TINONS Miniproject

Speaker Recognition

**<date>**

Kim Bjerge (KBE / 20097553)

Bjarke Møholt (20041658)

Contents

[Introduction (Kim) 3](#_Toc326085106)

[Theory (Kim) 4](#_Toc326085107)

[Mel-cepstrum (Kim) 5](#_Toc326085108)

[Principal Component Analysis (Bjarke) 6](#_Toc326085109)

[Fisher Linear Discriminant (Bjarke) 8](#_Toc326085110)

[Linear Regression Classifier 11](#_Toc326085111)

[Artificial Neural Networks (Kim) 12](#_Toc326085112)

[Bayesian Classifier / probabilistic classifier (Bjarke) 16](#_Toc326085113)

[Gaussian Mixture Model (Kim) 17](#_Toc326085114)

[Conduct of Experiments 20](#_Toc326085115)

[Audio recordings 20](#_Toc326085116)

[MATLAB program 22](#_Toc326085117)

[Results 23](#_Toc326085118)

[Principal Component Analysis 24](#_Toc326085119)

[Multi-Discriminant Analysis 27](#_Toc326085120)

[Linear Regression Classifier 29](#_Toc326085121)

[Artificial Neural Networks 32](#_Toc326085122)

[Bayesian Classifier/ probabilistic classifier 35](#_Toc326085123)

[Gaussian Mixture Model 37](#_Toc326085124)

[Generative and discriminative models on all recordings 40](#_Toc326085125)

[Discussion of Results 42](#_Toc326085126)

[Future Work 43](#_Toc326085127)

[Conclusion 43](#_Toc326085128)

# Introduction (Kim)

This project encompasses implementation of different pattern classification methods for speaker recognition based on the theory presented in the course “Nonlinear signal processing and pattern recognition” (TINONS) at Aarhus University, School of Engineering. Speaker recognition systems can be characterised as *text-dependent* or *text-independent*. The methods we are aiming to develop are text-independent, meaning the system can identify the speaker regardless of what is being said. Pattern classification methods typical contain two functionalities: A training mode and a recognition mode. The training mode will allow the user to record voice and performs feature extraction used for pattern classification. The recognition mode will use the information that the user has provided in the training mode and attempt to isolate and identify the speaker based on distinguishing features. Most of us are aware of the fact that voices of different individuals do not sound alike. This important property of speech being speaker dependent is what enables us to recognize a friend over a telephone. Speech is usable for identification because it is a product of the speaker’s individual anatomy and linguistic background. In more specific the speech signal produced by a given individual is affected by both the organic characteristics of the speaker (in terms of vocal tract geometry) and learned differences due to dialects. To consider the above concept as a basic we have tried to study different algorithms based on supervised and unsupervised learning presented in the course TINONS. The algorithms presented in this report cover both discriminative models such as Artificial Neural Networks, and generative models like Gaussian Mixture Models. The theory behind these algorithms will be presented and validated by implementation of the different methods in MATLAB. In this work we will especially have focus on text-independent Speaker Recognition systems where we have studied recordings from two different speakers. Our work is based on a limited number of recordings used for training and validation of the different pattern classification algorithms presented in this report.

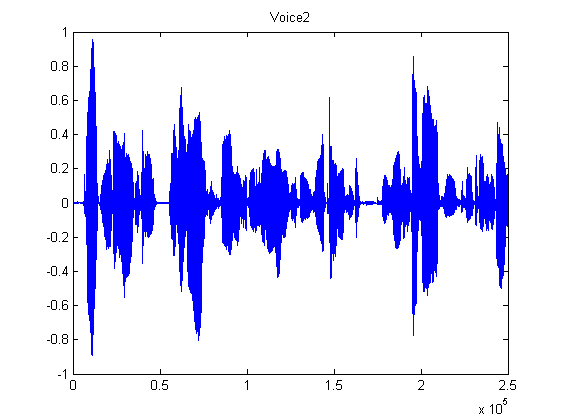


Figure 1 Recording from speaker reading a sentence

The overall goal for this work is to demonstrate the learning objectives of the course TINONS as listed below:

* *Explain* basic terminology such as supervised/unsupervised learning, likelihood, the bias-variance relation and discriminative/generative models.
* *Compare*, *relate* and *analyze* different methods for feature extraction and feature selection on real world signals.
* *Relate* and *compare* Nonlinear Signal Processing to previously learned material such as linear FIR/IIR digital filters and adaptive filter theory.
* *Design* and *evaluate* algorithms for Linear Regression and Classification on real world signals.
* *Apply* and *explain* Artificial Neural Networks on real world signals.
* *Apply* and *explain* Gaussian Mixture Models and EM-algorithm on real world signals.
* *Apply* and *explain* Sampling Methods on real world signals.
* *Apply* and *explain* Principal Component Analyses on real world signals.
* *Apply* and *explain* Hidden Markov Models on real world signals.

# Theory (Kim)

This chapter will describe the basic theory of the algorithms and methods used in our project. Most of the described theory is based on the book “Pattern Classification” [2]. The theory covers how to make feature selection and extraction on speech signals based on the Mel-Frequency Cepstrum followed by a description of methods to reduce the feature dimensions. Feature reduction can be done by Principal Component Analysis (PCA) or by Multi-Discriminant Analysis (MDA). In the PCA method focus is to find a projection of the feature space that best represent the data in a least-square sense. The MDA method focus is to find a projection that best separates more classes from each other.

In the following chapters are described different discriminative and generative models that we have used in our work for classification of speech signals. Linear Classification and Artificial Neural Networks (ANN) are both in the category of discriminative models. Hyperplane decision boundaries as defined in linear classification are surprisingly good on a range of real-world problems. For more demanding application the approach of ANN or multilayer Perceptrons (MLP) can provide a better solution to an arbitrary classification problem. Here we seek a way to learn the nonlinearity of the problem at the same time as the linear discriminant.

The project finally explores generative classification models by using a probabilistic approach. Here the Bayesian decision theory and the general multivariate Gaussian distribution are introduced. The maximum-likelihood estimation is presented which is the fundament for finding an optimal solution in the Gaussian Mixture Model (GMM). GMM is in this project used to see if it is possible to use an unsupervised learning method in finding a Gaussian mixture for two different speakers. Would it be possible to determine who is speaking without supervised training? Hidden Markov Models (HMM) are not investigated in this project and therefore the theory is not described. HMM is suited for recognizing a sequence of patterns which could be good for text-dependent classification where the purpose is to like to identify who is reading a certain sentence. (TBD or only briefly described – future work)

## Mel-cepstrum (Kim)

A range of possibilities exist for parametrically representing the features of the speech signal for the speaker recognition, such as Linear Prediction Coding (LPC) [2] and Mel-Frequency Cepstrum Coefficients (MFCC) [1]. In this work we have chosen to use the MFCC’s coefficients that represent audio, based on perception. The MFCC is derived from the Fourier Transform of the audio clip. The basic difference between the FFT and the MFCC is that in the MFCC, the frequency bands are positioned logarithmically (on the mel scale) which approximates the human auditory system's response more closely than the linearly spaced frequency bands of FFT. This allows for better processing of data in our case for speaker recognition. The main purpose of the MFCC processor is to mimic the behavior of the human ears. The MFCC process is subdivided into a number of phases or blocks as illustrated below.



Figure 2 Mel Cepstrum block diagram

In the frame blocking section, the speech waveform is more or less divided into frames in this work of 30 to 60 milliseconds. The windowing block minimizes the discontinuities of the signal by tapering the beginning and end of each frame to zero. The FFT block converts each frame from the time domain to the frequency domain. In the Mel frequency wrapping block, the signal is plotted against the Mel-spectrum to mimic human hearing. Studies have shown that human hearing does not follow the linear scale but rather the Mel-spectrum scale which is a linear spacing below 1000 Hz and logarithmic scaling above 1000 Hz. In the final step, the Mel-spectrum plot is converted back to the time domain by performing a Discrete Cosine Transform (DCT). The resultant matrices are referred to as the Mel-Frequency Cepstrum Coefficients. This spectrum provides a fairly simple but unique representation of the spectral properties of the voice signal which is the key for representing and recognizing the voice characteristics of the speaker. Speaker voice patterns may exhibit a substantial degree of variance: identical sentences, uttered by the same speaker but at different times, result in a similar, yet different sequence of MFCC matrices. The purpose of speaker modelling is to build a model that can cope with speaker variation in feature space and to create a fairly unique representation of the speaker's characteristics.

In order to produce a set of acoustic vectors, the original vector of sampled values is framed into overlapping blocks. Each block will contain N samples with adjacent frames being separated by M samples where M < N. The first overlap occurs at N-M samples. Since speech signals are quasi stationary between 5msec and 100msec, N will be chosen so that each block is within this length in time. In order to calculate N, the sampling rate needs to be determined. N will also be chosen to be a power of 2 in order to make use of the Fast Fourier Transform in a subsequent stage. M will be chosen to yield a minimum of 50% overlap to ensure that all sampled values are accounted for within at least two blocks. Each block will be windowed to minimize spectral distortion and discontinuities. A Hann window will be used. The Fast Fourier Transform will then be applied to each windowed block as the beginning of the Mel-Cepstral Transform. After this stage, the spectral coefficients of each block are generated. The Mel Frequency Transform will then be applied to each spectrum to convert the scale to a mel scale. The following approximate transform can be used.

Mel(f) = 2595\*log10 (1 + f /700)

The MATLAB toolbox voicebox has been used to create the MFCC where we have created 12 cepstral coefficients for each sample with a window of 30 ms for each speech recordings. With a sampling rate of 44.1 kHz (fs) we have:

N = 1320 for a window size of 30 ms at 44.1 kHz

M = 660 for a minimum of 50% overlap of samples within two blocks

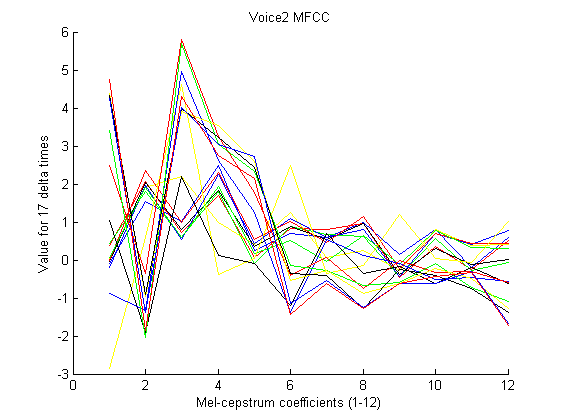


Figure 3 Mel-Frequency Cepstrum Coefficients for 17 frame blocks

## Principal Component Analysis (Bjarke)

After extracting the features, the key point is to reduce the feature dimensionality. The principle of Principal Component Analysis (PCA) is to perform an orthogonal linear transformation projecting the data onto a new coordinate system, so that the greatest variance by any projection comes to lie on the first direction (first principal component), the second greatest variance along the second direction and so on.

We consider the problem of representing all of the vectors in a set of n-dimensional samples x1…xn by a single vector x0, so that the squared distances between xk and x0 are as small as possible. The squared error function is then

We want to minimize this. This is a trivial 0-dimensional representation of the data set, and it can be shown that *x0* = **m** is the minimizer, where m is the sample mean. A 1-dimensional representation can be obtained by projecting the data onto a line that goes through the sample mean. The equation of the line is

where **e** is a unit vector in the direction of the line. If we then represent *xk* by ,we can find an optimal set of coefficients ak by minimizing the squared error, also known as the cost function:

Remembering |**e**|=1, partially differentiating with respect to *ak* and setting the derivative to zero, we obtain

We note that the best set of *ak* depends on the direction **e**. Substituting the *ak* we obtain

It appears obvious that the **e** that minimizes *J1* also maximizes **e**t**Se**, under the constraint that |**e**|=1. To maximize **e**t**Se**, we use the method of Lagrangian multipliers and get:

From this, we can see that a minimizer **e** is one of the eigenvectors of **S**. Furthermore, the eigenvector corresponding the largest eigenvalue also represents the direction of the largest variance in the feature space and, thus, the direction we want to project our features on to in order to maintain the highest level of information through a dimensionality reduction. The second-largest eigenvalue corresponds to the eigenvector representing the direction of the second-largest variance and so on.

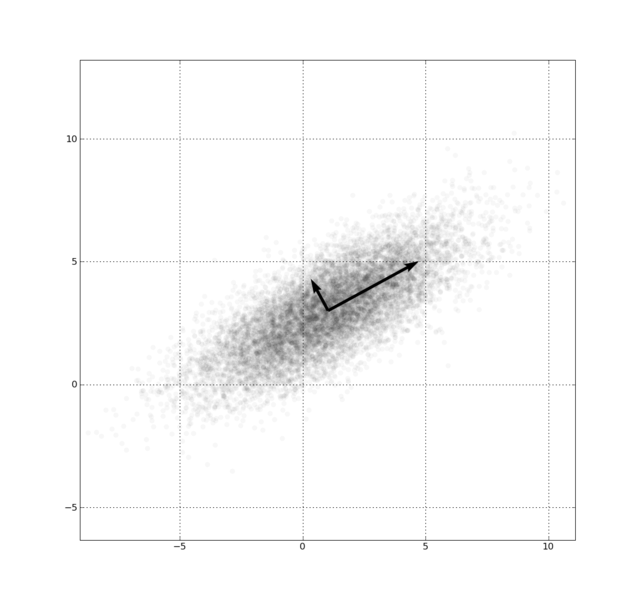


Figure 4 - First and Second Principle Component of 2D feature set

PCA enables a reduction of feature dimensionality for an unsupervised data set.

## Fisher Linear Discriminant (Bjarke)

PCA is useful for finding greatest variance and thus, representing data. However, finding the best direction for *representing* data is not necessarily the same as finding the best direction for *discriminating* data.

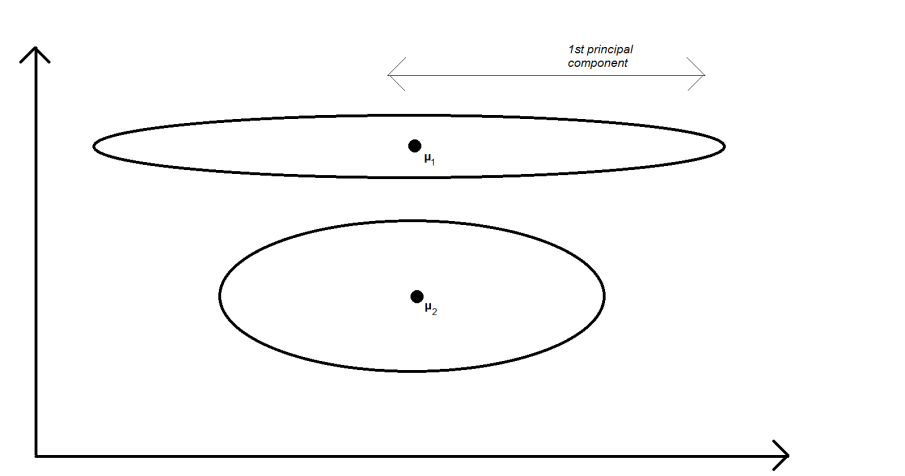


Figure 5 - PCA on two known classes

This is an example of how PCA can fail. Projecting the 2D feature space above onto the 1st principal component will make it very hard to distinguish the two classes.

For discriminating two known classes, the Fisher Linear Discriminant (FLD) method has been proposed. Assuming that we know the classes of the data we look at (supervised data), the best discrimination between the data would be a projection that 1) seeks the greatest separation between projected class means, and 2) seeks to minimize the projected variance of the classes. For discriminating more than two classes, Multiple discriminant Analysis based on the FLD is used, so I’ll start with the Fisher Linear Discriminant:

FLD begins by supposing we have a set of *n* samples **x***1*, …, **x***n*. These samples are divided into the subsets *D1* and *D2*. The subsets *Y1* and *Y2* are obtained from **x** by a linear projection *y =* **w**t**x**, where **x** ϵ *D1* and *D2* respectively. Having these definitions, we move on to finding the best direction of **w** for separating the data of the subsets, or, as it turns out, for getting the greatest difference in the sample mean values of the subsets: Let **m**i be the sample mean given by

then the sample mean for the projected points is given by

This can be seen as the sample mean projected onto **w**. From this we can derive the distance between the projected means as

We define the scatter for projected samples by

As described before, FLD employs a linear function **w**t**x** for which the criterion function

Is maximized. To obtain the **w’** maximizing *J*(**w**) we define the scatter matrices **S***i* and **S***w* by

Then

Similar to this we write

where **.**

Expressing the criterion function *J*(**w**) using **S***W* and **S***B* we can write

This expression is known from physics as the generalized Rayleigh quotient, and for such we know that any **w** that maximizes *J* must satisfy

For some constant λ. This can turn into an eigenvalue problem if we let it, but, knowing that is always in the direction of we can find **w** in an easier way: **.**

The direction **w** is the optimal direction to project the data in order to separate them.

As mentioned before, Multiple Discriminant Analysis is a generalization of the Fisher Linear Determinant for a *c­*-class problem, leading to *c* – 1 discriminant functions:

Moving from vector to matrix form, instead we write

The projected samples can be described by their own means and scatter matrices:

From here, it is easy to see that

The generalized criterion function *J*(**W**) can be written in terms of and as

Finding a matrix **W** that maximizes *J* can be done by considering the Rayleigh quotient property

A shortcut to finding the eigenvalues can be to find the roots of the characteristic polynomial

And solve, for **w***i*

Finding the directions **w***i* gives us (**her gik jeg I stå!!**)

## Linear Regression Classifier

To recognize known classes it is necessary to distinguish between them. Classifiers are mathematical tools designed for this purpose. A linear discriminant function can be written as

Where **w** is the *weight vector*. The last element in **w** makes for the *bias*.

To train a classifier in a supervised set of samples, we define the *excitation functions* *ti*(***x***) as

We also define the **T** matrix as

For each *yi* we want to find a set of coefficients ***w*** that gives us the best linear approximation, by considering the linear equation

There are numerous methods for this problem, we have used linear least-squares regression:

We define the cost function

For each class *cm*. Next we minimize this cost by differentiating for **w** and solving for 0 and get

Doing so for each class, we are now able to find the decision boundary between any two classes by finding the intersection of the linear approximations. For example, we want to separate *c1* and *c2*; we find the linear approximations *y1* and *y2* using the method described above and can now describe the decision boundary by .

Note that this method has weaknesses: Outliers have a comparably large weight due to the squared errors regression used. Also, samples should be weighted by so that any uneven distribution of samples among the classes is evened out.

## Artificial Neural Networks (Kim)

Multilayer Perceptron (MLP) or Artificial Neural Networks (ANN) implements linear discriminants like the linear regression classifiers, but in a space where the inputs are mapped nonlinearly. MLP are fairly simple algorithms where the form of the nonlinearity can be learned from training data and applies to a number of real-world applications. The most popular method for training a MLP network is the *backpropagation* algorithm that is a natural extension to the LMS algorithm. In this chapter we will only go in detail with the MLP network architecture and how to use the training algorithm.

We are using a three-layer neural network which consists of an input layer, a hidden layer and output layer. The layers are interconnected with modifiable weights. Each hidden unit computes the weighted sum of its inputs to form a scalar net activation. The hidden layer and the output layer emit an output that is a nonlinear function of its activation. For classification each feature dimension are assigned to an input and each class to an output of the MLP network. For speaker recognition the inputs would be the Mel-spectrum coefficients and the output the different speakers to classify.

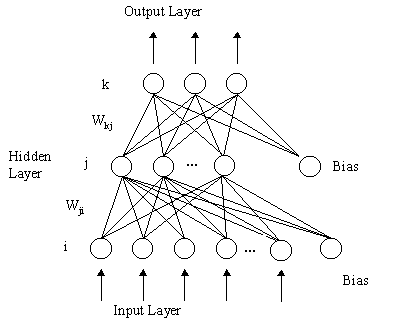


Figure 8 Multilayer Neural Network with three-layers

The output discriminant functions can be expressed as

Each output unit computes its net activation based on the hidden unit signals. Different types of activation functions can be used. The activation functions are nonlinear to ensure that points close to the discriminate line has the biggest influence on the classification. In the following we will describe the logistic, sigmoid and softmax activation functions. The sigmoid is smooth, differentiable, nonlinear and saturating. The softmax function is similar to a probability estimate with values between 0 and 1. The equations for each activation function are described below.

The Logistic activation function ensures values between 0-1 and good for two classes

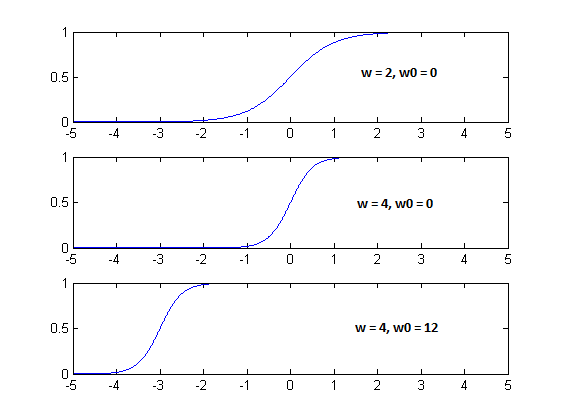


Figure 9 Logistic activation function with variation of w (slope) and w0 (offset)

The Sigmoid activation function

The generalized softmax activation function for output where and , and good for more than two classes

The steps in using and training a MLP network for classification are:

1. Choose the MLP network where the number of inputs is equal to the dimension or number of features. The number of outputs reflects the number of classes to determine. An appropriate activation function is selected.
2. A training set is selected and must be proportional to the number of chosen hidden units. The network is trained using e.g. the backpropagation algorithm.
3. The trained MLP network is validated on a test set.

We define a cost function that needs to be minimized in order to find the best MLP network.

The criterion function is defined as the sum of square errors of the training error to where a regularization term is added. The regularization term adds a value to the training error where we take into account the complexity of the network. The parameter alpha () is adjusted to impose the regularization more or less strongly. The solution for two classes we get the *cross-entropy* cost function.

Training is very time consuming and it is difficult to automate since the training for an optimal set of variables are dependent on each other. Below are listed the variables that is dependent on an optimal training:

1. Size of the training set
2. Number of hidden units (Many hidden units requires more training data)
3. Value of the alpha valued used in the regularization term
4. Initial values of the weights
5. The Bayes error

The MLP network can never be better than the Bayes error which means it will not be better than the overlap of the class distributions. Like illustrated in Figure 10 where we have two classes C1 and C2 with a Gaussian distribution and a region of overlap that defines the minimal theoretical error.

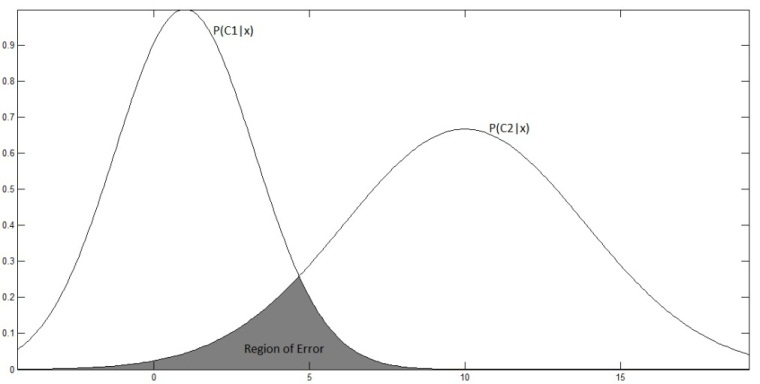


Figure 10 Bayes error for overlapping densities

The error is typical high before the training has begun. Through learning the error becomes lower, as shown in the learning curve (Figure 11). The training error reaches an asymptotic value which depends on the Bayes error, amount of training data and the number of weights in the network (hidden units). When to stop training will depend on a validation set. We will use the validation set as stopping criterion when the minimum gradient descent is reached. The optimal way would be to use cross-validation by using different blocks of random samples for the training and test set.

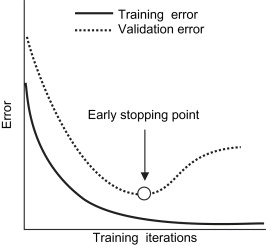
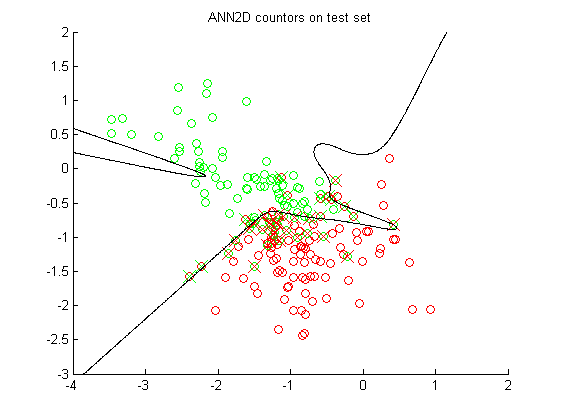
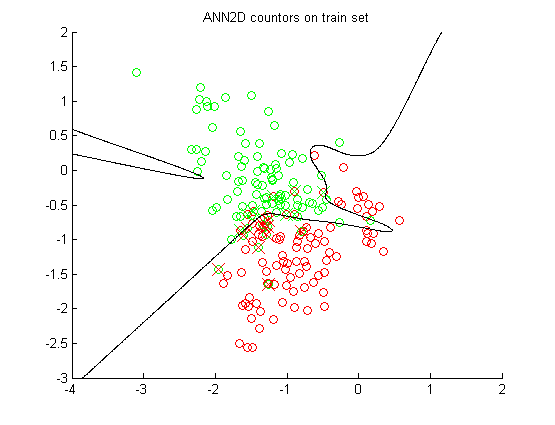
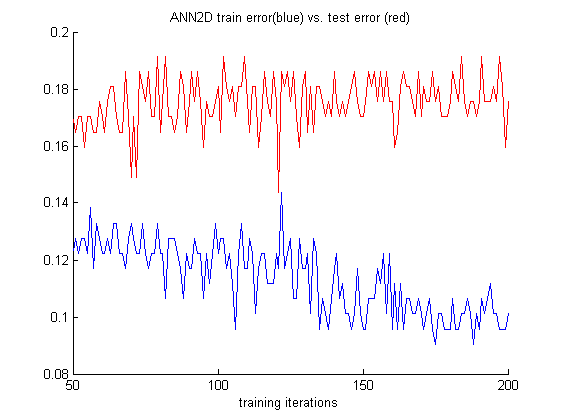


Figure 11 Learning and validation curves

The problem we get is overfitting when selecting too many hidden units with too many training iterations or improper adjustment of the regularization term. In the example illustrated below we will get an error of 0.10 on the training data but on the test data we have an error of 0.20 after 200 training iterations.





## Bayesian Classifier / probabilistic classifier (Bjarke)

Again considering a supervised set of data, we want to classify the data by soft assignment, stating a probability that a sample belongs to one or the other class, instead of the previously described hard assignment, where a hard decision boundary assigns a 100% association to a single class.

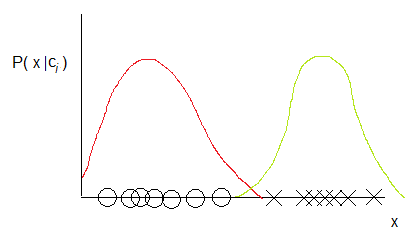


Figure 6 - probability density of two classes

Fig. 3 shows a set of 1-D samples belonging to two classes, and their distribution functions **P(x|c***k***)**. Knowing that can be found by assuming that the distribution of samples in the training set is representative, we can calculate the probabilities of a given sample belonging to a class **P(c***k***|x)** by utilizing Bayes’ Theorem:

For example, if we know the probability densities of two classes and we have an unclassified sample **x**new, we could calculate the probability of it belonging to class 1 by

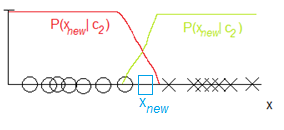


Figure 7 - probability of a sample belonging to one or the other class

The above sketch also shows a decision boundary, being where **.** Note that, the probability of **x** belonging to another class than indicated by the decision boundary is not 0, which means that it is to be expected that a portion of the samples will be incorrectly classified. We denote this as the Bayes error.

## Gaussian Mixture Model (Kim)

Unsupervised methods can be used to find patterns in data without training. Collecting and labeling a large set of sample patterns can be costly. In unsupervised learning we achieve to find methods that can be used to decide on patterns for features in classification. We will train with a large unlabeled set of data, but we still have to use supervision to label the groupings found in the data. In our project we could use such an approach in recording speech from different speakers and to use unsupervised training in looking for groups/clusters of patterns that matches the individual speaker’s identity.

A popular approximation method is the k-Means algorithm that could be used for unsupervised classification finding clusters of patterns. The method is more simple that the Gaussian Mixture Model (GMM) and could be used as a way to find means for every cluster in the training data set. The method is used to compute and accelerate the convergence of finding clustering patterns in the sample data by means of the k-Means algorithm as summarized below

1. Choose a value of *k* the number of clusters, given the number of samples *n*
2. Initiate the mean values for each cluster by choosing random samples
3. Assign sample points to each mean cluster using the minimum Euclidian distance
4. Calculate new mean values: , where is 1 if *xn* belongs to cluster *i*, 0 otherwise.
5. Iterate point 3+4 until means variance between iterations is below a treshold or that a defined cost function based on the squared Euclidian distance to the mean is small. The cost function is

If we assume that the complete probability structure for the problem can be described by a normal distribution of each cluster, we could use GMM. We have to find the unknown parameters of the probability distribution and the number of components/clusters (*k*). Assuming that the distribution for each component is Gaussian we need to find the unknown variance and mean parameters:

The probability density function for the samples is then given by

where and *k* are the unknowns. The conditional densities are called *component densities*, and the prior probabilities are called the *mixing parameters*. For the Gaussian Mixture Model (GMM) the conditional densities are the multivariate normal distribution.

where ∑ is the *covariance matrix* and *d* is the feature dimensions of the sample. We can choose 3 different types of the covariance matrix: isotropic/spherical where , diagonal where and full where

To find the optimal solution to GMM we will use the maximum-likelihood to estimate the unknown parameters which is similar to find the maximum of the log-likelihood given by

where we have set of *n* samples that we assume are *independent and identical distributed* (i.i.d.). From the probability density function we get

here the mixing coefficient are called , we have

where is a normal distribution. We can write our solution formally as the argument that maximizes the log-likelihood

In finding the maximum we need to differentiate the log-likelihood according to the parameters: to find:

The expectation-maximization or EM algorithm is used to iteratively estimate the likelihood for the above problem and finding the optimal parameters for the solution. We start with a guess for the parameters . Then we used the Bayes formula to compute the probability for the samples belonging to class in the E-step

where is the normal distribution. In the M-step we compute new estimates for the means , covariance matrices and mixing coefficients .

We continue to iterate between the EM-steps until the computed log-likelihood stops changing, reaches a certain steady value like 0.001. Alternatively we stop when the number of iterations exceeds a maximum. The algorithm is sensitive to estimates of the covariance matrix. When the eigenvalues becomes very small, the value of goes to infinite, therefore the eigenvalues () of the covariance are typical reset to 1 in this case.

# Conduct of Experiments

In this project we are aiming to experiment with different methods to distinguish between different speakers. The methods we are aiming developed are text-independent, meaning the system can identify the speaker regardless of what is being said. We have limited our experiments to focus on text-independent algorithms that would be able to identify two different male voices. We have made a number of recordings from where training and test data can be produced. The recordings are recording with the audio/MIDI-sequencer Cubase from Steinberg. All recordings are normalized and stored in wave files with a sampling rate of 44.1 kHz. The recordings contain speech from a male voice 1 and 2. The recordings contain different speech scenarios. A single word repeated a number of times, where the silence part is removed. Different sentence recorded one or more times. The length of each recording is between 3 – 6 seconds.

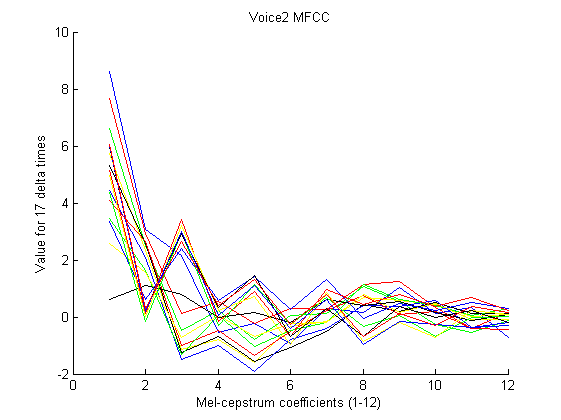
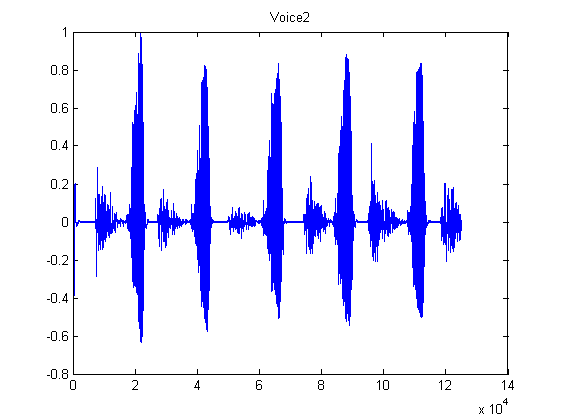
## Audio recordings

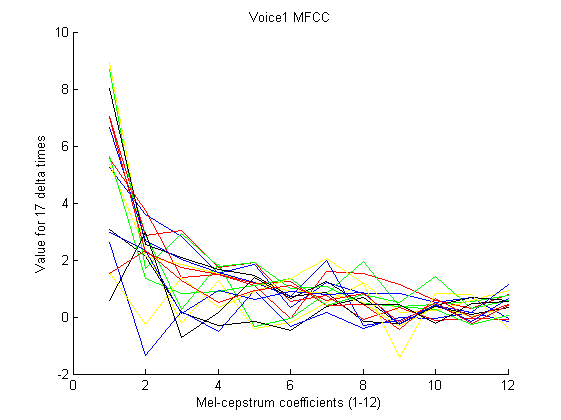
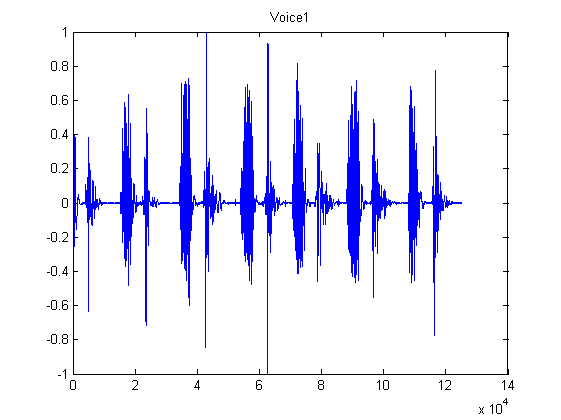
A MATLAB function is made that creates the sample features set for the recordings. With the CreateMFCCSamples function it is possible to specify the type of recordings from where the sample feature set should be generated. The result is a sample set for voice 1, 2 and silence, containing a feature set of 12 MFCC coefficients.

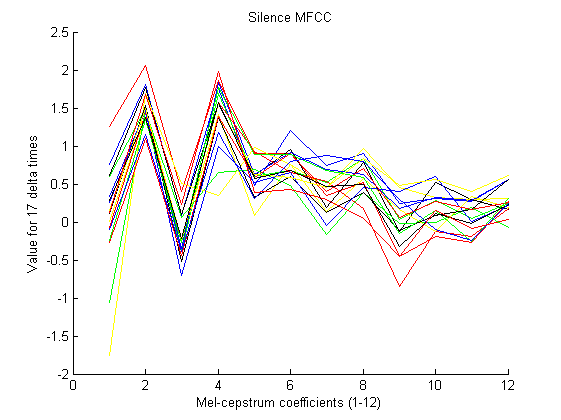
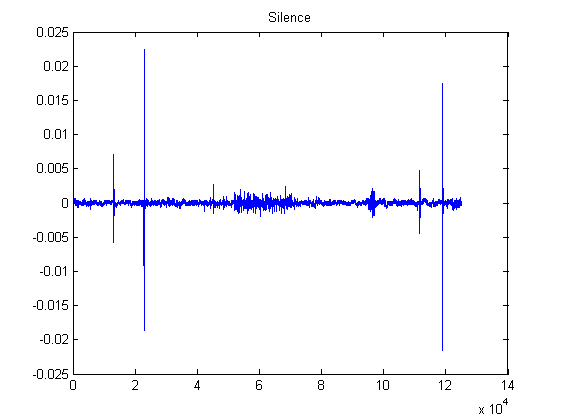
function [mfcc\_voice1, mfcc\_voice2, mfcc\_silence] = CreateMFCCSamples(PlotMFCC, Pause, Start, End)

The parameters for the function specify whether to plot the audio clip and Mel-Frequency Cepstrum Coefficients. The Start and End parameters specifies the recordings that should be used in extracting the sample feature set. Specifying Start=0 and End=5 gives a set of recordings that contains MFCC samples of all recording in total . There are 6 different recoding sets as illustrated in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Audio file names** | **Description** | **MFCC Samples** | **Start, End**  **parameter values** |
| OpBjarkeC.wav, OpKimC.wav, Silence.wav | Recordings of speaker voice pronouncing “Op”. | 188 | 0 |
| NedBjarkeC.wav, NedKimC.wav, Silence.wav | Recordings of speaker voice pronouncing “Ned”. | 188 | 1 |
| Speech1\_1.wav, Speech2\_1wav, Silence2.wav | Reading sentence 1 | 377 | 2 |
| Speech1\_2.wav, Speech2\_2wav, Silence2.wav | Reading sentence 1 (Same as above) | 377 | 3 |
| Speech1\_A.wav, Speech2\_Awav, Silence2.wav | Reading sentence A | 377 | 4 |
| Speech1\_B.wav, Speech2\_B.wav, Silence2.wav | Reading sentence B | 377 | 5 |

****

****

****

The above audio recordings visualize the speaker one and two pronouncing the word “Op” a number of times. The plot of the 12 MFCC coefficients varies over 17 delta time intervals. The window size is 1320 audio samples with a step size of 660 that means the MFCC plots covers the variation over 17\*660 = 11220 audio samples or approx. 250 ms at the sample rate of 44.1 kHz.

## MATLAB program

All experiments described in this report are combined in one MATLAB program with the purpose of exploring the different methods and algorithms for classification. The program calls functions to generate the training and test data sets, performs feature reduction and plotting the sample feature space. Finally it uses the different classification methods and algorithms as described in this repport. A number of parameters can be set to specify the execution of the program (VoiceClassificationAllRand.m) as listed below.

UsePCA\_MDAFeatureReduction = 2 % 0 = none, 1 = PCA, 2 = MDA

% Classification Methods:

% 0 = 2D, 1 = 3D,

% 2 = ANN2D, 3 = ANN3D,

% 4 = Bayesian decision theory,

% 5 = GMM2D, 6 = GMM3D, 7 = GMM2DComp

UseClassificationMethodStart = 2

UseClassificationMethodEnd = 2

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 1

% Start, End parameters

% 0,1 Op/Ned

% 2,2 Same speech

% 2,3 Same speech twice

% 2,5 All speeches

% 0,5 All recordings

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(2, 0, 0, 0);

Feature reduction can be specified to use the PCA (1) or MDA (2) methods. The classification methods can be specified as listed in the table below. The program iterates all possible classification methods specified by the UseClassificationMethodStart and UseClassificationMethodEnd parameters.

|  |  |
| --- | --- |
| **Classification method** | **Description** |
| 0 | Performs linear classification on a sample set with 2 dimensions and plotting results in 2D feature space |
| 1 | Performs linear classification on a sample set with 3 dimensions and plotting results in 3D feature space |
| 2 | Performs ANN/MLP classification on a sample set with 2 dimensions and plotting results in 2D feature space |
| 3 | Performs ANN/MLP classification on a sample set with 3 dimensions or higher |
| 4 | Uses the Baysian decision theory assuming that the training data has a normal distribution |
| 5 | Performs unsupervised classification using the Gaussian Mixture Model on the training set of 2 dimensions and plotting the result for probabilistic classification with test data |
| 6 | Performs unsupervised classification using the Gaussian Mixture Model on the training set of 3 dimensions or higher for probabilistic/generative classification with test data |
| 7 | Performs supervised classification using GMM for each of 2 classes with 2 dimensions. It finds a number of k-components of Gaussian mixtures that is then used for a generative classification. |

The UseRandomisation parameter defines whether to randomize the MFCC sample set before selecting the training and test data set. The parameter specifies to randomize the results returned from the CreateMFCCSamples function before selecting the training and test data sets. Each classification method returns the *confusion matrix* for the train and test sets being able to calculate the correct classification percentage.

switch (UseClassificationMethod)

case 0

% 2D classification training set with 2 classes and 2 features

[Ctrain, Ctest, W] = linear2D(V1new, V1tnew, V2new, V2tnew); % training

case 1

% 3D classification training set with 2 classes and 3 or more features

[Ctrain, Ctest, W] = linear3D(V1new, V1tnew, V2new, V2tnew); % training

case 2

% 2D classification using Artificial Neural Networks

[Ctrain, Ctest] = ANN2D(V1new, V1tnew, V2new, V2tnew, Snew, Stnew, 2); % 2 or 3 features

case 3

% 3D classification using Artificial Neural Networks

[Ctrain, Ctest] = ANN3D(V1new, V1tnew, V2new, V2tnew, Snew, Stnew, size(subSet,2));

case 4

% Classification based on Bayesian decision theory

% assuming a normal distribution of class features

[t\_est, Ctest] = gausianDiscriminant(V1new, V1tnew, V2new, V2tnew); % 2 features only

case 5

% 2D classification using the Expectation-Maximation (EM)

% algorithm for Gaussian Mixture Models in 2 dimensions

% A training is performed for each class V1, V2 and silence

% finding a Gaussian mixture for each model

[Ctrain, Ctest] = GMM2D(V1new, V1tnew, V2new, V2tnew, Snew, Stnew);

case 6

% 3D classification using the Expectation-Maximation (EM)

% algorithm for Gaussian Mixture Models in 3 dimensions or more

% A training is performed for each class V1, V2 and silence

% finding a Gaussian mixture for each model

[Ctrain, Ctest] = GMM3D(V1new, V1tnew, V2new, V2tnew, Snew, Stnew, size(subSet,2));

case 7

% 2D classification using the Expectation-Maximation (EM)

% algorithm for Gaussian Mixture Models in 2 dimensions

% A training is performed for each class V1, V2

% finding Gaussian mixture components (GMM) for each class

[Ctrain, Ctest] = GMM2DComponents(V1new, V1tnew, V2new, V2tnew, 5);

otherwise

% Invalid classification parameter specifier

end

# Results

In this section we will present the results of our work with the project.

The project work has developed in parallel with the material that has been taught in the course and so, many of the results have been produced in order to be easy to visualize rather than being useful for classification. For a more detailed discussion on this topic and others related to the results, please refer to the Discussion section. As a testing metric for all our classification methods, we have used the Confusion matrix found in the prtools toolbox (xxx ref?). The Confusion matrix looks like this:

The results are mainly based on two voices pronouncing the words “Op” and “Ned”. “Ned” consists of the phonemes n, e and ð. “Op” consists of the phonemes ʌ and b.

Since we assume each sample to only contain one phoneme, the number of samples with the same phoneme is related to the time in which each phoneme is pronounced. For example, if a person takes a total of 5 samples to pronounce the word “Ned” and 3 of these are ‘n’, 1 is ‘e’ and 1 is ‘ð’, we would expect the number of samples containing the phoneme ‘n’ to be 3 times as high as the samples of the other phonemes.

In the following, each subsection will start with a MATLAB code snippet for setting up the control parameters to perform the operations necessary to produce our results. After this, the results are shown as 2D images. For the classification methods, a confusion matrix is shown as a summary result of the classification.

## Principal Component Analysis

UsePCA\_MDAFeatureReduction = 1 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 0

UseClassificationMethodEnd = 0

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 0

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 0, 0); % “Op”

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 1, 1); % “Ned”

First we perform PCA on a portion of all “Ned” samples. The following is a non-randomized set of samples:



Figure 12 PCA feature reduction in 2D for "Ned" recording

Next we performed the same operation on a non-randomized set of “Op” samples. We can almost distinguish a pattern indicating the different phonemes in the projection, as well as the speakers:

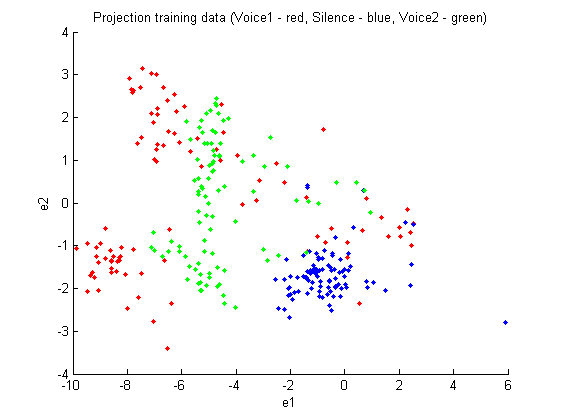


Figure 13 PCA feature reduction in 2D for "Op" recording

Our method assumes that our samples are independent. To check this, we also performed PCA on a random set of “Op” samples:



Figure 14 PCA feature reduction of randomized "Op" training data

This is basically the same projection as before, only the first principal component seems to be mirrored compared to the non-randomized.

Being –indeed– random we ran the PCA algorithm on a random set of “Op” samples once more:

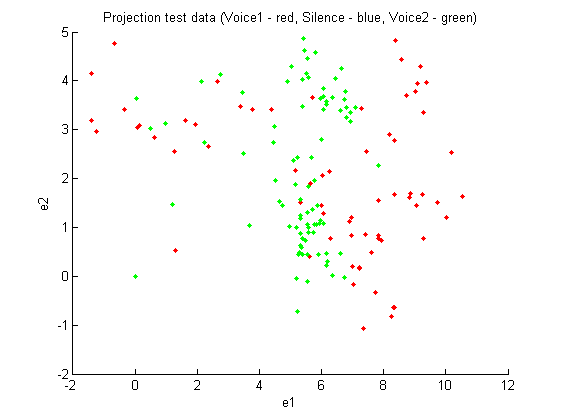


Figure 15 PCA feature reduction of randomized ”Op” test data

Except for the absence of silence samples, the projection looks very much as the previous.

## Multi-Discriminant Analysis

UsePCA\_MDAFeatureReduction = 2 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 0

UseClassificationMethodEnd = 0

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 0

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 0, 0); % “Ned”

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 1, 1); % “Op”

Since we were doing supervised training, we were able to use MDA which is better suited for classification than PCA. Performing MDA on 3 classes inherently limits our feature space to dimensions.

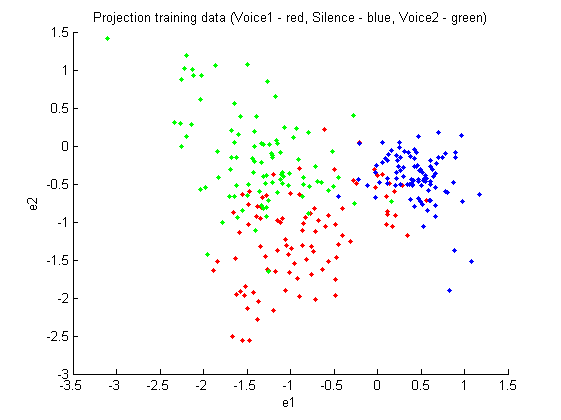


Figure 16 MDA feature reduction in 2D for "Op" recording

We see this method is clearly better suited for classifying the samples than PCA was. This plot is beautiful:



Figure 17 MDA feature reduction in 2D for "Ned" recording

Again, checking we’d get the same results using a random set of samples, it appears that the projection has somehow ‘flipped’, Voice1 and Voice2 being as clearly separable but in swapped positions:

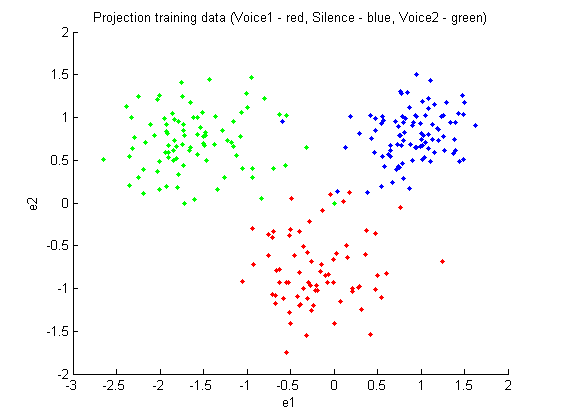


Figure 18 MDA feature reduction in 2D for randomized "Ned" recording

## Linear Regression Classifier

UsePCA\_MDAFeatureReduction = 2 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 0

UseClassificationMethodEnd = 0

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 0

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 0, 0); % “Ned”

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 1, 1); % “Op”

LRC was the first classification method we were taught and so, the first classification method we employed on our data. After our work with feature space dimensionality reduction and some initial tests with classifiers, we’ve decided to use MDA due to the better classification results it produces.

The first plot shows the results of the classifier training. Samples classified incorrectly are marked with an x:

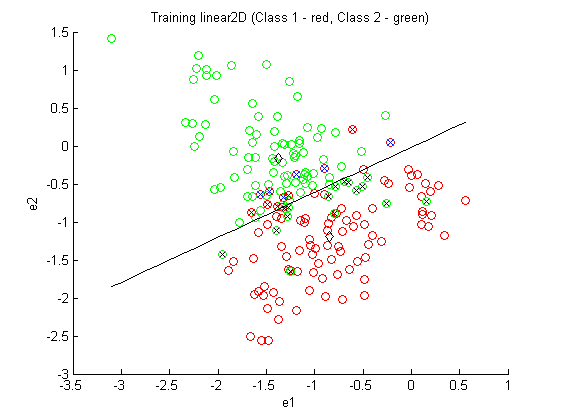


Figure 19 Training for linear classification on MDA feature reduction in 2D for "Op" recording

The second plot shows the results of applying the found decision boundary on a test set. No samples from the training are used in the test.



Figure 20 Test for linear classification on MDA feature reduction in 2D for "Op" recording

As expected, the number of classification errors is a little higher in the test than in the training.

We performed the same as above on the “Ned” samples:

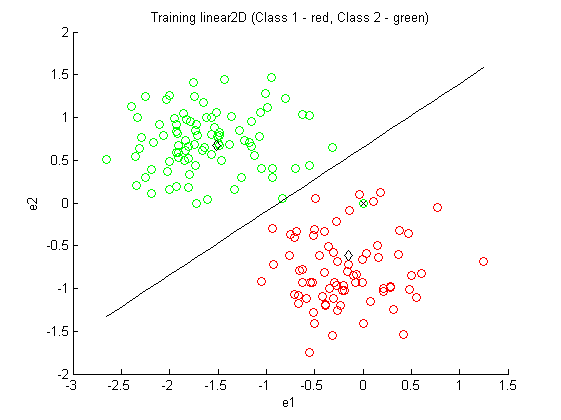
****

Figure 21 Training for linear classification on MDA feature reduction in 2D for randomized "Ned" recording

****

Figure 22 Test for linear classification on MDA feature reduction in 2D for randomized "Ned" recording

## Artificial Neural Networks

UsePCA\_MDAFeatureReduction = 2 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 2

UseClassificationMethodEnd = 2

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 0

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 0, 0); % “Op”

ANN2D.m – parameters

outputfunc = 'logistic';

nout = 2; % Number of outputs.

% Parameters to vary

nhidden = 8; % Number of hidden units.

alpha = 0.001; % Coefficient of weight-decay prior (regularization).

This chapter presents the results performing using Multilayer neural networks for discriminative classification. We have chosen to use MDA feature reduction in achieving a feature dimension of 2 using recordings from voice 1, 2 and silence as illustrated below. We have selected 8 hidden units for the ANN 2D network. A logistic output activation function is selected since we are using 2 output classes (Voice 1 and 2) for training. The coefficient alpha is chosen to 0.001 ensuring that a moderate impose of the regularization term.

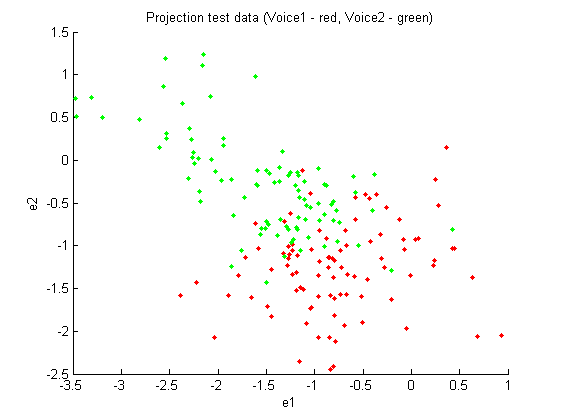
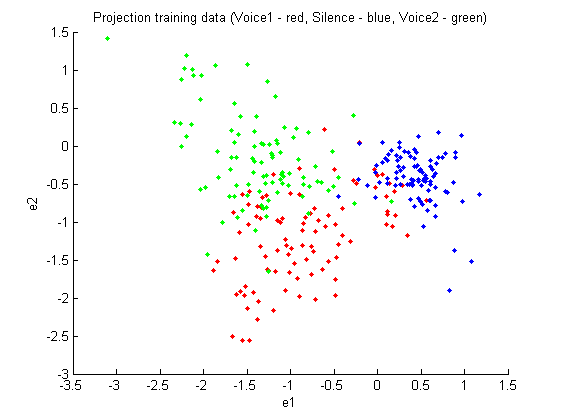


Figure 27 Training and test set in 2D for "Op" recording

In Figure 28 is shown the discriminating contour that separates the features of voice 1 from voice 2. We see that the training achieves a better result than on the testing data set

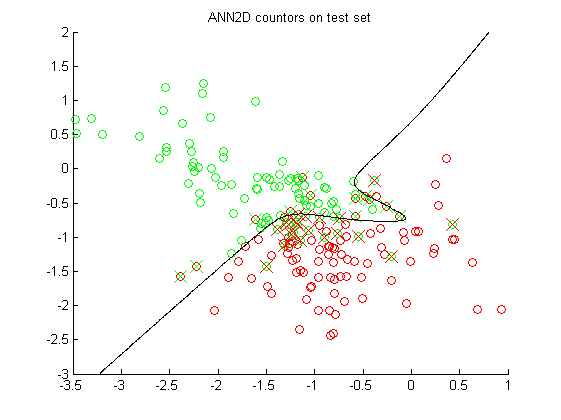
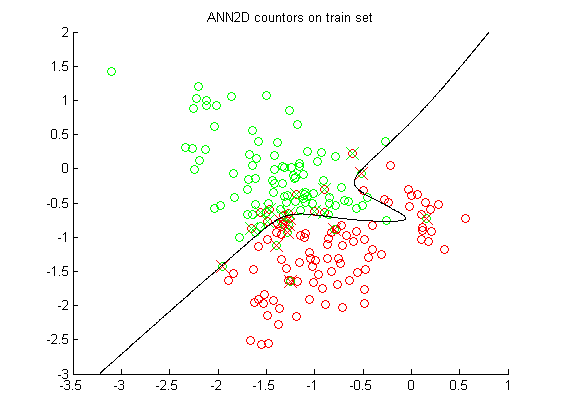


Figure 28 ANN 2D classification in 2D for "Op" recording

Confusion matrix for the training and testing data set from the “Op” recordings.

UsePCA\_MDAFeatureReduction = 2 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 3

UseClassificationMethodEnd = 3

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 0

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 0, 0); % “Op”

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 1, 1); % “Ned”

ANN3D.m – parameters

outputfunc = 'softmax';

nout = 3; % Number of outputs.

% Parameters to vary

nhidden = 12; % Number of hidden units.

alpha = 0.001; % Coefficient of weight-decay prior.

In the next experiment we have used Multilayer neural networks with three classes. We are using PCA features reduction to reduce the 12 MFCC to 6 features. By plotting the eigenvalues Figure 29 we see that the first, second, third and fourth have the biggest values.

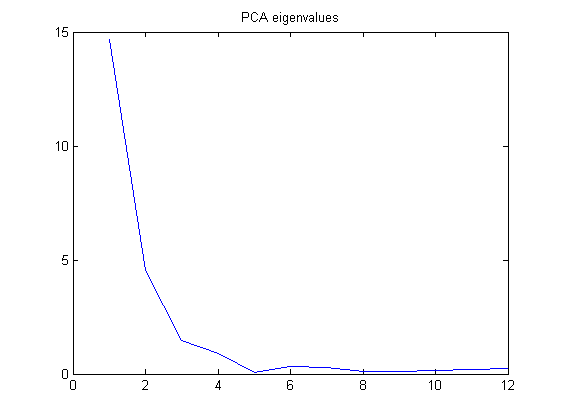


Figure 29 Eigenvalues from the Principal Component Analysis

We have selected 6 features in trying to improve the results using ANN for 6 dimensions in classification of 3 output classes. The softmax output activation function is used since we now have more than 2 classes. We have selected 12 hidden units for the ANN 6D network.

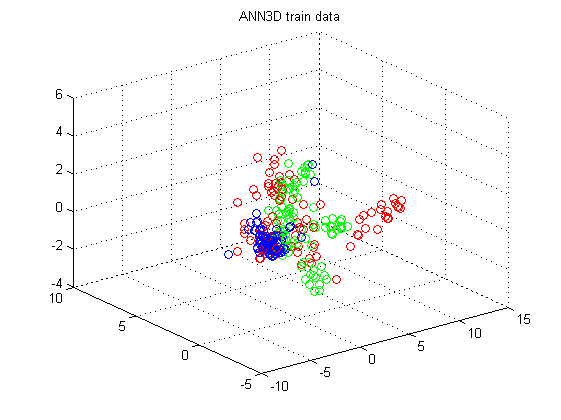
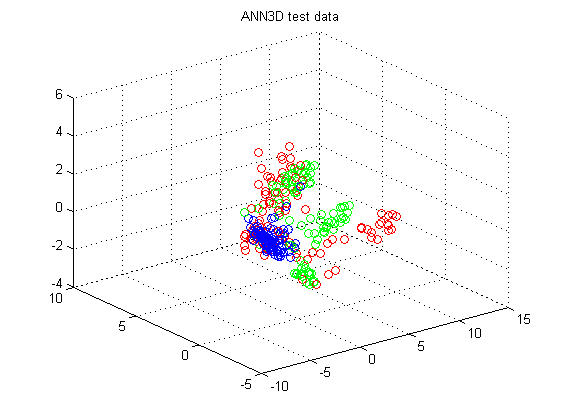


Figure 30 Training and test set in 3D for "Op" recording

Looking at the number training and test errors as function of training cycles we see in Figure 31 that an optimal number of cycles are between 40 – 60 cycles where the test errors starts to increase.

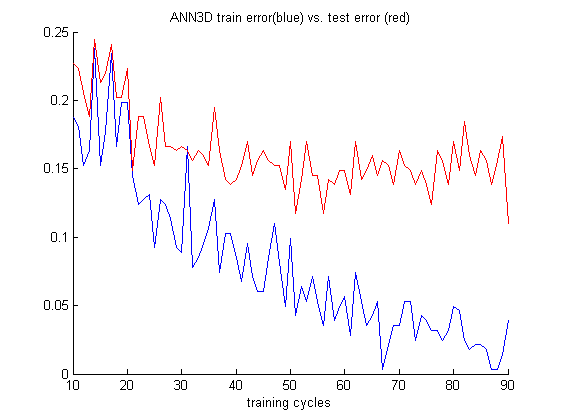


Figure 31 Errors as functions of training cycles for training and test set

## Bayesian Classifier/ probabilistic classifier

UsePCA\_MDAFeatureReduction = 2 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 4

UseClassificationMethodEnd = 4

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 0

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 0, 0); % “Ned”

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 1, 1); % “Op”

The probabilistic classifier is, opposed to the previously presented classifiers, ‘soft’ assignment, which means that each classification is given a probability value, e.g. “we are 92% certain this sample belongs to *c1*.”

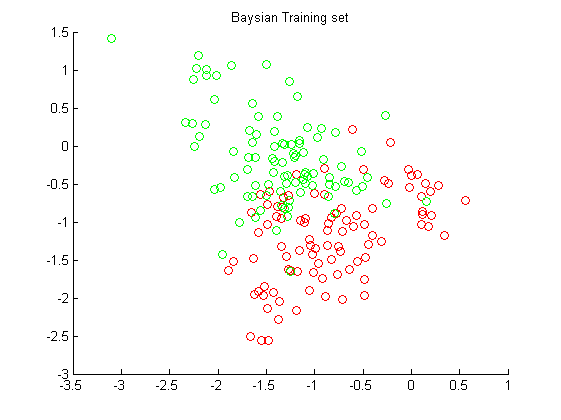
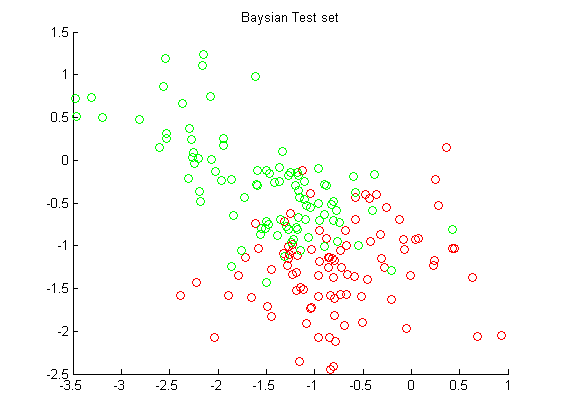
 

Figure 23 Training and test set in 2D for "Op" recording

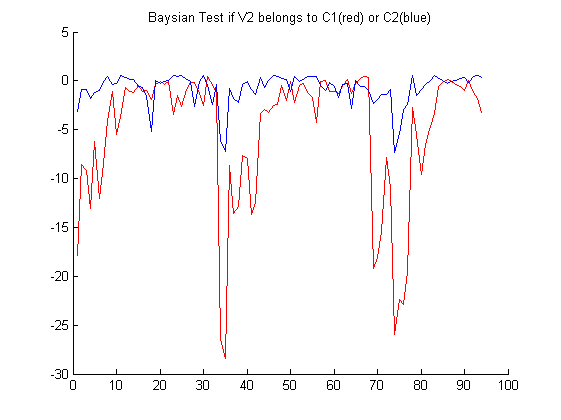
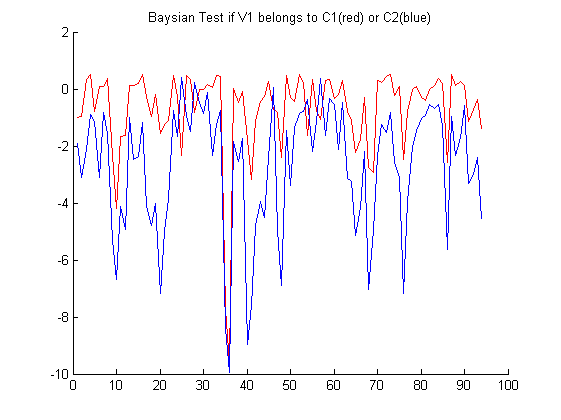


Figure 24 Bayesian probability classification for voice 1 and voice 2 belonging to class 1 and 2 on test set

The Bayesian plots depict the logarithmic probability of V1 (Voice1) and V2 (Voice2) belong to one or the other class. We have attempted to classify into two classes, C1 containing V1 samples and C2 containing V2 samples. As we can see on the plots, the probability that a V1 sample belongs to C1 is generally higher than the probability it belongs to C2 and vice versa. This gives us the information necessary to make the classification.

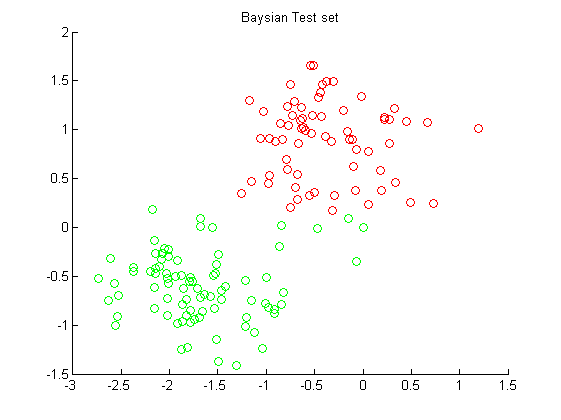
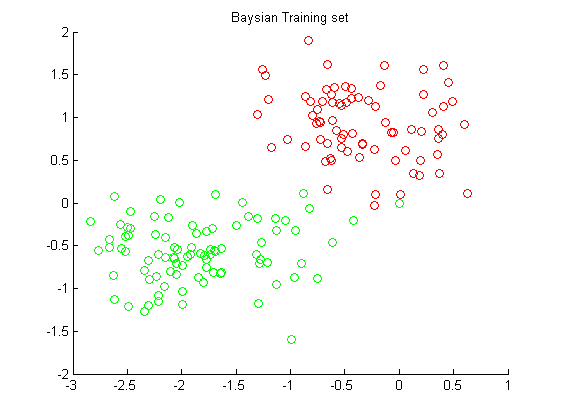


Figure 25 Training and test set on MDA feature reduction in 2D for randomized "Ned" recording

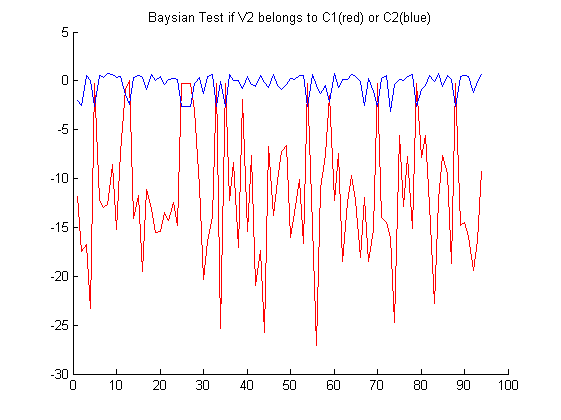
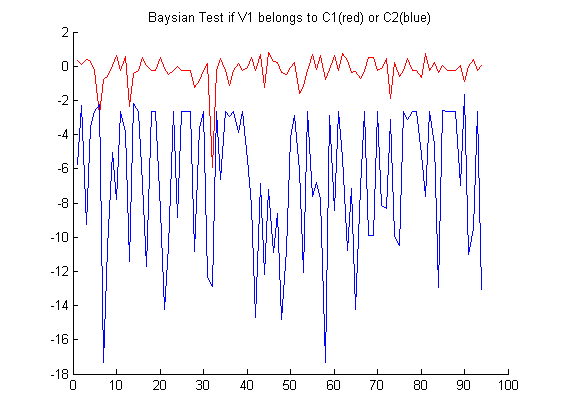


Figure 26 Baysian probability classification for voice 1 and voice 2 belonging to class 1 and 2 for “Ned” recording

The result shows that we achieve a better result on the “Ned” recordings even though it is randomized, this is due to that the feature set of voice 1 and 2 are better separated.

## Gaussian Mixture Model

UsePCA\_MDAFeatureReduction = 2 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 5

UseClassificationMethodEnd = 5

UseSizeTrainSet = 94

UseSizeTestSet = 94

UseRandomisation = 0

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 1, 1); % “Ned”

GMM2D.m – parameters

dimensions = 2;

ncentres = 3; % number of mixtures - try using e.g. 3, 5 and 7..

covartype = 'diag'; % covariance-matrix type.. 'spherical', 'diag' or 'full'

mix = gmm(dimensions, ncentres, covartype);

opts(3) = 0.001; % stop-criterion of EM-algorithm

opts(5) = 1; % do reset covariance matrix in case of small singular values.. (0=don’t reset..)

opts(14) = 100; % max number of iterations

[mix, opts, errlog] = gmmem(mix, data, opts);

This chapter presents the results performing using Gaussian Mixture Models (GMM) for generative unsupervised classification. We have chosen to use MDA feature reduction in achieving a feature dimension of 2 using recordings from voice 1, 2 and silence as illustrated in Figure 29. The three feature set of these recordings are combined in one training set to evaluate unsupervised training in (GMM2D.m).

data = [Ynew(:,[1 2]); Wnew(:,[1 2]); Znew(:,[1 2])];

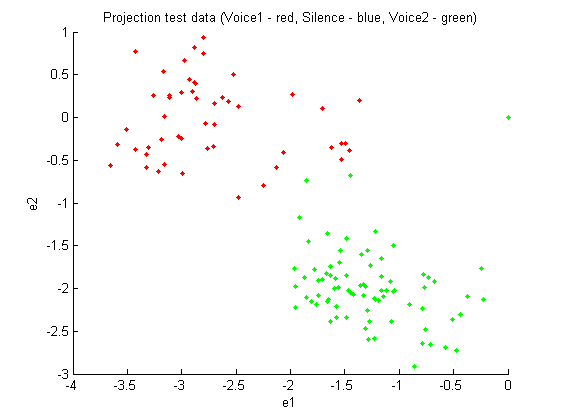
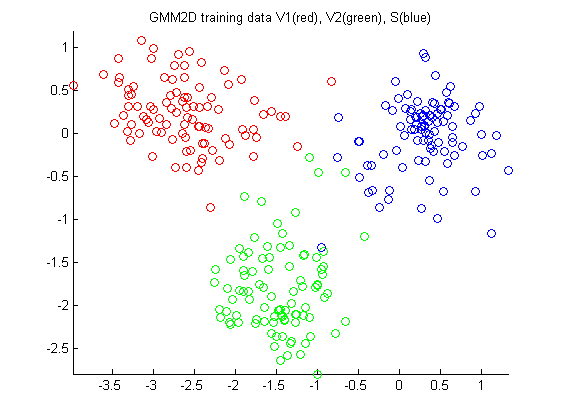


Figure 28 Training and test set in 2D for "Ned" recording

Selecting a mixture of components where we have 3 Gaussian centers we are able to let the GMM algorithm to automatically identify the 3 classes as illustrated in Figure 29. The covariance matrix is selected using the diagonal type as we can see in the resulting mixtures.

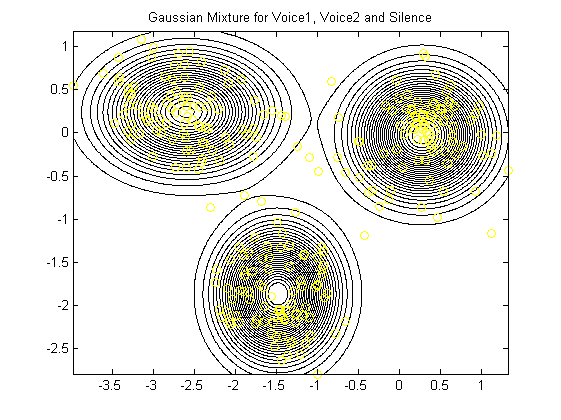
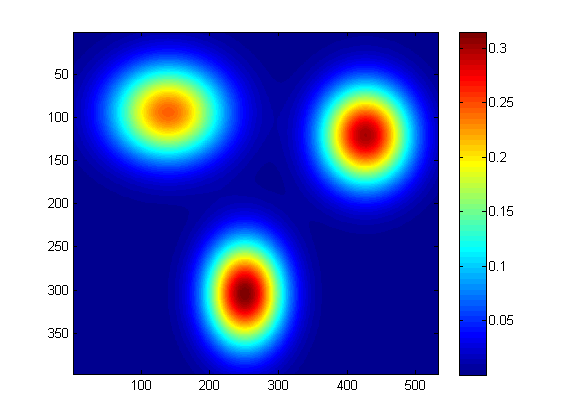
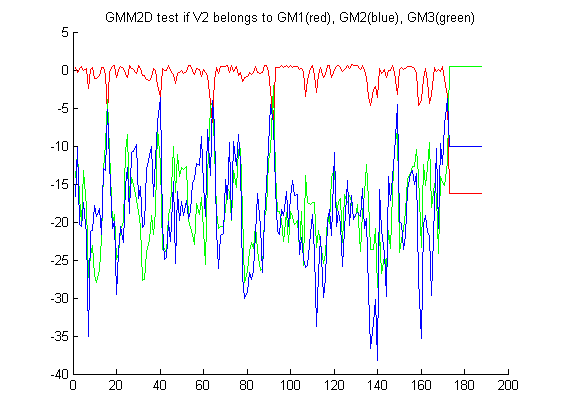
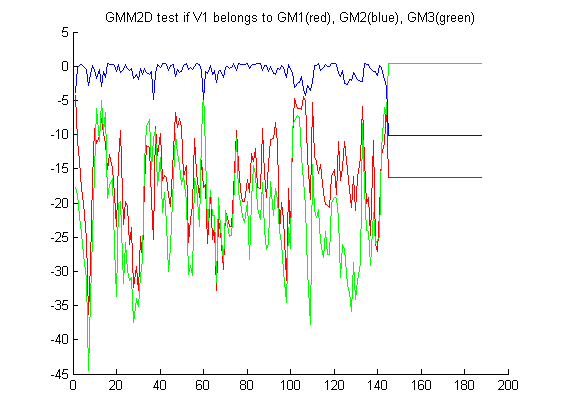


Figure 29 GMM in 2D for unsupervised "Ned" recording

The resulting GMM mixture in MATLAB is listed below:

mix = type: 'gmm’, nin: 2, ncentres: 3, covar\_type: 'diag', priors: [0.3246 0.3415 0.3339], centres: [3x2 double], covars: [3x2 double], nwts: 15



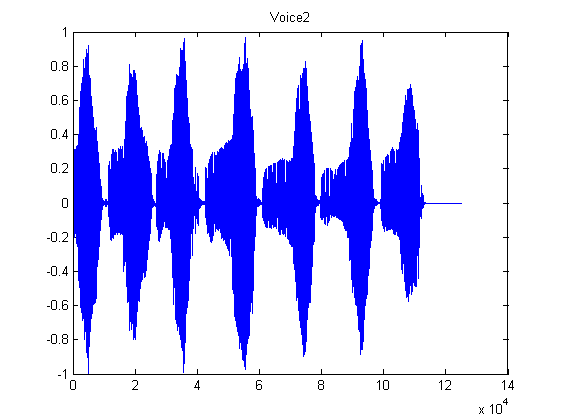
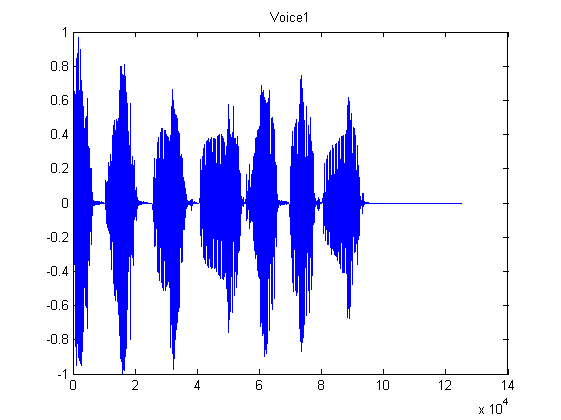


Figure 30 Unsupervised GMM Classification of voice 1 and 2

The Bayesian plots depict the logarithmic probability of V1 (Voice1) and V2 (Voice2) belong to one or the other class. We have attempted to classify into three classes, GM1 containing V1 samples and GM2 containing V2 samples and GM3 containing silence. As we can see on the plots, the probability that a V1 sample belongs to GM1 is generally higher than the probability it belongs to GM2, GM3 and vice versa. This gives us the information necessary to make the classification. The audio recordings are plotted to compare against the Bayesian plots. Here we can see that the silence parts of the recordings are correctly classified. The confusion error we get on the test set is illustrated below.

Here we are not considering the silence part of the recordings resulting in a higher error than expected according to the Bayesian plots.

## Generative and discriminative models on all recordings

The final experiment we have made is illustrated in the following. Here we will like to conclude on comparing a generative and discriminative method used on all voice 1 and 2 recordings.

UsePCA\_MDAFeatureReduction = 2 % 0=none, 1=PCA, 2=MDA

UseClassificationMethodStart = 7

UseClassificationMethodEnd = 7

UseSizeTrainSet = 1700

UseSizeTestSet = 100

UseRandomisation = 1

[mfcc\_voice1 mfcc\_voice2 mfcc\_silence] = CreateMFCCSamples(0, 0, 0, 5); % “All recordings”

………

[Ctrain, Ctest] = GMM2DComponents(V1new, V1tnew, V2new, V2tnew, 5); % ncentres = 5

GMM2DComponents.m – parameters

dimensions = 2;

covartype = 'diag'; % covariance-matrix type.. 'spherical', 'diag' or 'full'

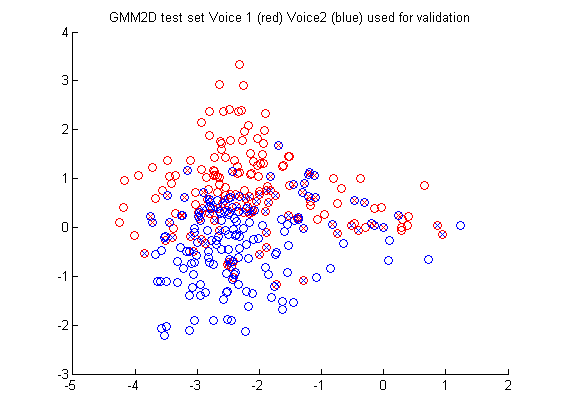
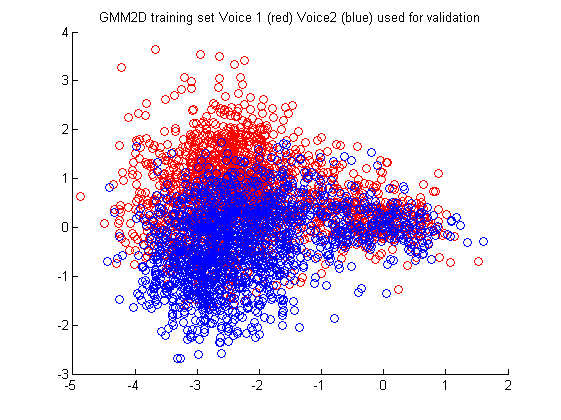
mix = gmm(dimensions, ncentres, covartype);

opts(3) = 0.0001; % stop-criterion of EM-algorithm

opts(5) = 1; % do not reset covariance matrix in case of small singular values.. (1=do reset..)

opts(14) = 100; % max number of iterations

First we will try to use the GMM in combination with supervised learning. We have extending our training to cover all recordings including in total 1700 samples as plotted in Figure 31. A Gaussian mixture model with 4 components is selected. Training is performed for each known class that includes the voice 1 and voice 2 of the sample feature set. A test set of 150 samples are used for generative classification using the 4 Gaussian components from each class. The Bayesian formula is used in finding the mixture of 2x4 that has the highest likelihood that the test sample belongs too.



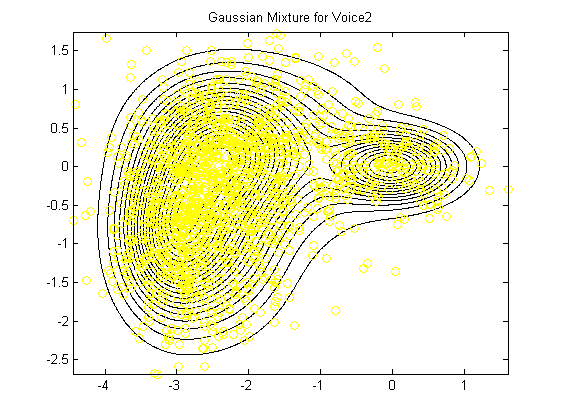
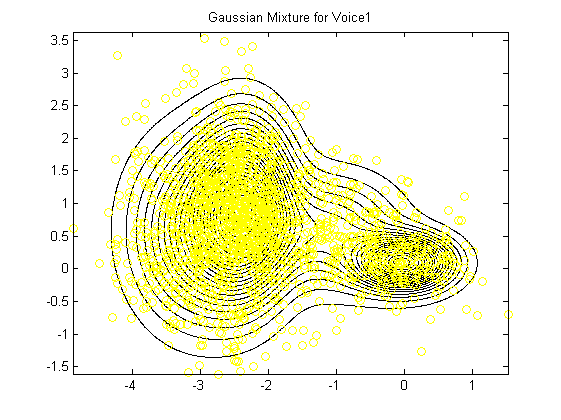


Figure 31 Supervised GMM 2D Classification of voice 1 and 2

By this method we get a classification error of 0.28.

If we use the same setting as described above for the ANN 2D method we get nearly the same error.

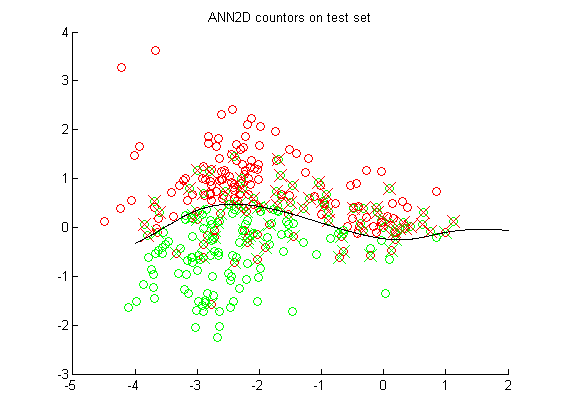
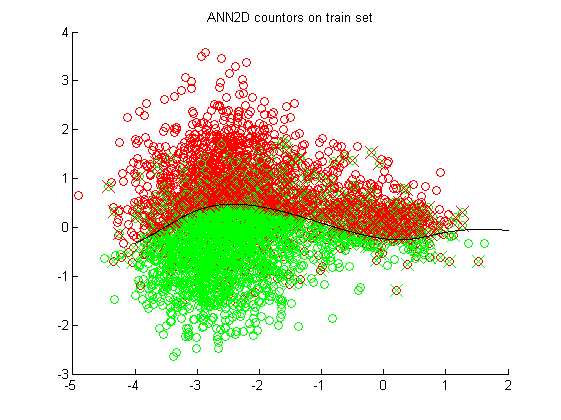


Figure 31 Supervised ANN 2D Classification of voice 1 and 2

# Discussion of Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Method*** | ***Reduction*** | ***Train “Ned”*** | ***Test “Ned”*** | ***Train “Op”*** | ***Test “Op”*** |
| 0.Linear 2D | PCA | 0.26 | 0.48 | 0.45 | 0.39 |
| 0.Linear 2D | MDA | 0.01 | 0.23 | 0.14 | **0.16** |
| 1.Linear 3D | PCA | 0.44 | 0.52 | 0.42 | 0.52 |
| 2.ANN 2D | PCA | 0.14 | **0.20** | 0.14 | 0.21 |
| 2.ANN 2D | MDA | 0.00 | 0.25 | 0.13 | **0.19** |
| 3.ANN 6D | PCA | 0.004 | 0.27 | 0.02 | **0.18** |
| 4.Probalistic 2D | PCA | - | **0.20** | - | 0.27 |
| 4.Probalistic 2D | MDA | - | **0.09** | - | **0.13** |
| 5.GMM 2D | PCA | - | 0.51 | - | 0.61 |
| 5.GMM 2D | MDA | - | **0.05** | - | 0.46 |
| 6.GMM 3D | PCA | - | 0.37 | - | 0.57 |
| 7.GMM 2D Comp | PCA | - | 0.34 | - | **0.18** |
| 7.GMM 2D Comp | MDA | - | **0.09** | - | **0.16** |

* PCA vs MDA

MDA is a more demanding procedure than PCA due to the requirement of supervised learning, where PCA can perform unsupervised. However, as can be seen when comparing the plots and classification results based on PCA and MDA, MDA is superior when the goal is to classify data and should be the preferred procedure for dimensionality reduction whenever possible.

* Randomisering af data – hvorfor? Hvad effekt forventer/ser vi?

We were asked to perform the operations on both randomized and non-randomized sets of data to check if the samples were mutually dependent. The representation of these samples should be invariant whether we present the neighbouring samples in the same set or not, so we would not expect any difference. In reality, the only difference was how the samples were projected on the 2D feature space. Separation/overlap, which should be the criticial metric for classification, remained largely unchanged.

* Valg af data set, varians i textindhold vs varians stemme. Kommer vi til at klassificere samples efter phonemer i stedet for talers identitet når vi laver MDA/PCA?

As could be seen when performing PCA on the data sets, samples of the same class tended to ‘cluster’ in groups, which led to hard times classifying the samples into voices. We believe this was due to phoneme variance. For MDA we did not see this effect as long as the data sets contained only few different phonemes. We also attempted to classify voices using data sets containing up to 20 different phonemes. For PCA it proved to be utterly impossible, and also the results of the MDA were suffering from the great variance.

* Valg af features, er MFCC det rigtige? Er det nok?

MFCC is usually utilized when performing speech recognition, the process of translating an audio signal into text. When we decided to use it for speaker recognition in the first place, it was mostly due to the fact that it was a proven method for generating features in a speech-related scenario.

* ANN, variation af parametre, overfitting,
* GMM, en kommentar på at der er en fejl et sted
* Hovedspørgsmålet: *kan vi klassificere samples ud fra hvem der taler?*

# Future Work

* Sample Dependency, vi antager uafh. Men det passer ikke, går vi glip af information ved denne antagelse?

One aspect that was left unexplored was that of time variance – we’ve assumed that there is no correlation between our samples in time but this is indeed not true; If a sample would be classified as belonging to one voice, each neighbouring sample would have a high probability of belonging to the same voice, due to the fact that no human only speaks for 50ms at a time! Introducing a model that incorporates dependency between samples, such as the Markov models, could add a reluctance to the system

* Videre udforskning af mulighederne ved GMM
* Brug af andre features end MFCC
* Valg af data sets *continued*

# Conclusion

1. Davis and Paul Mermelstein. Steven B. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.
2. M. H. Hayes. *Statistical Digital Signal Processing and Modeling*. J. Wiley & Sons, Inc., New York, 1996.
3. Richard O. Duda, Peter E. Hart, David G. Stork, Pattern Classification, Wiley, 2001, ISBN 978-0-471-05669-0