TINONS Project: Speaker recognition

Kasper Nielsen and Alexander Rasborg Knudsen
Department of Engineering, Aarhus Universitet
(Dated: June 2, 2014)

The principal objective of this paper is finding a good method for speaker classification. The dataset consisted of three different speakers, saying the digits 0-9, divided onto three datasets. One with one digits, one with two digits and one with ten digits expressed. The five classifiers is linear models, probabilistic generative models, support vector machines, Gaussian mixture models and artificial neural networks. The best model on the dataset was the artificial neural network, with an accuracy of 95.3 % for one digit and 89.4 % for ten digits with three male speakers.

INTRODUCTION

The purpose of this case project is to take the learnings of the course *Non-linear Signal Processing and Pattern Recognition (TINONS1)* and apply them to a specific signal processing or pattern recognition problem. In this case the object is to create a speaker recognition system. The system shall be able to recognize a specific speaker from a ensemble of speaker models.

DATA COLLECTION

This projects dataset consist of 1.656 sound files. Each file is 2 seconds long and was sampled in $16\frac{bits}{sample}$ at $48kHz$. The data structure consist of 4 persons stating a numbers between 0 to 9 a number of times. One person was removed from the dataset because of lack of data and low *SNR*. Of the remaining speakers 15 utterings were chosen at random from each and put into the training set. Another five from each was put into the test set.
The data is borrowed with permission from [1].

FEATURE EXTRACTION

When doing speaker recognition it is advantageous to classify on the basis of extracted features from speech data, rather than the audio samples themselves [2]. Features were extracted on a frame-to-frame basis. This was done on the assumption of pseudo-stationarity of human speech in the scale of a few tens of ms [2]. The frames consisted of 256 samples, with a new frame staring every 100 samples. This means that we have a frame length of

$$t_{framelength} = \frac{N_{frame}}{F_s} = \frac{256}{48\ kHz} = 5.33\ ms \qquad (1)$$

and a frame interval of

$$t_{frameinterval} = \frac{N_{interval}}{F_s} = \frac{100}{48\ kHz} = 2.08\ ms \qquad (2)$$

The features extracted for use in classification were 12th order Mel-frequency Cepstral Coefficients (MFCC), as they have proven effective in speaker recognition applications and speech processing in general. MFCCs describe the the Mel cepstrum, a spectrum-of-a-spectrum on the non-linear Mel-scale. The process for calculation MFCCs is described in figure 1 below.

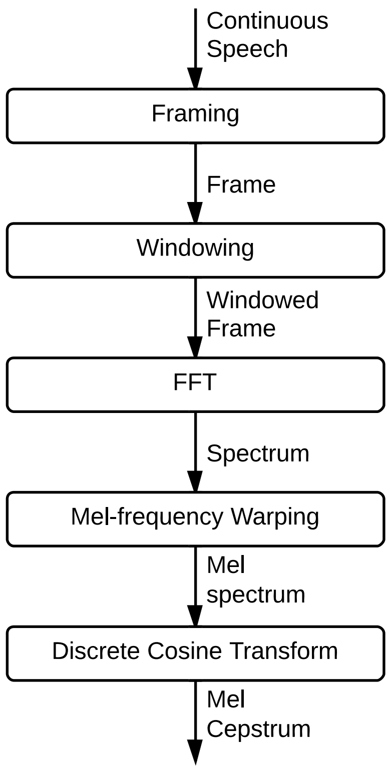

Figure 1: Process for calculating MFCC

Further, so-called delta and double-delta coefficients, the temporal derivatives $\frac{dC}{dt}$ and double-derivatives $\frac{d^2C}{dt}$ of the MFCCs respectively, are used as features. In this project the calculation of MFCCs, delta and double-delta coefficients is done using the free Voicebox toolbox for MATLAB [3]
For more on MFCC and feature extraction, see Appendix.

LINEAR CLASSIFICATION

In linear classification the decision surfaces are linear functions of the input vector $\mathbf{x}$. the target of the classification is labelled in the target variable $\mathbf{t}$, using the target values to represent class labels. In this project there are 3 different speakers, $K = 3$ then the target vector for class 3 be $\mathbf{t} = (0, 0, 1)^T$. The value of t_k can be interpreted as the probability of the given class being class C_k. To assign each vector $\mathbf{x}$ with a specific class. The linear discriminant function in its simplest form

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \tag{3}$$

Training

The linear classifier is applied to the training dataset, containing feature vectors $(\mathbf{x}_n)$ and the target vectors $(\mathbf{t}_n)$. The vectors are on the form:

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \multicolumn{2}{c}{\dots} \\ \mathbf{x}_n^T & 1 \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \dots \\ \mathbf{t}_n^T \end{bmatrix} \tag{4}$$

This calculation are done to determine the $\tilde{\mathbf{W}}$:

$$\tilde{\mathbf{W}} = \tilde{\mathbf{X}}^{\dagger}\mathbf{T} \approx (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} + \mathbf{I})^{-1}\tilde{\mathbf{X}}^T\mathbf{T} \tag{5}$$

The resulting parameter matrix $\tilde{\mathbf{W}}$ was tested using the test set.

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T\tilde{x}, \qquad \tilde{x} = [\mathbf{x} \quad 1] \tag{6}$$

The most likely class for each of the feature vectors is found as the index corresponding to the largest value in $\mathbf{y}$.

Results

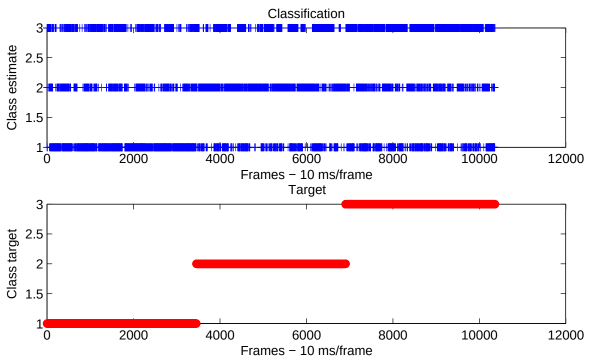

Figure 2: Results of using linear classifiers and single digit spoken

The result of the linear classifier is a total accuracy of 55.9, 51.2 and 50.6 % respectively for one, two and ten digits. The estimated for one digit is shown on figure 2, where the the blue is the estimated.

DIMENSIONALITY REDUCTION

Dimensionality reduction is used to reduce the number of variables of the feature extraction. The method used in this project is Principal component analysis (PCA), which can be used to make a dimensionality reduction of the dataset. In this project PCA is used to make the dimensional reduction focusing on the maximum variance approach.
In order to gauge the effect of dimensionality of our data, we first analyse how much of the variance is contained in how many dimensions, and then look at the cost in accuracy of truncating dimensions of the de-correlated dataset.

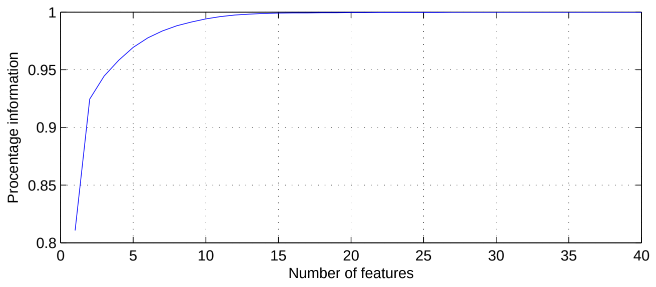

Figure 3: Cumulative distribution of variance (information) in dimensions sorted by highest eigenvalue

Figure 3 shows cumulative distribution of variance in the separate dimensions of the feature space. In this it is apparent most of the variance is contained within the first 5-10 dimensions. 94.9% at 5 dimensions and 99.4% at 10 dimensions. This does not necessarily imply that most of the information is contained within these, but is a good indicator of this.
In this we see that truncating the feature space to one quarter of the dimensions (10) after PCA, has a very little effect on the overall accuracy of the classification. This is important, since it can allow the classification algorithms to run with a much lower computational load.

PROBABILISTIC GENERATIVE MODELS

This part of the article, describes the Probabilistic Generative models (PGM). The assumed probabilistic model for each class is a multivariate normal distribution.

$$p(\mathbf{x}|C_k) = \mathcal{N}(\mathbf{x}; \mu_k,\ \Sigma_k) \tag{7}$$

Training

In the training process the mean vector and covariance matrix of each class is estimated.

$$\boldsymbol{\mu}_k = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_{kj} \tag{8}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N}\sum_{j=1}^{N}(\mathbf{x}_{kj}-\boldsymbol{\mu}_k)\cdot(\mathbf{x}_{kj}-\boldsymbol{\mu}_k) \tag{9}$$

The mean and covariance from the training is used to calculate the probability density for each class.

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)} \tag{10}$$

The class prior is selected as an equal likelihood for each class. The probability of a given feature vector $\mathbf{x}$ being part of a class C_k is given by.

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} \tag{11}$$

Result

The result of the probabilistic generative classifier is a total accuracy of 62.7, 58.1 and 57.2 % respectively for one, two and ten digits.

ARTIFICIAL NEURAL NETWORKS

The neural network model is a nonlinear function from a set of input variables $\{x_i\}$ transformed to a set of output variables $\{y_k\}$ controlled by the weight vector $\mathbf{w}$, which is made of adjustable parameters. The number of hidden units between the input and the output can be adjusted to the dataset. This process is described in the following section.
The overall network function for a two layer neural network, takes the form

$$y_k(\mathbf{x},\mathbf{w}) = \sigma\left(\sum_{j=1}^{M}w_{kj}^{(2)}h\left(\sum_{i}^{D}w_{ji}^{(1)}x_i + w_{j0}^{(1)}\right)+w_{k0}^{(2)}\right) \tag{12}$$

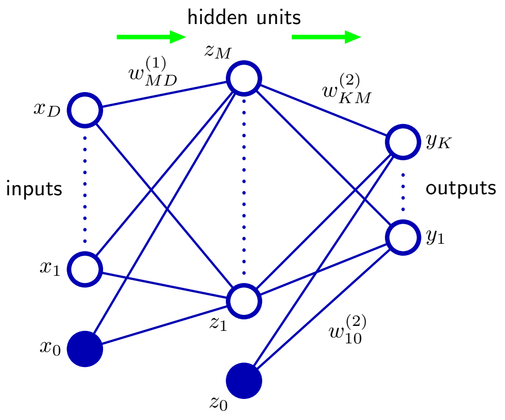

Figure 4: Diagram of the two layer neural network. The nodes in the figure represents the input, hidden and output variable. The links between the nodes are the weight parameters. The figure is borrowed from [4]

Parametetric analysis

The ANN was trained for the following number of hidden values: [10 20 50 100 200 300]. For each number of hidden values the training was done for 8 different values of α, uniformly space between 0 and 1. for each value of alpha the training was tried 8 times and the mean and variance of the error was recorded. The results are visible below in Figure 5.

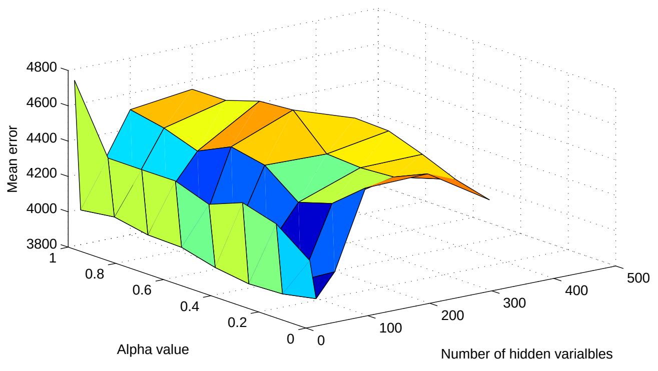

Figure 5: Results of parametric analysis. Mean error displayed.

The number of hidden units was found to be optimal at 30, with an α of 0.30. This is used for all of the datasets.

Training

In the training of the neural network for a multi-class classification problem the following error-function is minimized.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\|\mathbf{y}(\mathbf{x}_n,\mathbf{w})-\mathbf{t}_n\|^2 + \lambda|\mathbf{w}^T\cdot\mathbf{w}| \tag{13}$$

To do the actual training and parametric analysis of the neural networks the free Netlab toolbax for Matlab was used [5]

Results

The result of the ANN is a total accuracy of 95.3, 93.1 and 89.4 % respectively for one, two and ten digits. The overall accuracy of the ANN model is very good, and the model can be used as an reliable speaker recognition classifier.

GAUSSIAN MIXTURE MODELS

Gaussian Mixture Models (GMM) is a way of finding and describing sub-populations in clusters of data points. It is done by fitting a specified number of Gaussian distributions to a population of data points. Each distribution is a component of the model. The individual data points are then arranged into clusters based on which model component is most likely given the observed data point. The distributions of the mixture model are fitted to data by iteratively employing Expectation Maximization (EM).

The EM Algotrithm

In EM a mixture model is iteratively updated by estimating model parameters $\boldsymbol{\mu}_k$ *component means* (16) and $\boldsymbol{\Sigma}_k$ *component covariance matrix* (17) based on current assignment of responsibility and then reassigning responsibility based on new model parameters (15), starting out with a random guess of $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ for $k = 1..K$

Initialization:

1. Choose initial estimates for model parameters $\pi_k, \mu_k, \boldsymbol{\Sigma}_k$.
   - k is the component number out of K.
   - π_k is the weight of the kth component.
   - μ_k is the mean of the kth component.
   - $\boldsymbol{\Sigma}_k$ is the covariance matrix of the kth component.
2. Compute the initial log-likelihood og the model

$$\ln p\left(X|\mu, \boldsymbol{\Sigma}, \pi\right) = \sum_{n=1}^{N} \ln \sum_{k=1}^{N} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \boldsymbol{\Sigma}_k) \qquad (14)$$

E-step:

Calculate the probability of each point in each component in order to assign responsibility

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_n \mathcal{N}(\mathbf{x}_n|\mu_j, \boldsymbol{\Sigma}_j)} \qquad (15)$$

M-step:

Estimate new guesses for $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$.

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_k \qquad (16)$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_k - \boldsymbol{\mu}_k^{new})(\mathbf{x}_k - \boldsymbol{\mu}_k^{new})^T \qquad (17)$$

$$\pi_k^{new} = \frac{N_k}{N} \qquad (18)$$

Convergence check:

Recalculate log likelihood using (14). If the log likelihood has not changed more than some predetermined threshold stop iteration. Otherwise continue from E-step.

Training

GMM is a type of unsupervised machine learning, and is a first hand not useful in our project, but by training a GMM for each speaker based on the pre-classified training set, you end up with a very effective model. When applied in this way the components of the individual speaker models can be thought of as a PGM for each characteristic sound a speaker makes. Combined into a mixture model, these components make up a accurate and narrow model of an individual speakers voice. This allows for speaker classification on the basis of highest overall likelihood.
To further improve classification, overall likelihood is averaged, for each speaker model, over a superframe of 100 frames. This is done on the assumption that only one speaker is speaking at a time, so this speakers model is most likely to have the greatest overall likelihood for most frames out of the superframe.

Results

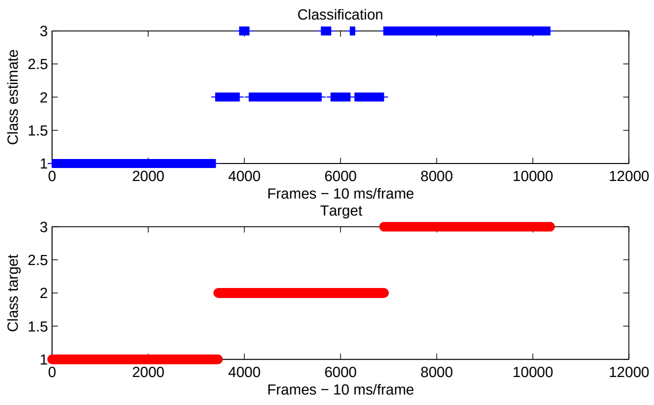

Figure 6: Results of using GMM with 3 speakers, 8 centers per model and 1 digit spoken

Gaussian Mixture Models with a trained model for each speaker yielded a very high classification accuracy. 94.6 %, 91.2 % and 89.1 % for 1, 2 and 10 digits spoken, respectively. The estimated for one digit is shown on figure 6, where the the blue is the estimated.

SUPPORT VECTOR MACHINES

In this section the soft margin support vector machine (SVM) is described. The SVM is a binary classifier, which is a problem with multi class. The solution for multiclass classification is using a one-vs.-one setup. The decision function for a standard SVM for a test point x_{new} is given by

$$t_{new} = \texttt{sign}(\mathbf{w}^T \mathbf{x}_{new} + b) \tag{19}$$

Training

The parameter vector $\mathbf{w}$ is found by maximizing the margin or minimizing the length of the parameter vector, because of the inverse relationship $\gamma = \frac{1}{\|\mathbf{w}\|}$. In the training process of a soft margin SVM, the following expression is minimized.

$$\mathbf{w}^* = \texttt{argmin}_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N} \xi_n, \tag{20}$$

$$\texttt{w.r.t.} \qquad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n \tag{21}$$

The influence of each training point in the decision boundary is proportional to α_n. After the training process, the classification can be done by using the following expression, where $\mathbf{x}_n$ are the support-vectors, and α_n is their weights.

$$t_{new} = \texttt{sign}\left(\sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}_n, \mathbf{x}_{new}) + b\right) \tag{22}$$

with $k = \texttt{exp}(-\gamma\|\mathbf{x}_n^T - \mathbf{x}_{new}\|^2)$, which is the Gaussian kernel, taken from [6].

Results

The result was obtained by using a Gaussian kernel, and the classification accuracy was found to be 73.7 % for the one digit dataset after training on the training set and afterwards classifying the test set.

DISCUSSION

This section contains a brief discussion of the results of the five different classification methods which have been used throughout this project. The resulting accuracy is shown on table I.

Model	One digit	Two digits	Ten digits
ANN	95.3 %	93.1 %	89.4 %
GMM	94.6 %	91.2 %	89.1 %
SVM	73.7 %	-	-
PGM	62.7 %	58.1 %	57.2 %
Linear	55.9 %	51.2 %	50.6 %

Table I: The table shows the overall accuracy of classification for the model used in this project.

In this project the speaker base only contained three people and is therefore fairly limited. Some of the models that were used to classify the data may not work as well, or at all, with different speakers or a larger group of speakers. To make the models more robust are large base of speakers is needed. For some of the models three speakers base is already at the limited of the computer power and time available, in the current implementation.

The lowest classification accuracy is found in the linear models, which was expected because of the simplicity of the model. The dataset has a lot of dimensions and overlapping, and is therefore too complex for a linear decision boundary. The second lowest classification accuracy is found in the probabilistic generative model. This is to be expected, as modelling all the sounds in a word or digit precisely, let alone 10 of them, with a single multivariate Gaussian is not feasible. One of the problems with PGM, as with GMM, is that numbers of independent variables that have to be determined for each class scale with the number of dimensions.

The middle accuracy model is the support vector machine. The training phase needs to be iterated a large number of times, and therefore takes a considerable time to do. The result of the smallest dataset is not very accurate compared to other methods, so it was not deemed worthwhile applying the model on the other two larger datasets.

The model with the second highest classification accuracy is the Gaussian mixture model. The model have a overall accuracy close to ANN, and are working well as a classifier for speaker recognition. The reason for the good result can be that the GMM have more than one Gaussian to describe each class. The training phase of GMM was computational load heavy, but the classification did not demand so much.

The model that have the highest classification accuracy of speaker recognition is the artificial neural network. The model was found to be very demanding regarding the computational load during the training phase. This is because the ANN doesn't have a single minimum, and therefore the training needs to be iterated a large number of times.

A way to make the models more robust is the usage of a universal background model(UBM). With this we would havve a reference threshhold for classification. This way input from speakers outside the test ensemble or simple silence/background noise would not nescesarilly be forced into a classification, but could instead be put into a null classification. Likewise the use of a null class for silence could improve separation of speakers in the training set featurespace, leading to more refined models. The use of a UBM is described in further detail in [2].

CONCLUSION

In this project five classification methods were used for speaker recognition. The dataset consisted of three different speakers, saying the digits 0-9, divided onto three datasets. One with one digits, one with two digigts and one with ten digits expressed.

The different methods was implemented through MATLAB and compared using the confusion matrix and the overall accuracy. The result is that we can determine who is speaking it is down to 95.3 % for one digit and 89.4 % for ten digits. The model that gave the best result was the Artificial Neaural Networks with Gaussian Mixture Models as a close contender.

[1] J. H. H. Christoffer Mose, Simon L. Madsen, "Non linear signal processing in speech recognition.".

[2] S. P. A. E. Rosenberg, F. Bimbot, *Overview of Speaker Recognition.*

[3] M. Brookes, *VOICEBOX: Speech Processing Toolbox for MATLAB.* Department of Electrical & Electronic Engineering, Imperial College, Exhibition Road, London SW7 2BT, UK.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 1 ed., 2007.

[5] NETLAB_toolbox, *available at: www.aston.ac.uk... /eas/research/groups/ncrg/resources/netlab/.*

[6] P. Ahrendt, "Svm_notes.pdf.".