MACHINE LEARNING
Group Assignment

Professor:	G.A. Chrupala
Date:	14th December 2018
Group number:	01
Codelab account	skaran
Levi Pols	[u320513]
Sema Karan	[u924823]
Atilla Asar	[u462846]

Group Tasks

Member	Data processing	Feature Engineering	Algorithms	Tuning	Report
Levi Pols	Functions for processing	Feature summaries Ablation Analysis	Perceptron,Logistic Regression,SVC	Hyperparameter tuning	1,2,3 parts of report
Sema Karan	Output function	PCA, label encoding	KNN, Multilayer Neural Net	Hyperparameter tuning	3,4,5,6 parts of the report
Atilla Asar	Functions editing	-	Sequential model with “selu” activation and batch normalization	Hyperparameter tuning	Code Cleaning

1. Introduction

The task of this challenge was to classify which of several English words is pronounced in an audio recording, which is a multiclass classification task. MFCC coefficients were given from the audio files with train, test data.

2. Data Exploration, Preprocessing & Feature Engineering

There were 105835 objects (.wav files) with the MFCC's coefficients as features. Each object is represented in a matrix, with 13 columns that are representing the coefficients, and a number rows as timeframes.
There was 94824 audio file for training set and 11005 for the test set. There were 35 words (labels) for classification.

2.1. Data transformation

All of the features were in one array, so we wrote *"find_indices" &"create_feature_array"* function that groups the train and test features together and filtered out the features with random noise. Most of the features had 99 timeframes, but a fraction of the features had a smaller number. As machine learning algorithms like features to be in the same shape, we transformed our smaller features by padding zeros so that the total number of rows was equal to 99 (*"make_3d"* function). For the model validation we split 30% of data aside.

2.2. Normalization, Scaling & Label Encoding

Before model running, we standardized our features by removing the mean and scaling to unit variance, by using the StandardScaler from Scikit Learn This benefited the accuracy of our models substantially.

2.3. Feature Engineering - Principal Component Analysis (PCA)

To decrease the dimensionality, we run PCA and observed that first 200 component explains the 90% of the variability as in ***Plot.2*** (see appendix) in the data and we used 200 component of the features for KNN.

2.4. Feature Engineering - Ablation Analysis

Since structure of our features are spatial (coefficients) and temporal (timeframes), we transformed the features to certain summaries (e.g. the mean) per coefficient and the summary functions such **mean, min, max, std and skew** are used. We found that this lead to a higher accuracy than using all concatenated data.
We performed an ablation analysis by removing one feature at a time and measuring the relative drop in accuracy. By doing this we quantify the contribution of the summary features, given all other features.The results of our ablation analysis are show in ***Plot.1*** (see appendix).

3. Learning algorithm(s) & Model Selection

We tried a number of different algorithms on processed data with the results presented in the ***Table.1*** (see appendix) and explained below in terms of features we used.

3.1. Perceptron

The accuracy scores for perceptron trials were 21% and 29% with running all concatenated data and summarised features data respectively.

3.2. Logistic Regression (SGD)

Logistic regression(SGD) gave slightly better result with 48% accuracy.

3.3. K-Nearest Neighbor (KNN)

KNN is worked better than perceptron & logistic regression while using 200 components (PCA) and gave 57% accuracy.

3.4. Support Vector Machine (SVM)

For computational time reasons, we only used the mean, min and max summary function for the support vector classification accuracy., this gave 57% accuracy.

3.5. Neural Networks (Multilayer Perceptron)

MLP neural network gave the highest accuracy before tuning hyperparameters with validation data (71%), so we chose MLP for our multiclass classification task to tune hyperparameters before running test data.

4. Parameter Tuning & Validation

MLP is tuned with the model hyperparameters such as size and number of hidden layers, learning rate, epoch, batch size and activation function. We found out that activation function "selu" with 6 layers of network gave the best accuracy result in a 40 epochs on validation data (80%) and final test data (77%). We have also plotted accuracy and loss versus amount of epoch.

5. Discussion of the performance

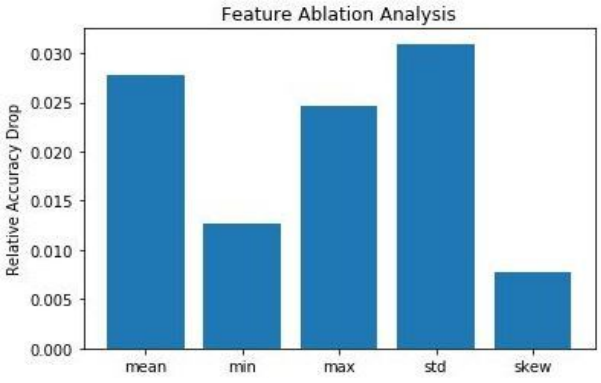
Since MLP gave the have tried MLP neural network with varying parameters, namely; network depth, activation function, batch sizes, initializations, and epochs. We've run MLP ~30 times with different hyperparameters but regarding script

running time & report limitations, we've added only the first and final model setup in the report. We found out that activation function "selu" with 6 layers of network gave the best accuracy result in a 40 epochs. We have also plotted accuracy and loss versus amount of epoch in Plot.3 (see appendix). We concluded that MLP gives better accuracy with high dimensional data for multiclass classification task, though it requires more computational cost to increase the accuracy. As we searched during model selection Recurrent neural networks are the best to model speech data but it was beyond our ML course and we had time limitations to build such a deep learning model.

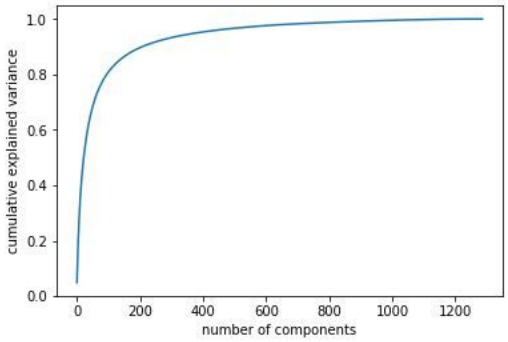
6. Appendix

Table.1

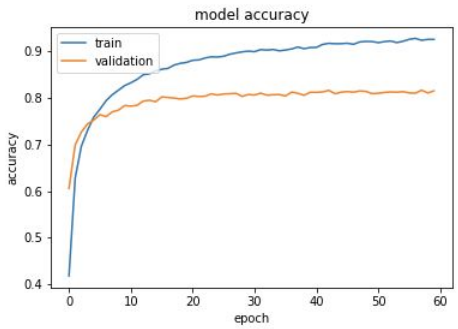
Features	**Algorithm**	**Highest Accuracy**
All features	Perceptron	21 %
All summary functions concatenated	Perceptron	29 %
All summary functions concatenated	Logistic Regression	48 %
Mean,min,max concatenated	Support Vector Classification	57 %
200 Components (PCA)	KNN	57%
All features	Neural Network (MLP) (# of hidden layers:3, size:512,epoch:15, batch:256, learning rate;0.2, activation func:relu,	77%
All features	Neural Network (MLP) (# of hidden layers:6, size:1024,epoch:40, batch:512, learning rate;0.2, activation func:relu,	81% - validation **78% - test (codelab)**



Plot.1



Plot.2



Plot.3