosting

This figure expresses the methods advantages really well. Bagging is a method where various independent models are built and then a voting method is applied. This presumes that several ”weak”performance models together can improve the overall accuracy. Boosting is a method where the first decision tree built is a simple model that struggles with some difficult samples of data. A new model is then built focusing on achieving better accuracy on those specific data samples. This is done recursively and then each model is joined through a voting method.

Tab. 2: UAR of eGemaps and MFCC feautres in several models

Model/Features	eGemaps	MFCC
Gradient Boosting	0.521	0.598
Random Forest	0.615	0.54
Decision Trees	0.718	0.59

These models achieved the highest performance for a singular model (see Table 3) yielding good results. The eGemaps yielded much better results that the MFCC ones (Table 2) therefore we conclude that the eGemaps are the best tradeoff between feature quantity and feature relevance in comparison to MFCC’s. It is however debatable (without training data plots of loss) if the decision trees overfit or

not.

VI. Applying Gaussian Naive-Bayes Classifier

The Naive Bayes Classifier is a simple model and we reasoned that we should test at least one probabilistic model. It is based on the Bayes rule but assumes that the features are independent (which is a strong assumption). The basic idea is that each feature is modeled by a Gaussian distribution according to the data and the data determines the parameters (mean and standard deviation) of these features. This is done, generally to maximum likelihood where the posteriori probability is maximized.

This model yielded good results but as expected its not the best since there are obvious correlations between the eGemaps features (e.g.: spectral, energy, frequency features) which lowers the system performance to UAR = 0.629.

VII. Voting models

The final touch comes in the form of voting models. Voting models allow to take advantage of strengths of every model at the same time. We picked our top performing models (excluding NN that were implemented in Keras and not in SciKit) and tried hard and soft voting. In this case we used two systems with voting models:

- (1) Decicion Trees, Random Forests, Naive Bayes Classifier and Gradient Tree Boosting with soft voting
- (2) Random Forests, Naive Bayes Classifier and Gradient Tree Boosting with soft voting

The difference is simple: hard voting takes the decision of the models as 0/1 and decides the label based on majority voting; soft voting takes the values 0-1 and averages them and then rounds to 0/1. Soft voting yielded the best results in both systems. Also, Voting system (2) was substantially better than voting system (1). We infer that this is due to removing decision trees which had by far the best performance of an individual system but may suffer from overfitting as we have discussed. This would affect the average voting system of the other systems that together, without decision trees, yielded a better result.

The overall improvement of the voting methods facing the other models is noticeable in Table 3.

VIII. Conclusions

All in all, after testing several models, parameters and ideas we come to the results in Table 3. We accept that we were not capable of pursuing the full potential of SVMs since they are a powerfull binary classifier with the right features and parameters. Neural Networks where difficult to optimize and were also not very promising in general. The best results came from the decision trees and related methods which adapted well to the data. In the end the voting models really improved the system's performance

Tab. 3: Unweighted and Weighted Average Recall results on Dev. set

Model/Metric	UAR	WAR
SVM	0,565	0,898
Gradient Boosting	0,598	0,901
Neural Networks	0,613	0,807
Random Forests	0,615	0,911
Naive Bayes	0,629	0,435
Decision Trees	0,718	0,886
Voting System 1	0,721	0,897
Voting System 2	0,772	0,927

We really enjoyed doing this challenge, it was a lot of fun. We were able to obtain an UAR on test set on the same level of the paper made by the INTERSPEECH organization (UAR = 0.72). In the end we conclude that our models were much better than our human ears (UAR = 0.5) but maybe we should just leave this job for the doctors for now.

Referências

[1] Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Nitish Srivastava **http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf**

[2] Batch normalization in Neural Networks, Firdaouss Doukkali **https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c**

[3] Apresentações da Unidade Curricular, Isabel Trancoso, 2018 IST **http://www.l2f.inesc-id.pt/~imt/priv/pf/**

[4] Gradient Boosting from Scratch, Prince Grover **https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d**

[5] A Gentle Introduction to Gradient Boosting, Jason Brownlee **https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-**