Speech Features for Optimal Discrimination of Phonemes

Andreas Kettner

Semester Thesis SA-2012-04

Computer Engineering and Networks Laboratory

Advisers:
Dr. Beat Pfister und Sarah Hoffmann

Responsible Professor:
Prof. Dr. Lothar Thiele

June 28, 2012

Summary

In the present work it is investigated how a transformation which aims at making mel-frequency cepstral less speaker-dependent for better phoneme recognition can be implemented using artificial neural networks. While traditional approaches try to achieve speaker-independence by adapting the computation of the speech features, this study introduces the use of coordinate transformations and investigates how artificial neural networks can be trained to learn these transformations.

The results of the conducted research show that the proposed neural network setup is indeed capable of learning a transformation with the desired properties. Evaluations using the Fisher distance confirm that the phoneme discriminability is increased significantly when the obtained transformation is applied to speech features from speakers not present in the training set.

Contents

List of Figures

List of Tables

1 Introduction

Speech recognition, and phoneme recognition in particular, aims at detecting *what* a speaker is saying by computing speech features from the audio signal and analyzing them. Unfortunately, speech features (for example the very popular mel-frequency cepstral coefficients) do not only capture *what* is actually said, but are also influenced by *who* is speaking and *how* the utterances are spoken. This introduces undesired variance into the speech features and deteriorates the recognition rate achievable with speech recognizers. To make things worse, the vocal characteristics mentioned above do not only vary among different speakers (these variations are caused by the gender, age, speaking habits etc. of each speaker), but also for the same speaker (due to the state of health, emotional status, etc.).

Hence, it is clearly desirable to have speech features which abstract from the undesired speaker characteristics and focus on the desired phoneme characteristics. So far, the research efforts focussed on adapting the computation of the speech features by computing more accurate estimates of the signal spectrum, applying different kinds of filterbanks and so on.

In contrast, this works investigates how coordinate transformations can be applied to make speech features more robust against the influence of undesired speaker characteristics. More specifically, a close look is taken at how an artificial neural network can be configured and trained to learn such a transformation. The Fisher distance is used to evaluate the phoneme discriminability of the conventional and the newly obtained speech features.

This report is organized as follows. Section 2 starts with a description of the task and explains what steps need to be taken. Section 3 looks at how mel-frequency cepstral coefficients are computed, introduces the artificial neural network setup used in this work and deals with how discriminability can actually be measured. In Section 4, the conducted experiments are presented along with the analysis of the obtained results. The conclusion of this work is formulated in section 5, including some possibilities for future work.

2 Problem Statement

The goal of speech recognition is to recognize which phonemes occur in a speech signal. To do so, a feature extractor computes a spectrum from a windowed segment of the signal and calculates certain features which capture the spectral characteristics. The extracted features are then passed to a speech recognizer which analyses them and determines by which phoneme they have most likely been produced.

However, in addition to the desired information about the phonemes themselves, the signal (and hence the features) also contains undesired information about the voice of the speaker. This is due to the fact that the way a person speaks is affected by various factors as the speaker's gender, state of health, emotional status, dialect, speaking habits and so on. Unfortunately, these characteristics do not only vary among different speakers (gender, age, etc.) but also for the same individual (state of health, emotional status, etc.), which makes it particularly difficult to train a reliable speech recognizer.

Since speech recognition only takes interest in *what* is said and not in *who* says it or *how* it is said, it is desirable to have speech features with a strong dependence on the phoneme information and a weak dependence (or ideally no dependence at all) on the speaker. In other words, one would like to eliminate all variance within the features which is not due to the phonemes themselves such that only the relevant information is still contained in the feature vectors.

Basically, there are two different ways to achieve this: one could either come up with a completely new set of features which have the desired property or develop a transformation for existing features which yields transformed features with the desired property.

Obviously, the first approach is rather unfavourable because one can not utilize experience gained previously from working with established speech features. Moreover, the actual effect of the speaker characteristics on the features is not known in detail, which makes it almost impossible to counteract them.

However, a transformation can for example be found by training an artificial neural network. This comes with some advantages. To start with, one can base one's work on well-understood speech features which are already optimised to a certain degree. Furthermore, one does not have to know the actual effect of the speaker on the features in detail because the artificial neural network itself can eliminate it in the learning process (given that enough training data is available).

This work investigates the transformation approach. The features to be transformed are mel-frequency cepstral coefficients (MFCC) which are widely used in the field of speech recognition. It shall be investigated whether and how an artificial neural network (ANN) can be trained to learn a feature transformation with the desired properties and to what extent the discriminability of phonemes can be improved.

The following tasks are to be completed (confer to appendix A for the handout of the task description).

First of all, a framework which allows to evaluate and compare the quality of speech features is to be developed. The investigation is restricted to phonemes of the English language. However, all phonemes are to be taken into consideration for the evaluation. Specifically, recordings from the TIMIT database are to be used for this task.

Secondly, the computation of the MFCCs needs to be optimised prior to transformation. Using the evaluation framework developed in the first phase, it shall be investigated to what extent modifications in the computation of the MFCCs alone can reduce the speaker-dependence and increase the phoneme discriminability.

Furthermore, an ANN setup capable of learning the desired feature transformation needs to be developed. Design decisions include number of layers, number of neurons per layer, connectivity and so on. To implement the ANNs, the NICO toolkit shall be used.

Moreover, the training of ANNs requires suitable training-, validation- and test-data. As in the previous steps, these data can be taken from the TIMIT database. After defining the required datasets, the training of the ANNs can be carried out with NICO.

Finally, it has to be investigated whether and how much the found transformation improves the phoneme discriminability using the developed evaluation framework.

3 Approach

3.1 Mel-Frequency Cepstral Coefficients (MFCC)

In the following, it is briefly presented how MFCC are actually computed (for a more in-depth discussion, confer to [16]). Options for optimisations are discussed in section 4.

Computation The process of feature computation aims at extracting information from the signal spectrum in a reduced form more suitable for classification. Feature extraction often mimics the way how the ear processes incoming sound because human hearing obviously works very well (hence, one can possibly profit from imitating its functionality).

For example, psychoacoustic investigations have shown that the intensities of signals in narrow frequency bands are combined into one overall intensity for the respective band and that the perceived pitch is logarithmic with frequency.

Therefore, it makes sense to introduce a nonlinear frequency scale which attributes to these characteristics for feature computation: the mel-scale. If we denote the frequency in Hertz by f and the perceived pitch in mel by h, the following equation holds

$$h(f) = 2595 \cdot log_{10}\left(1 + \frac{f}{700\ \text{Hz}}\right) \tag{1}$$

However, one does not normally use this relation to obtain the mel-spectrum. Rather, frequency-domain filtering is performed on the spectrum obtained from discrete Fourier transformation (DFT) with a mel-filterbank (as depicted in figure 1), which consists of non-uniformly spaced triangular filters of different bandwidth. If we denote the triangular filters by H_j $(1 \leqslant i \leqslant J)$ and the DFT-spectrum of the signal by $X(k)$ $(0 \leqslant k \leqslant N-1)$, the mel-spectrum S_j $(1 \leqslant j \leqslant J)$ is given by

$$S_j = \sum_{k=0}^{N-1} X(k) \cdot H_j(k) \qquad (1 \leqslant j \leqslant J) \tag{2}$$

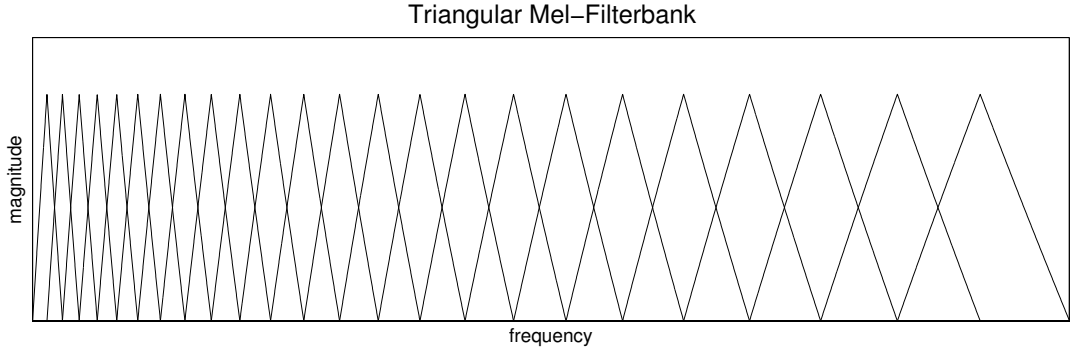


Figure 1: Mel-filterbank with J filters (usually $J = 24$)

The cepstral coefficients $c(m)$ $(0 \leqslant m \leqslant D)$ are then obtained from the mel-spectrum by computing the discrete cosine transform of the mel-cepstrum by

$$c(m) = \frac{1}{J} \sum_{j=1}^{J} \log\left(S_j\right) \cos\left(m \left(j - \frac{1}{2}\right) \frac{\pi}{J}\right) \qquad (0 \leqslant m \leqslant D) \tag{3}$$

where D is the number of cepstral coefficients (usually $D = 13$).

3.2 Artificial Neural Networks (ANN)

To improve the quality of speech features (given as feature vectors $\mathbf{x}$) one may try to find a transformation $\Psi : \mathbf{x} \mapsto \mathbf{y} = \Psi(\mathbf{x})$ that minimizes the dependency on the speaker information and maximizes the dependency on the phoneme information.

The key idea of this work is to estimate the transformation Ψ by training an artificial neural network (ANN) such that it learns an approximation $\tilde{\Psi}$ of this transformation. If a sufficiently high number of input/output-pairs $(\mathbf{x}, \mathbf{y})$ of the transformation Ψ is available, an ANN can easily be trained to learn $\tilde{\Psi}$.

Unfortunately, output vectors $\mathbf{y}$ of the transformation Ψ are not available. As the portion of the features only due to the undesired speaker information is unknown, it can not be eliminated and hence no "cleansed" feature vectors can be produced.

However, it is in principle still possible to perform a training with a slightly different setup. A shown in figure 2a, the transformation net NN_T can be integrated into a larger system of nets NN_S. The system contains two identical instances of the transformation net NN_T, which perform the coordinate transformation on the input vectors x_1, x_2 to obtain y_1, y_2. The subsequent stage NN_D computes the distance D between the transformed vectors y_1 and y_2. The overall system can be taught to output a small distance d_0 if the input vectors x_1, x_2 stem from speech signal segments of the same phoneme and a large distance d_1 if the input vectors x_1,x_2 stem from speech signal segments of different phonemes.

To train NN_S, one simply has to define a sufficiently large number of input/output pairs consisting of speech feature samples x_1,x_2 from many different speakers and the corresponding output value d_0 or d_1. If the overall systems manages to learn this discrimination with sufficient adequacy, it is reasonable to assume that the subnets in turn have learned the desired coordinate transformation.

This work uses the NICO toolkit (project homepage under [18]) to define and train the artificial neural networks. The problem with the setup as it is shown in figure 2a is that the weights of the two instances of NN_T have to be the same at any time during and after the training (otherwise the two subnets would not perform the same transformation). As the NICO toolkit does not provide the ways and means to support such a configuration, a slightly different configuration as in figure 2b is proposed. Here, the overall system only contains one single instance of the transformation net, thus the need to provide two identical instances is eliminated. As there is now only space for one single input vector at a time, the vectors x_1, x_2 which the system seeks to discriminate are fed in during two subsequent cycles. By

inserting delay units between NN_T and N_D (delay 0 and 1 cycles), it is guaranteed that NN_D has the vectors available at the same time.

3.3 Measuring Feature Quality

The goal of this work is to reduce the undesired variance within the MFCC speech features in order to make the different phoneme classes more easily distinguishable. Hence, a quality measure is needed to quantitatively evaluate the discriminability of the data. Some such measures are presented in the following.

Self- and Cross-Distance Let us assume that data of different classes are given as vector samples in $\mathbb{R}^n$ where $n \in \mathbb{N}$ is an arbitrary dimensionality. By trend, data points belonging to the same class will be grouped rather close together, thereby forming clusters ("clouds") in space. However, there may be some regions where different clusters touch or even overlap depending on how strongly the data are scattered (scattering may for instance be due to measurement inaccuracies, noise and other distortions).

Intuitively, one would say that the different classes are well-distinguishable if the data of each single cluster do not scatter too much and if all clusters are spaced sufficiently far apart in space. Quantitatively, this can be expressed with the idea of *self-distances* d_{self} and *cross-distances* d_{cross}. A self-distance is computed from two feature vectors from the same class, whereas a cross-distance is computed from two feature vectors from different classes. For two classes C_A, C_B we have

$$d_{self}\left(C_A\right) = \left|\left|\mathbf{x}_{A,1} - \mathbf{x}_{A,2}\right|\right|_2 \qquad \text{where } \mathbf{x}_{A,1}, \mathbf{x}_{A,2} \in C_A \text{ and } \mathbf{x}_{A,1} \neq \mathbf{x}_{A,2} \tag{4}$$

$$d_{cross}\left(C_A, C_B\right) = \left|\left|\mathbf{x}_A - \mathbf{x}_B\right|\right|_2 \qquad \text{where } \mathbf{x}_A \in C_A,\ \mathbf{x}_B \in C_B \tag{5}$$

Note that $\left|\left|\mathbf{x}_A - \mathbf{x}_B\right|\right|_2$ computes the Euclidean distance $\mathbf{x}_A$ and $\mathbf{x}_B$.

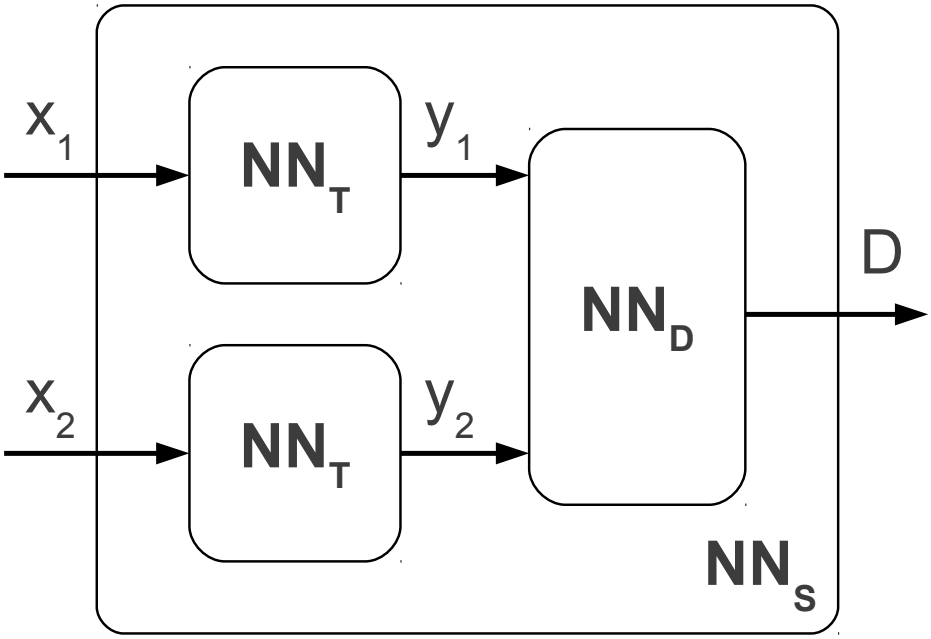
Usually, the mean of (4) and (5) is computed over all pairs $\mathbf{x}_{A,1}, \mathbf{x}_{A,2}$ and $\mathbf{x}_A, \mathbf{x}_B$ to get an average value for the cross- and self-distances

$$\mu\left(d_{self}\left(C_A\right)\right) = \frac{1}{\binom{N_A}{2}} \sum_{\mathbf{x}_{A,1} \neq \mathbf{x}_{A,2}} \left|\left|\mathbf{x}_{A,1} - \mathbf{x}_{A,2}\right|\right|_2 \tag{6}$$

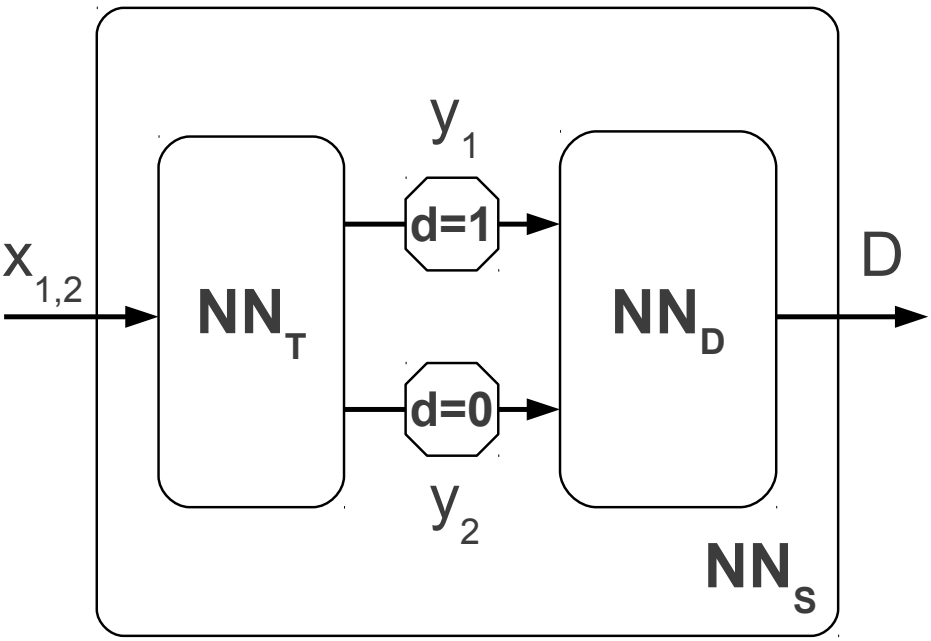
$$\mu\left(d_{cross}\left(C_A, C_B\right)\right) = \frac{1}{N_A \cdot N_B} \sum_{\mathbf{x}_A, \mathbf{x}_B} \left|\left|\mathbf{x}_A - \mathbf{x}_B\right|\right|_2 \tag{7}$$

where N_A, N_B are the total numbers of vectors contained in the classes C_A, C_B. Note that $\binom{N_A}{2}$ is the total number of feature vector pairs within the same class C_A (each pair gives one self-distance). Similarly, $N_A \cdot N_B$ is the total number of feature vector pairs from different classes C_A and C_B.

The mean self- and cross-distances as computed in (6) and (7) give a rough idea of how the data are distributed. However, it is not so easy to deduce from them how well the different classes can be distinguished from each other, so a more sophisticated measure for discriminability would be nice.



(a) Basic setup with two transformation blocks and one discrimination block



(b) Equivalent setup with one transformation block, one discrimination block and delayed connections

Figure 2: Systems for training an ANN to learn the transformation Ψ

Fisher Distance The *Fisher distance* is a better measure for discriminability based on the mean and variance of self- and cross-distances

$$\mathcal{F} = \frac{\left(\mu\left(d_{self}\right) - \mu\left(d_{cross}\right)\right)^2}{\sigma^2\left(d_{self}\right) + \sigma^2\left(d_{cross}\right)} \tag{8}$$

Note that a larger Fisher distance means better discriminability.

To obtain a large Fisher distance, the mean of the self-distances should be much smaller than the mean of the cross-distances, and the variances of self- and cross-distances should be generally small. Obviously, this corresponds exactly to what was discussed further above in the intuitive description of discriminability. $\mu(d_{self}) \ll \mu(d_{cross})$ means the classes are spaced far apart in feature space and small $\sigma^2(d_{self})$, $\sigma^2(d_{cross})$ imply that there is not much scattering.

The Fisher distance can be used to evaluate the discriminability for just two classes ("pairwise" Fisher distance) or for all classes as a whole ("overall" or "global" Fisher distance). In the first case, the computation of mean and variance only includes the self- and cross-distances of the corresponding pair of classes (as shown in equation (6) and (7)), whereas in the second case, the self- and cross-distances of all class pairs have to be taken into account.

4 Experiments and Results

4.1 Optimisation of Pre-Transformation Features

Although we seek to find a transformation which improves the discriminability of phonemes by minimizing the variance in the MFCC caused by speaker characteristics, it makes sense to perform some optimization (if possible) already prior to the transformation. Intuitively, it should be easier for an ANN to learn a transformation if the data it operates on are already somewhat optimised with regard to phoneme discriminability.

Potential for Optimisation Looking at the computation of the MFCC as presented in section 3.1, one can see that there are a lot of parameters that could in principle be varied to improve the MFCC. Basically, the following choices are free

(1) the windowing function used for signal segmentation

(2) the length of the signal segment for spectrum computation

(3) the way the spectrum is obtained

(4) the number of filters J in the mel-filterbank

(5) the shape of the filters in the mel-filterbank

(6) the number D of cepstral coefficients

Given so many degrees of freedom, some restrictions clearly need to be made.

On one hand, there are certain "standards" for the above choices in the speech processing community. For instance, the windowing function (1) is usually hamming-shaped, the mel-spaced filterbank (4) normally consists of $J = 24$ filters (confer to [16]) and commonly $D = 13$ cepstral coefficients are used (also confer to [16]). It makes sense to adhere to these standards as they are very popular in the speech processing community and hence proven to give good results.

On the other hand, much of the above choices still are the topic of ongoing research. For example, researchers have explored new robust ways of computing a spectrum for (3) and experimented with different filter shapes for the filterbank (5). Moreover, various lengths for the segmentation window (2) are in use (depending on the specific task). It clearly makes sense to look at these parameters more closely in order to determine their effect on the phoneme discriminability.

In the following, the investigations regarding the length of the segmentation window, the shape of the filters in the filterbank and the spectrum computation are presented. This includes a brief discussion of the ongoing research as well as the experiments conducted in this work.

Length of the Segmentation Window The spectrum required for the computation of the MFCC is computed on a window/frame of the speech signal. Various different window-lengths have been applied for this purpose. According to the performed literature research, the most commonly used lengths are rather short ones of roughly 15-20 ms duration (confer to [2], [5] and [3]) and longer ones of approximately 25-32 ms duration (confer to [13], [11], [1], [20], [19], [8], [4] and [17]).

What factors must be taken into account when choosing a suitable window-length? On one hand, the signal segment should be long enough to provide sufficient spectral resolution. On the other hand, it should not be too long either because temporal resolution is of course also important (in other words, the signal segment should contain one single and not several different phonemes). According to [15], a frame-length of 25-30 ms is a reasonable choice for speech recognition.

When working with the phonemes of the TIMIT database (more details about the TIMIT database follow in section 4.4), one finds that many phoneme occurrences are rather short and could not make up a whole frame. There are two ways to deal with this. One could, as the authors of [6] who also work on the TIMIT database, fill up the "empty space" of frames taken from short phonemes with samples of the preceding or succeeding phoneme. However, this also somewhat dangerous as it distorts the characteristics of the phoneme actually under investigation. Therefore, the more "purist" alternative is to simply discard any instance of a phoneme which is shorter than the frame-length to ensure that feature computation only captures the characteristics of one single phoneme.

In this work, the more conservative approach is used. To find out what phonemes actually limit the choice of the window-length, the average length of each phoneme class was computed. The results are shown in table 1 (for an explanation of the phoneme labels please confer to section 4.4).

Obviously, the phonemes `b`, `d`, `dx` and `g` with average durations of 17, 25, 28 and 29 ms are the most critical. The extremely short average duration of 17 ms of `b` seems to speak in favour of using a shorter window of less than 20 ms. However, it was found during this work that even with a window of 25 ms duration sufficiently many occurrences of `b` can be provided to conduct useful experiments. Hence, a window-length of 25 ms is chosen (which is in accordance with the optimum stated in [15]).

Shape of the Mel-Spaced Filter Bank The purpose of the mel-filterbank is to model the logarithmic relationship between perceived pitch and frequency and to account for the fact that the energy contributions of signals within certain bands are combined into one value. The choice of implementing such a filterbank with triangular filters (as described in section 3.1) is however not at all sacrosanct and unalterable. In fact, the triangular shape is just the simplest one available, but other shapes may very well be used. It is therefore not surprising that some research has been dedicated to optimise the shape of the mel-filterbank.

Phoneme	Duration	Group	Duration
aa	127 ms	k	52 ms
ae	138 ms	m	63 ms
ah	63 ms	n	53 ms
aw	175 ms	ng	59 ms
axr	98 ms	ow	129 ms
ay	153 ms	oy	185 ms
b	17 ms	p	44 ms
ch	88 ms	pau	92 ms
d	25 ms	r	58 ms
dh	37 ms	s	114 ms
dx	28 ms	sh	124 ms
eh	90 ms	t	45 ms
el	67 ms	th	78 ms
ey	135 ms	uh	75 ms
f	105 ms	uw	93 ms
g	29 ms	v	60 ms
hh	67 ms	w	61 ms
ih	59 ms	y	59 ms
iy	87 ms	z	85 ms
jh	64 ms		

Table 1: Average phoneme durations (TIMIT database)

The authors of [3] used a data-driven approach where shapes, bandwidths and positions of the filters in the bank can be tuned in order to minimize the classification error. This way, a recognizer (consisting of feature extractor and classifier) can be trained to minimize recognition error.

In [10], the authors proposed that the shape alone of each filter in the bank can be derived by applying principal component analysis (PCA) on the dataset. They sought to maximize both the signal-to-noise variance ratio for the output of the filter and the variation of this output among the different phoneme classes, but to keep the variation among speakers low. The improvement of their method compared to standard methods was found to be only minor when considering clean speech, but more substantial in noisy speech.

Finally, in [5] the authors used Gaussian shaped filters instead of triangular ones because they provide smoother transitions from one subband to the other and because the overlap between the filters can be easily controlled. Compared to the standard triangular filterbank approach, their method provided some improvement, but it was not tremendous.

It is clearly beyond the range of this work to implement large and complicated frameworks dedicated solely to the improvement of the filter shapes. However, looking at the efforts made in [3], [10] and [5], it is still deemed worthwhile to explore the effect of at least a few different filter shapes.

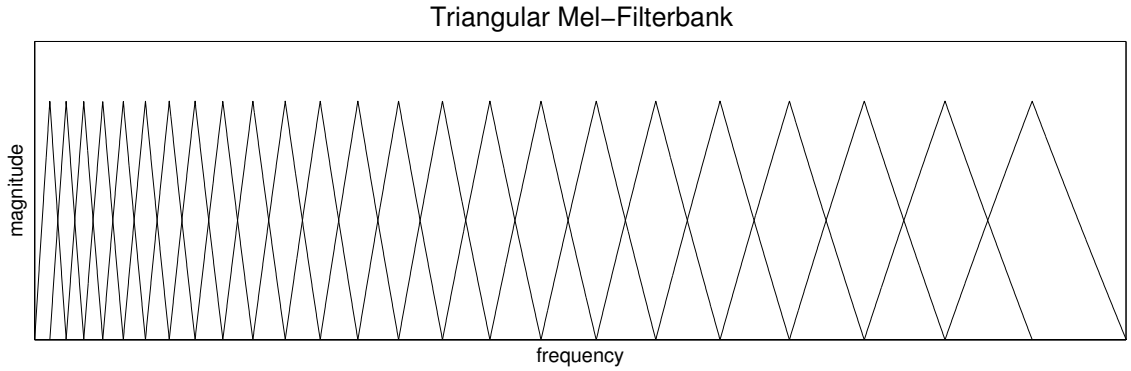
Fortunately, there is a matlab function `mfcc(...)` available which allows to easily produce triangular, hamming- and hanning-shaped filterbanks (please confer to figure 3 for example plots). Obviously, hamming- and hanning-shaped filters are also smoother than triangular filters and presumably offer advantages similar to the Gaussian filters described in [5]. So, if the filter shape does indeed have an effect on the phoneme discriminability, the investigation of these simple filter shapes should still reveal it.

To investigate how the use of Hamming or Hanning filters instead of triangular filters can improve the discriminability of the phonemes, MFCC derived from filterbanks with the respective shapes are computed using the following basic settings

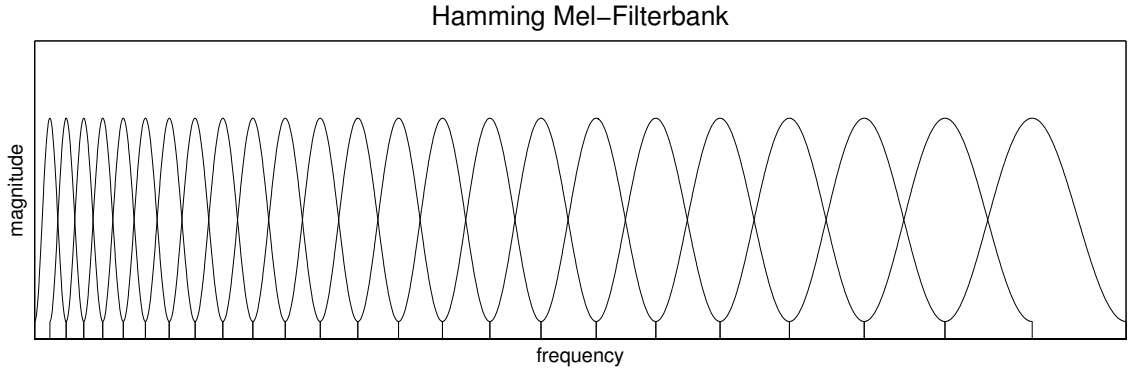
- segmentation of the signal into blocks of 25 ms
- spectrum is obtained from the DFT of the signal frame
- $J = 24$ filters in the mel-filterbank
- $D = 13$ cepstral coefficients

MFCCs are extracted from the records of the TIMIT database (confer to section 4.4 for more information about TIMIT) and the global Fisher distance is computed for each filter shape. The results are shown in table 2.

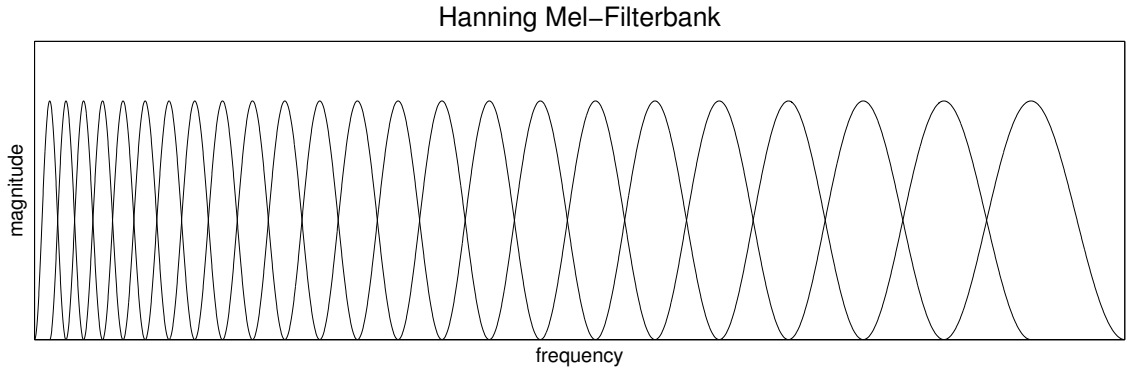
The use of Hamming and Hanning filters in the bank increases the global Fisher distance a little bit, but the improvement is negligible. Hence, it must be concluded either that the used filter shapes do not have a significant influence on the



(a) Triangular mel-filterbank



(b) Hamming mel-Filterbank



(c) Hanning mel-filterbank

Figure 3: Choices of shapes for the mel-filterbank investigated in this work

Filter Shape	Fisher Distance
triangular	0.6759
hamming	0.6761
hanning	0.6765

Table 2: Global Fisher Distance for the different filterbank shapes

discriminability or that the Fisher distance is not sensitive enough to capture the influence. In any case, the results do not justify the use of a different filterbank, so the traditional triangular filterbank is taken for the MFCC computation.

Multitaping for Spectrum Computation Usually, the spectrum required for the extraction of the MFCCs is obtained by computing the DFT of a windowed excerpt of the speech signal. However, one is of course free to apply other spectrum computation methods.

The windowed DFT has the disadvantage that its variance is relatively high (the shape of the spectrum may change quite much when the window is shifted only a little bit). To counteract this effect, the so called *multitaper* method (confer to [8] and [1]) has been developed. Multitaping is basically an extension of the windowed DFT which uses multiple windows (called *tapers*) of different shapes instead of just one. Each taper produces a subspectrum of the frame. In the end, the different subspectra are averaged in the frequency-domain to obtain the overall spectrum. This frequency-domain averaging makes the multitaper spectrum robust against variations. The authors of [1] could show that the multitaper method performs well compared to the windowed DFT, especially for clean speech.

When investigating the effect of the multitaper method, one could in principle vary the shape of the tapers as well as the number of tapers. However, as a matlab implementation of the so-called sine-weighted cepstrum estimator (SWCE) tapers is available from [7], the investigation of this work is confined to these.

In order to determine the effect of multitaping on the phoneme discriminability, spectra of frames (with 25 ms length) from the TIMIT database are computed by applying the SWCE multitaper method with a varying number of tapers. From these multitaper spectra, MFCCs are computed using the following settings

- $J = 24$ triangular filters in the mel-filterbank
- $D = 13$ cepstral coefficients

To quantitatively evaluate the effect of multitaping, the global Fisher distance is calculated for each multitaper setup. The results can be found in table 3.

By trend, the Fisher distance increases with the number of tapers. However, the increase is not very large and even diminishes for large numbers of tapers. Note that compared with the results from table 2 where fisher distance was around 0.676, the values are mostly at 0.683 and above when using multitapers. Hence, at least a slight improvement can be attributed to the multitaper method.

It remains to decide how many tapers shall be used. Going above 6-7 tapers does not bring much more improvement. As the investigations in [8] showed that the use of more than 8 tapers tends to reduce the spectral resolution ("smearing"), the use of 6 tapers seems reasonable. This provides us at least with some improvement compared to the windowed DFT and should not compromise the quality of the obtained spectrum.

Tapers	Fisher Distance
3	0.6798
4	0.6814
5	0.6825
6	0.6835
7	0.6841
8	0.6844
9	0.6847

Table 3: Global Fisher distance for SWCE multitaping with different numbers of tapers

Decisions Let us briefly summarize the decisions made regarding the computation of the MFCCs before we move on

- segmentation of the speech signal into **frames of 25 ms** duration
- spectrum obtained from **SWCE multitaping** with **6** tapers
- bank with $\mathbf{J = 24}$ mel-spaced **triangular** filters
- computation of $\mathbf{D = 13}$ cepstral coefficients

4.2 Configuration of the Artificial Neural Networks

As described in section 3.2, the basic setup for learning a feature transformation consists of a transformation block NN_T and a distance computation block NN_D. However, the details of the implementation still need to be discussed. In the following, basic architectural considerations are explained and the detailed configuration of the ANNs used in this work is presented. The training modes and the corresponding results can be found in section 4.3 and 4.4, respectively.

Complexity of the Transformation Stage The Transformation block NN_T which implements $\Psi : \mathbf{x} \mapsto \mathbf{y} = \Psi(\mathbf{x})$ is the core of this entire work. Obviously, the performance and the "learning capability" of this ANN depend on how many layers and neurons it is made of. Hence, it is worthwhile to take a closer look at this.

For NN_T, it is mandatory to have at least an input layer and an output layer. The input layer serves to distribute the input data among the subsequent layers of neurons whereas the output layer performs the final part of the computation and thereby creates the transformed data. In between the input and the output layer, further layers of neurons may be inserted to provide the network with more flexibility and learning capability.

In principle, increasing both the number of layers and the number of neurons per layer gives the ANN more flexibility. However, one must not forget that this also calls for more training data (more weights need to be trained) and may lead to convergency problems. Hence, it does not make sense to unnecessarily "inflate" the network.

For the purpose of this work, 1-2 internal layers are deemed sufficient (confer to [9], where some general rules for choosing the number of layers are given). The transformation network with 1 internal layer corresponds to a 2-layer perceptron and is from now on referred to as *simple* configuration. The transformation network with 2 internal layers corresponds to a 3-layer perceptron and is referred to as *complex* configuration (For more information about multilayer perceptrons in general, please confer to [14],[9], [12] and [16]).

In this work, the number of neurons per layer was chosen to be 2-4 times the number coordinates ($D = 13$). For the *complex* configuration the two internal layers contain 30 and 50 neurons, and for the *simple* approach the one internal layer consists of 50 neurons (please also confer to figures 4 and 5).

Input and Output Dimensionality of the Transformation Stage When talking about the transformation $\Psi : \mathbf{x} \mapsto \mathbf{y} = \Psi(\mathbf{x})$, we have so far not mentioned the dimensionality of $\mathbf{x}$ and $\mathbf{y}$. To be more precise, the transformation should be written as

$$\Psi : \mathbf{x} \in \mathbb{R}^m \mapsto \mathbf{y} = \Psi(\mathbf{x}) \in \mathbb{R}^n \tag{9}$$

where $m, n \in \mathbb{N}$ are the dimensionality of the untransformed and the transformed feature space.

Since the untransformed feature vectors consist of $D = 13$ MFCCs, we have $m = 13$. It remains to decide what number n should be. If only pure mathematics are considered, the dimensionality of the transformed space is not important as one can always define a set of target clusters which are spaced far apart and have low variance (The corresponding transformation function may look extremely nasty and may be very hard to find, but it can still be assumed to exist).

However, the limitations inflicted by the actual implementation (artificial neural network) must of course also be taken into account. By definition, n is the dimensionality of the transformed feature vectors and hence equal to the number of neurons in the output layer of the transformation block. In contrast to the neurons of the input layer which just distribute the data among the neurons of the subsequent layers, the neurons in the output layer actually contribute to the "computational power" of the network. Hence, it seems probable that setting $n \equiv m$, $n < m$ or $n > m$ does have an influence on the networks performance.

One may argue that setting $n \equiv m$ is reasonable because the ANN can focus on reducing the undesired variance without having to worry about a dimensionality change as well. However, it is also possible that the MFCCs are very redundant such that feature vectors with less coordinates would still suffice to have an adequate description of the phonemes. In this case, it would make sense to set $n < m$ because the overall complexity of the ANN can be reduced. Finally, it may be that the MFCCs are not redundant, but the network could have insufficient computational power to learn the transformation. Hence, setting $n > m$ should be chosen to give the ANN more flexibility and possibly improve the learning performance.

In this work, all the effect of all three possible choices ($n \equiv m$, $n < m$ and $n > m$) is investigated. For the case $n < m$, it was chosen to have $n = 6$, whereas for $n > m$, $n = 20$ was taken (please also confer to figures 4 and 5). For ease of reference, the following short labels are used for these configurations

- $n > m$: higher dimensionality configuration (termed *higherdim* or *higher*)
- $n \equiv m$: same dimensionality configuration (termed *samedim* or *same*)
- $n < m$: lower dimensionality configuration (termed *lowerdim* or *lower*)

Implementation of the Distance Stage Finally, let us take a closer look at the distance computation stage NN_D. This network computes a distance measure between two transformed feature vectors. The computed distance should be large if the feature vectors stem from different phonemes and small if they stem from the same phoneme.

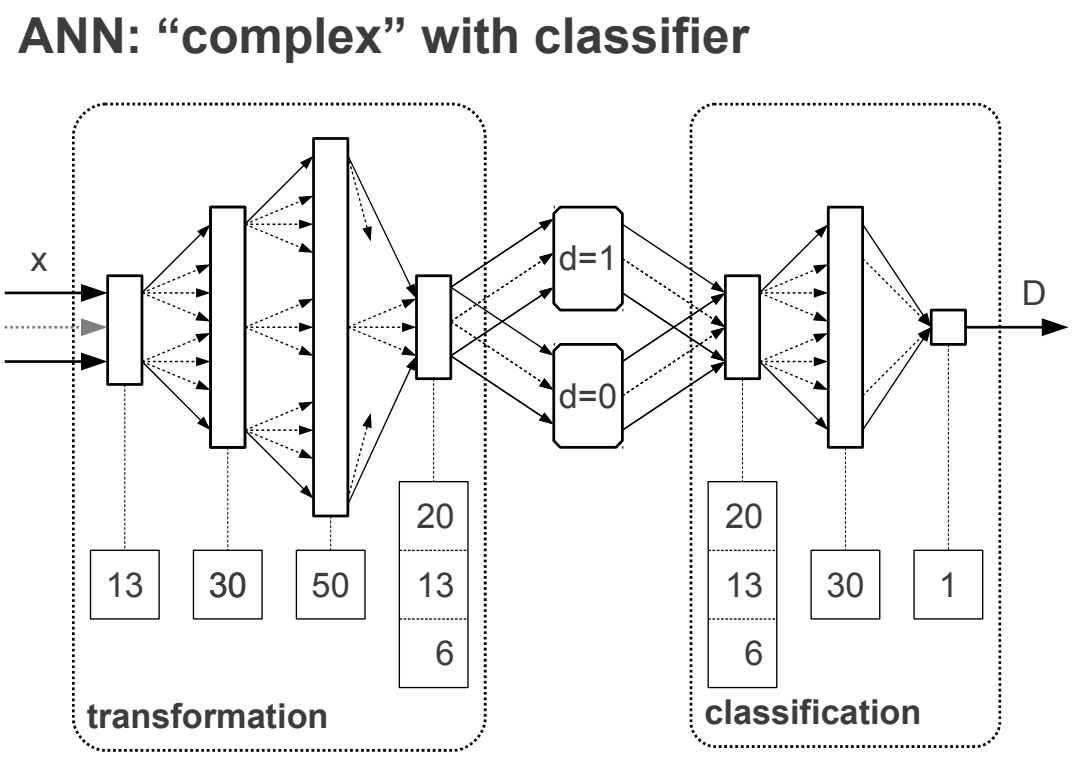
There are two basic possibilities how NN_D can be implemented. The first possibility assumes that this network is also trained (to learn the optimal distance measure) and can hence be defined as a "normal" neural network. This implementation offers more flexibility by making the entire ANN structure amenable to training, but might also cause problems with convergency.

The second possibility uses a predefined distance measure (i.e. the Euclidean distance) and specifically tailors this stage of the system to compute this distance measure. This way, the overall system becomes less flexible (it contains a static, non-trainable part) and should cause less problems during learning.

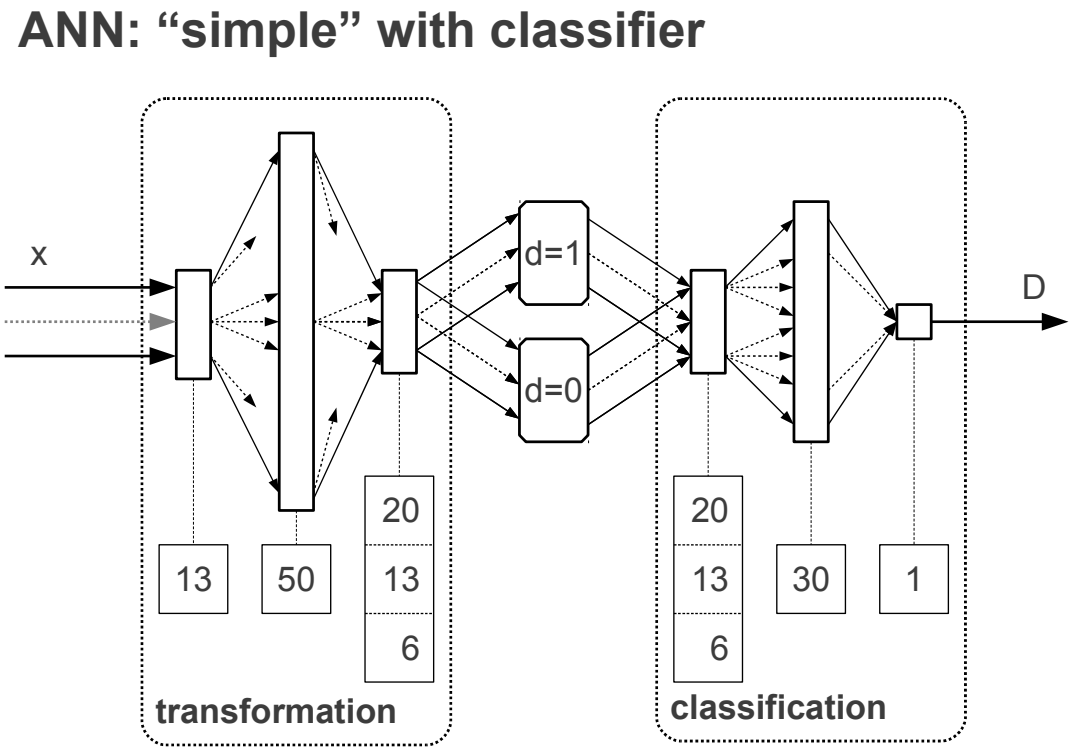
In this work, both of the above options are explored. The first alternative which includes NN_D in the training uses a simple structure with an input layer, an output layer and an internal layer with 30 neurons (effectively a 2-layer perceptron). This variant is referred to as the *classifier* configuration.

The second alternative which uses a predefined network implements the square Euclidean distance (the squared value was used because the NICO toolkit does not support the computation of $\sqrt{x}$). This implementation is termed *euclidean distance* or *euclid* configuration (please also confer to figures 4 and 5).

Network Topology Diagrams All the previously discussed network architecture variants are visualized in figure 4 and 5, which show the *classifier* and the *euclidean distance* configuration of the network in both the *complex* and the *simple* version. The boxes with $d = \ldots$ written in them symbolize the delays, the boxes with bare numbers in them indicate the number of neurons in the respective layer (6 is *lowerdim*, 13 is *samedim* and 20 is *higherdim*).

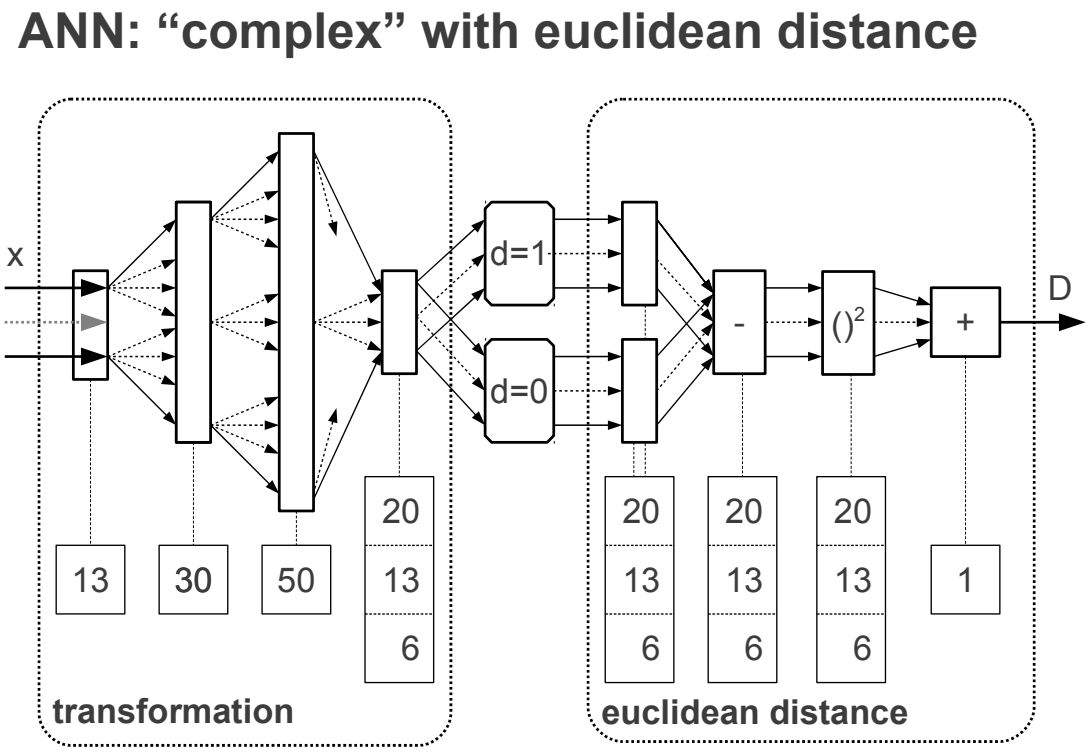


(a) Complex variant: transformation stage has totally three layers

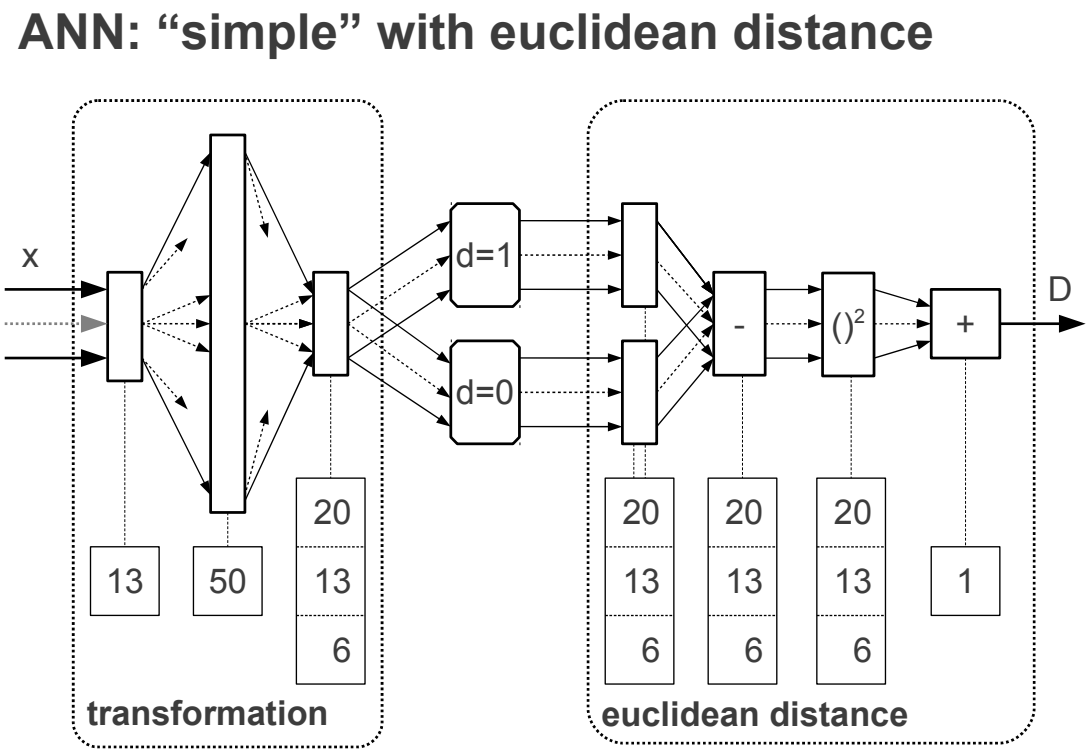


(b) Simple variant: transformation stage has totally two layers

Figure 4: ANN with transformation and **classifier** stage



(a) Complex variant: transformation stage has totally three layers



(b) Simple variant: transformation stage has totally two layers

Figure 5: ANN with transformation and **euclidean distance** computation stage

4.3 Training with Artificial Data

Motivation Before training and evaluating the ANN configurations defined in section 4.2 with real speech data (which are very complex), it makes sense to analyse their behavior with some artificially created (and therefore simpler) data. The complexity of the speech data is due to the fact that the undesired variance in the features comes from the speaker characteristics on one side and from measurement inaccuracy, noise and other distortions on the other side.

Let us now briefly look at how reasonable artificial data can be produced. Intuitively, different classes in a feature space can be modeled as distinct points in space (each location stand for a certain class). Moreover, scattering ("measurement uncertainty") can be introduced by using clusters around these defined locations instead of just one distinct point. Similarly, discrimination conflicts between the classes can be introduced by increasing the variance of the clusters to make them overlap to a certain degree.

To create these clusters, one can for instance use the multivariate Gaussian distribution. In this case, the mean vector of each cluster represents the desired information and the covariance matrix (which determines the shape and size of the cluster) models the undesired variance. To define the locations of the data clusters, some random vectors can be created within a finite volume. Assume the finite volume under consideration is $V_0 := [-L_0, +L_0]^k \subset \mathbb{R}^k$ where $L_0 \in \mathbb{R}$ is the maximum value allowed for each coordinate and $k \in \mathbb{N}$ is the dimensionality of the feature space. If N_C mean vectors need to be chosen, the average volume available for each cluster is approximately given by

$$V_C \propto \frac{(2 \cdot L_0)^k}{N_C} \tag{10}$$

For simplicity, V_C can be assumed to be of cubic shape (V_0 is partitioned into k-dimensional cubes). The average distance between two adjacent random vectors is in the same order of magnitude as the equivalent side length of the cube V_C

$$d_c \propto \left(\frac{V_C}{N_C}\right)^{\frac{1}{k}} = \frac{2 \cdot L_0}{(N_C)^{\frac{1}{k}}} \tag{11}$$

The overlap between the clusters can then be controlled by choosing the diagonal elements of each clusters covariance matrix Σ in the same order of magnitude as d_c (Note that the diagonal elements σ_{ii} of Σ determine the spread of the distribution along the coordinate axes; this is the easiest way to control the overlap). The σ_{ii} are chosen randomly in the range $[\alpha \cdot 0.75 \cdot L_0, \alpha \cdot 1.25 \cdot L_0]$ to provide both random variation and sufficiently large variance for all clusters. The parameter α tunes the amount of overlap (the larger α, the more overlap).

For this work, the following values are used

- $L_0 = 5$
- $k = 13$ (mimics the 13 MFCCs)

- $N_C = 39$ (mimics the 39 phonemes used)
- $\alpha \in \{0.25, 0.5, 1.0, 2.0\}$

Four different values are used for α to investigate cluster with different amount of scattering. For ease of reference, the different variance modes are termed *low* variance ($\alpha = 0.25$), *moderate* variance ($\alpha = 0.5$), *high* variance ($\alpha = 1.0$) and *extreme* variance ($\alpha = 2.0$).

Training and Results For each variance mode, independent training, validation and test data are created to train all the different ANN configuration discussed in section 4.2. The ANNs are defined and trained with the NICO toolkit (confer to [18]). The training, validation and test set consist of 70, 20 and 10 data files, each of which contains a sequence of 1755 vectors. During training, the net iterates over the sequences and learns to determine whether two subsequent vectors from the sequences are taken from the same cluster or not. If the two vectors belong to the same cluster, output 0 is expected; if they come from different clusters, output 1 is expected (as explained in section 3.2). The training is stopped when the error on the validation set has decreased to an optimal value. Then, the transformation network is extracted from the overall neural network and used to analyse the data given in the test set.

In a first step, the test sequences are fed into the overall network to obtain the actual output of the distance computation stage. The actual output sequences are compared to the expected output sequences (note that since the learning is never perfect, a distance value below 0.5 is interpreted as 0 and a value above 0.5 is interpreted as 1) and the overall discrimination error on the test set is calculated. The corresponding results are shown in table 4.

Note that the discrimination error is not equivalent to the recognition error. The discrimination error expresses how often the net has failed to determine whether two feature vectors belong to the same class or not, whereas the recognition error tells how often a speech recognizer fails to assign a single feature vector to the correct phoneme.

In a second step, the test data are transformed using the trained net NN_T and the pairwise Fisher distances as well as the global Fisher distance are computed from the transformed data. These results allow to determine whether and how much the discriminability (in terms of the used measure) has improved. Table 5 shows the global Fisher distances for all ANN configurations. The figures 6, 7 and 8 show colored maps of the pairwise Fisher distances computed from untransformed and transformed data.

Note that the same color map is used for all three Fisher distance maps to better visualize the changes (larger Fisher distance means brighter color). To do so, it was necessary to crop the scale for coloration (any Fisher distance larger than the defined maximum is simply painted in the brightest color).

Classifier Configuration						
Variance	complex			simple		
	higher	same	lower	higher	same	lower
Low	0.42 %	0.40 %	2.53 %	0.31 %	0.32 %	2.31 %
Moderate	1.58 %	1.64 %	4.25 %	1.93 %	2.10 %	5.13 %
High	4.62 %	5.01 %	9.31 %	4.80 %	6.07 %	10.27 %
Extreme	12.60 %	13.72 %	19.19 %	12.55 %	13.65 %	19.50 %
Euclidean Distance Configuration						
Variance	complex			simple		
	higher	same	lower	higher	same	lower
Low	0.54 %	1.29 %	6.66 %	0.33 %	1.36 %	6.51 %
Moderate	1.12 %	2.13 %	7.50 %	1.35 %	2.25 %	8.28 %
High	2.96 %	3.62 %	9.13 %	3.54 %	4.13 %	9.24 %
Extreme	11.12 %	10.63 %	14.87 %	11.09 %	10.85 %	14.25 %

Table 4: Discrimination error rate obtained from training with artificial data

Classifier Configuration							
Variance	original data	complex			simple		
		higher	same	lower	higher	same	lower
Low	12.76	13.34	13.00	6.78	16.18	17.08	7.44
Moderate	7.89	11.02	10.44	5.63	11.58	9.78	5.63
High	3.78	6.46	5.98	3.28	5.79	4.72	2.92
Extreme	1.51	2.58	2.34	1.46	2.44	2.12	1.41
Euclidean Distance Configuration							
Variance	original data	complex			simple		
		higher	same	lower	higher	same	lower
Low	12.76	40.43	30.79	14.07	39.61	28.80	13.22
Moderate	7.89	24.21	19.55	10.01	21.25	18.07	9.86
High	3.78	9.34	9.10	5.52	9.02	8.59	5.61
Extreme	1.51	3.42	3.48	2.57	3.30	3.34	2.67

Table 5: Global Fisher distance (obtained from training with artificial data)

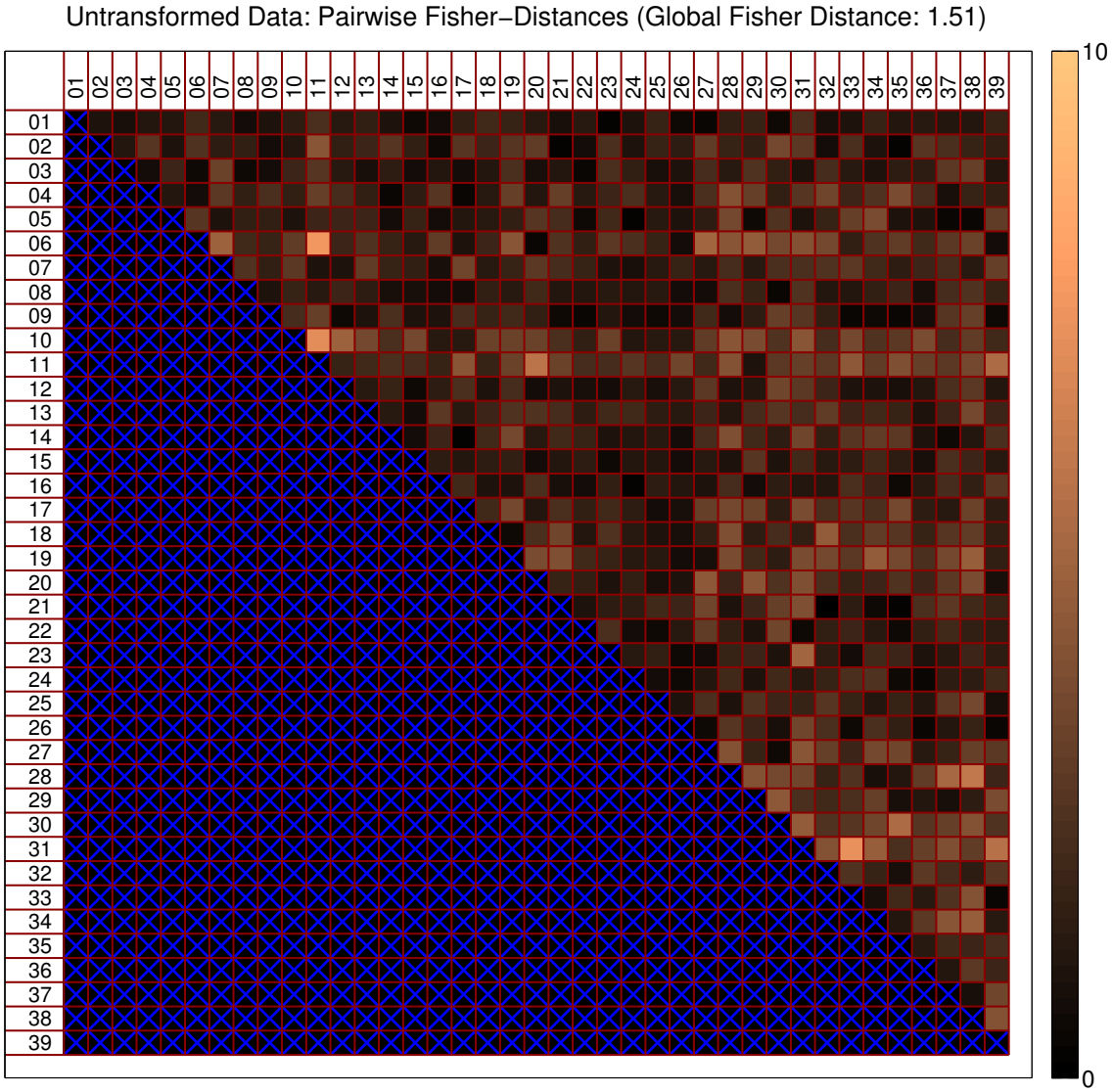


Figure 6: Pairwise Fisher distances obtained for untransformed artificial data with **extreme** variance

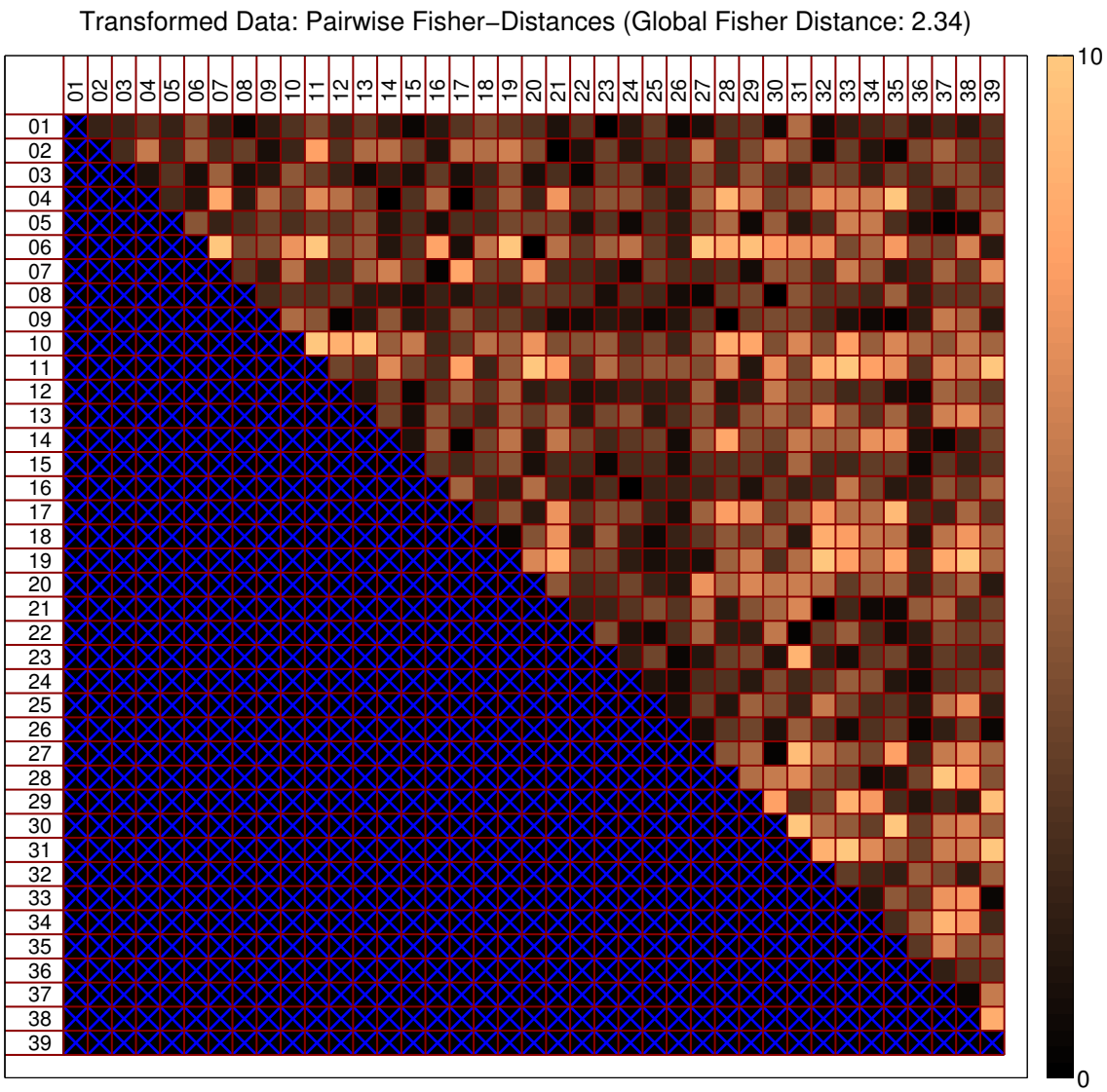


Figure 7: Pairwise Fisher distances obtained for transformed artificial data with **extreme** variance using the ANN configuration **classifier-complex-samedim** (NN_D is the *classifier* block, NN_T is the *complex* transformation net which maps to the *same* dimensionality)

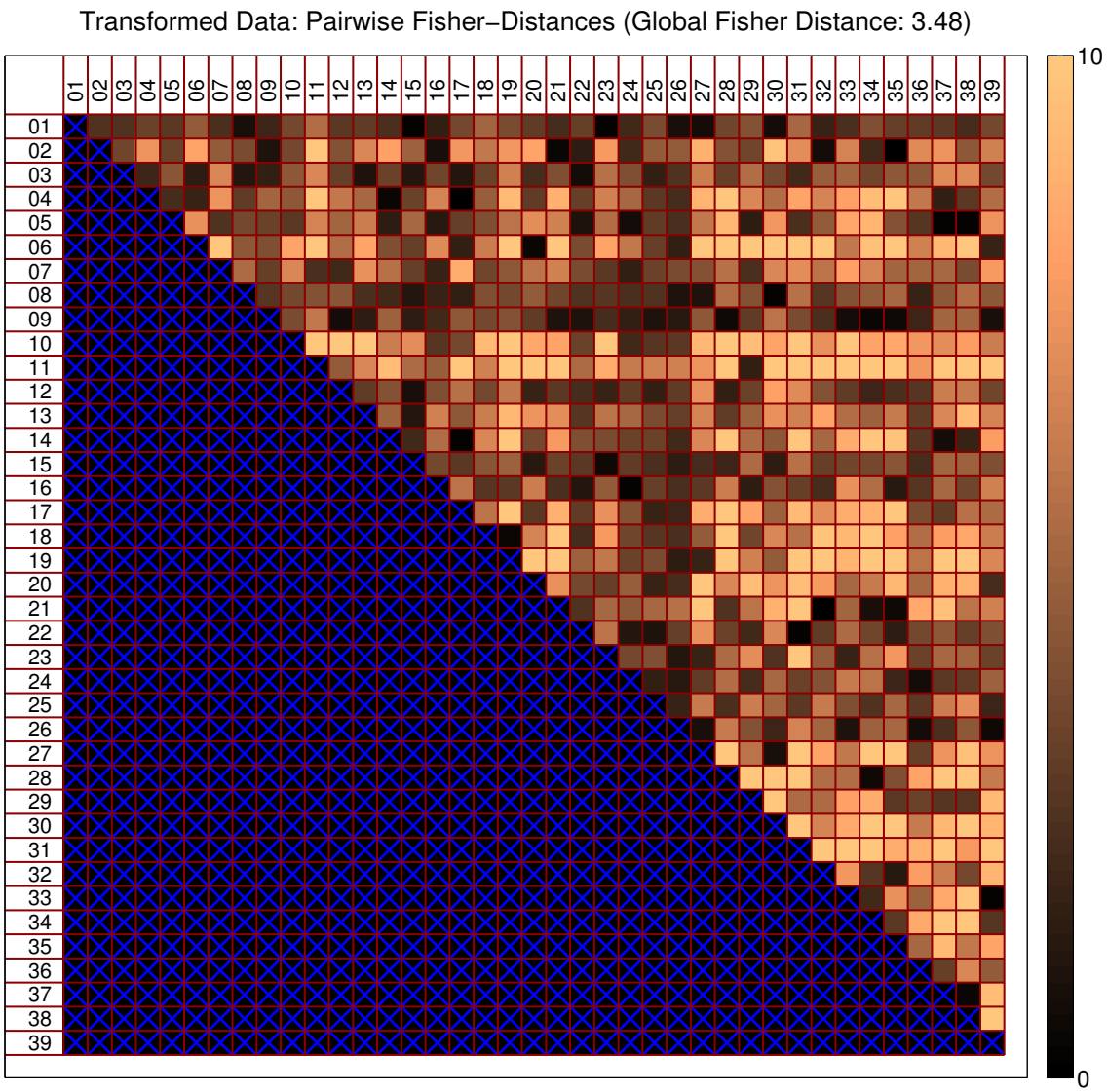


Figure 8: Pairwise Fisher distances obtained for transformed artificial data with **extreme** variance using the ANN configuration **euclid-complex-samedim** (NN_D is the *euclidean distance* block, NN_T is the *complex* transformation net which maps to the *same* dimensionality)

Discussion and Interpretation First of all, let us take a closer look at the discrimination error rates shown in table 4. As one would expect, the error is larger for datasets with more substantial variance because the data clusters overlap more. The error is on the same rather low level for transformation networks which map to the *same* or a *higher* dimensionality, but significantly larger when the transformation maps to a *lower* dimensionality. The discrimination performance is about the same for the *complex* and the *simple* configuration of the transformation network (however, the *simple* configuration seems to produce slightly less error in some cases). When comparing the results of the *classifier* and the *euclidean distance* stage, one can see that the *classifier* stage is superior for data of *low* and *moderate* variance whereas it is inferior for data of *high* and *extreme* variance.

Now, we look at how the discriminability has changed in terms of the global Fisher distance as shown in table 5. Obviously, the overall Fisher distance of the transformed data decreases for data sets with higher variance as one would expect (intuitively, data of larger variance cause more difficulties when learning the transformation and make it harder to reduce variance in general). As for the discrimination error, the variants which map the data to a *higher* or the *same* dimensionality perform equally well regarding the obtained Fisher distance, whereas the mapping to *lower* dimensionality performs worse. Furthermore, the *complex* and *simple* transformation networks produce comparable results.

However, the ANN configuration with a *classifier* or a *eudlidean distance* block behave quite differently. When comparing the Fisher distances of untransformed and transformed data, one can see that the *classifier* setup often only provides only a slight improvement (*higherdim*, *samedim*) or even diminishes the Fisher distance (*lowerdim*). The relative improvement is rather small for data with lower variance, but more substantial for data with higher variance.

The *euclidean distance* setup, on the other hand, manages to improve the Fisher distance for all data sets and all ANN configurations. Also, the improvements are much larger than for the *classifier* setup. Obviously, the use of a static distance computation block results in a transformation network which is superior to the one produced with a trainable distance computation block.

Lastly, let us investigate how the pairwise Fisher distances have changed. Specifically, the ANN configurations *classifier-complex-samedim* (confer to figure 7) and *euclid-complex-samedim* (confer to figure 8) are used to make this comparison for the case of data with *extreme* variance. The Fisher distances of the untransformed data are shown in figure 6.

It can be seen on the first sight that the color maps made for the transformed data are much brighter than for the untransformed data. As the same scale is used for coloration, this means that the discriminability has generally been improved. However, the increase is larger for the *euclidean distance* setup (the colors are even brighter than for the *classifier* setup). Finally, note that there are also many cluster pairs for which the painted fields in the map stay dark because nothing changes. This is probably due to larger overlap of the corresponding data clusters.

On the whole, the following insights can be gained from analyzing the training done with artificial data

- the proposed ANN training setup is capable of reducing variance in general (at least for artificial data obtained from multivariate Gaussian distributions)
- the *euclidean distance* configuration performs better than the *classifier* configuration in all cases. Apparently, the training is more successful when the task is restricted to discriminating between classes, but does not include finding an optimal distance measure at the same time.
- the *euclidean distance* configuration improves the global Fisher distance by a factor of 2-3 (the improvement diminishes with increasing variance of the untransformed data)
- transformation blocks which map data to a *lower* dimensionality are clearly inferior to those which map data to the *same* or a *higher* dimensionality
- *complex* and *simple* transformation networks produce comparable results

4.4 Training with Speech Data

Working with the TIMIT Database After training the ANNs with simplified artificial data, we can now dedicate our attention to the more complex speech data. The goal is to find out whether the ANN can also learn to reduce variance which is caused by speaker characteristics in the phonemes.

The MFCC features are computed using speech records from the TIMIT database (confer to section 3.1 and 4.1 for the detailed computation of the MFCC). The TIMIT corpus consists of English sentences uttered by both male and female speakers from 8 different dialect regions. The particular subset used in this work (taken from the folder `timitcore`) contains 16 male and 8 female speakers (please confer to table 6 for a detailed listing of the speakers). Each dialect region is represented with 2 male and 1 female speaker.

Phoneme segmentation information (boundaries and labels) is available for speech data. TIMIT uses totally 61 symbols for labeling, but usually only a reduced set of 39 symbols is used (confer to table 7 for the simplified phoneme dictionary and to [2] for further information).

male		female	
1	bpm0	1	dhc0
2	cmj0	2	elc0
3	dab0	3	jlm0
4	grt0	4	mgd0
5	jdh0	5	mld0
6	jln0	6	lnp0
7	jmp0	7	pas0
8	klt0	8	pkt0
9	lll0		
10	lnt0		
11	njm0		
12	pam0		
13	tas1		
14	tls0		
15	wbt0		
16	wew0		

Table 6: Listing of the different Speakers (TIMIT)

Label	Phoneme Group	Label	Phoneme Group
aa	aa, ao	k	k
ae	ae	m	m, em
ah	ah, ax, ax-h	n	n, en, nx
aw	aw	ng	ng, eng
axr	axr, er	ow	ow
ay	ay	oy	oy
b	b	p	p
ch	ch	pau	epi, q, bcl, dcl, gcl, kcl, pcl, tcl, pau
d	d		
dh	dh	r	r
dx	dx	s	s
eh	eh	sh	sh
el	el, l	t	t
ey	ey	th	th
f	f	uh	uh
g	g	uw	uw, ux
hh	hh, hv	v	v
ih	ih, ix	w	w
iy	iy	y	y
jh	jh	z	z, zh

Table 7: Simplified phoneme dictionary (taken from [2])

Set	Training		Validation		Test	
	male	fem.	male	fem.	male	fem.
1	1, 2, 3, 5, 6, 7, 9, 13	1, 2, 3, 4, 7, 8	4, 8, 11	6	10, 12, 14	5
2	1, 2, 3, 5, 6, 7, 9, 13	1, 2, 3, 4, 7, 8	10, 12, 14	5	4, 8, 11	6
3	2, 4, 6, 9, 10, 11, 12, 13	1, 3, 5, 6	1, 3, 5, 8	4, 7	7, 14, 15, 16	2, 8
4	2, 4, 6, 9, 10, 11, 12, 13	1, 3, 5, 6	7, 14, 15, 16	2, 8	1, 3, 5, 8	4, 7
5	4, 5, 7, 8, 13, 15	3, 4, 6, 7	6, 11, 12, 14	1, 2	2, 3, 9, 10	5, 8
6	4, 5, 7, 8, 13, 15	3, 4, 6, 7	2, 3, 9, 10	5, 8	6, 11, 12, 14	1, 2
7	1, 4, 5, 6, 7, 13, 14, 16	1, 2, 3, 5, 6, 7	8, 9, 11, 12	4	2, 3, 10, 15	8
8	1, 4, 5, 6, 7, 13, 14, 16	1, 2, 3, 5, 6, 7	2, 3, 10, 15	8	8, 9, 11, 12	4
9	3, 6, 7, 8, 9, 11, 15, 16	1, 2, 3, 4, 5, 8	2, 4, 10	7	5, 12, 13	6
10	3, 6, 7, 8, 9, 11, 15, 16	1, 2, 3, 4, 5, 8	5, 12, 13	6	2, 4, 10	7

Table 8: Speaker set partitions used for training (please consult table 6 for a listing of the speaker numbering scheme)

Training and Results In order to train the network to abstract from undesired speaker characteristics, the speech features need to be computed on phonemes taken from disjoint sets of speakers for training, validation and test. To allow cross-validation, 10 different set partitions were manually defined (confer to table 8). These partitions were arranged to contain instances of every phoneme from several different speakers. Note that the partitions of validation and test set are in fact interchangeable, but the amount of data used differs. The training, validation and test set consist of 70, 20 and 10 data files each of which contains a sequence of 3000 MFCC vectors. The training is performed in the same way as described in section 4.3: the net (in all different configurations) is trained to output 1 if two subsequent feature vectors stem from different phonemes and 0 otherwise. The training is stopped as soon as the error computed on the validation set is sufficiently low and the trained transformation network NN_T is extracted from the overall system for further investigation.

First of all, the discrimination error is computed by comparing the actual output sequences obtained for the test data to the expected output sequences. The results for the different speaker set partitions as well as the average value are presented in table 9.

In addition, the transformation block is used to obtain the transformed speech features. For all configurations, the global Fisher distance as well the average value are computed on all speaker sets. These results can be found in table 10.

Furthermore, the pairwise Fisher distances are visualized for the untransformed data and the transformed data obtained from two specific ANN configurations (*classifier-complex-samedim* and *euclid-complex-samedim*) in figure 9, 10 and 11 using color maps with the same scale. These maps are obtained from averaging the corresponding data over the different test speaker sets defined in table 8.

Discussion and Interpretation To start with, let us briefly investigate the results for the discrimination error shown in table 9. Seemingly, the achievable error rate is approximately the same for all ANN configurations and speaker sets. Although there is some variation among the different speakers sets, the average values are practically the same for all speaker sets. However, note that the error rates for the *euclid* configuration are slightly lower than for the *classifier* configuration.

Let us now move on to the global Fisher distance values listed in table 10. Obviously, the global Fisher distances vary quite much among the different speaker set partitions for both the untransformed and the transformed data, but this is probably due to the different characteristics of these specific speaker combinations (that is the reason why cross-validation is done after all). Furthermore, it seems that the different variants for dimensionality and complexity (within the *classifier* and *euclidean distance* setup) behave largely the same. It is more informative to look at how the values have changed after the transformation and how the results of the *classifier* and *euclidean distance* behave.

Except for one case, the *classifier* setup manages to improve the global Fisher

Classifier Configuration						
Set	complex			simple		
	higher	same	lower	higher	same	lower
1	25.4 %	24.6 %	24.8 %	24.9 %	24.6 %	25.0 %
2	23.5 %	23.7 %	23.5 %	23.8 %	23.5 %	24.4 %
3	24.7 %	25.7 %	25.3 %	24.4 %	24.7 %	25.7 %
4	22.3 %	21.4 %	21.8 %	21.3 %	22.4 %	22.1 %
5	23.1 %	22.8 %	23.7 %	22.9 %	23.4 %	23.8 %
6	24.2 %	24.1 %	25.1 %	23.4 %	24.0 %	24.3 %
7	24.5 %	25.7 %	24.7 %	24.8 %	22.5 %	24.6 %
8	23.2 %	21.7 %	23.7 %	22.4 %	22.4 %	23.4 %
9	22.7 %	22.5 %	25.1 %	21.1 %	21.1 %	21.8 %
10	24.7 %	25.5 %	24.7 %	25.0 %	24.3 %	25.2 %
Av.	23.8 %	23.4 %	23.8 %	23.5 %	24.3 %	24.0 %
Euclidean Distance Configuration						
Set	complex			simple		
	higher	same	lower	higher	same	lower
1	22.7 %	22.3 %	22.9 %	23.0 %	22.7 %	22.6 %
2	22.4 %	22.0 %	22.6 %	21.9 %	22.1 %	23.0 %
3	23.6 %	24.0 %	23.3 %	23.8 %	22.9 %	23.3 %
4	19.5 %	19.9 %	20.0 %	19.6 %	20.1 %	19.6 %
5	21.8 %	21.1 %	21.6 %	21.9 %	22.5 %	21.6 %
6	22.8 %	22.5 %	22.9 %	22.9 %	22.3 %	23.0 %
7	23.0 %	23.1 %	23.0 %	22.8 %	22.7 %	22.9 %
8	20.9 %	21.4 %	21.5 %	21.3 %	21.3 %	22.1 %
9	20.4 %	19.8 %	19.9 %	19.5 %	19.6 %	20.4 %
10	21.7 %	22.9 %	22.9 %	22.4 %	23.3 %	22.7 %
Av.	21.9 %	21.9 %	22.0 %	21.9 %	22.0 %	22.1 %

Table 9: Discrimination error rate obtained from training with speech data

Classifier Configuration							
Set	original data	complex			simple		
		higher	same	lower	higher	same	lower
1	0.68	0.69	0.95	0.76	0.94	0.98	0.85
2	0.78	0.97	1.11	0.97	1.16	1.06	0.86
3	0.36	0.72	0.85	0.78	0.80	0.83	0.72
4	0.87	1.29	1.30	1.21	1.36	1.35	1.24
5	0.72	1.17	1.13	1.16	1.23	1.18	1.09
6	0.34	0.89	0.89	0.76	0.93	0.84	0.77
7	0.76	1.10	0.93	1.05	1.07	1.16	0.91
8	0.65	1.06	1.19	1.03	1.18	1.19	1.03
9	0.84	1.16	1.03	0.93	1.25	1.36	1.07
10	0.83	0.89	0.76	0.89	1.05	1.03	0.97
Av.	0.68	0.99	1.02	0.96	1.10	1.10	0.95
Euclidean Distance Configuration							
Set	original data	complex			simple		
		higher	same	lower	higher	same	lower
1	0.68	0.99	1.10	0.97	1.13	1.09	1.02
2	0.78	1.18	1.19	1.17	1.30	1.21	1.22
3	0.36	0.92	0.95	0.92	0.76	0.74	0.85
4	0.87	1.64	1.66	1.64	1.60	1.65	1.65
5	0.72	1.34	1.40	1.41	1.41	1.37	1.42
6	0.34	1.04	1.02	1.06	1.03	0.91	1.02
7	0.76	1.25	1.26	1.24	1.23	1.25	1.25
8	0.65	1.38	1.33	1.28	1.36	1.29	1.30
9	0.84	1.45	1.44	1.39	1.54	1.45	1.50
10	0.83	1.39	1.26	1.27	1.18	1.26	1.21
Av.	0.68	1.26	1.26	1.24	1.26	1.22	1.25

Table 10: Global Fisher distance (obtained from training with speech data)

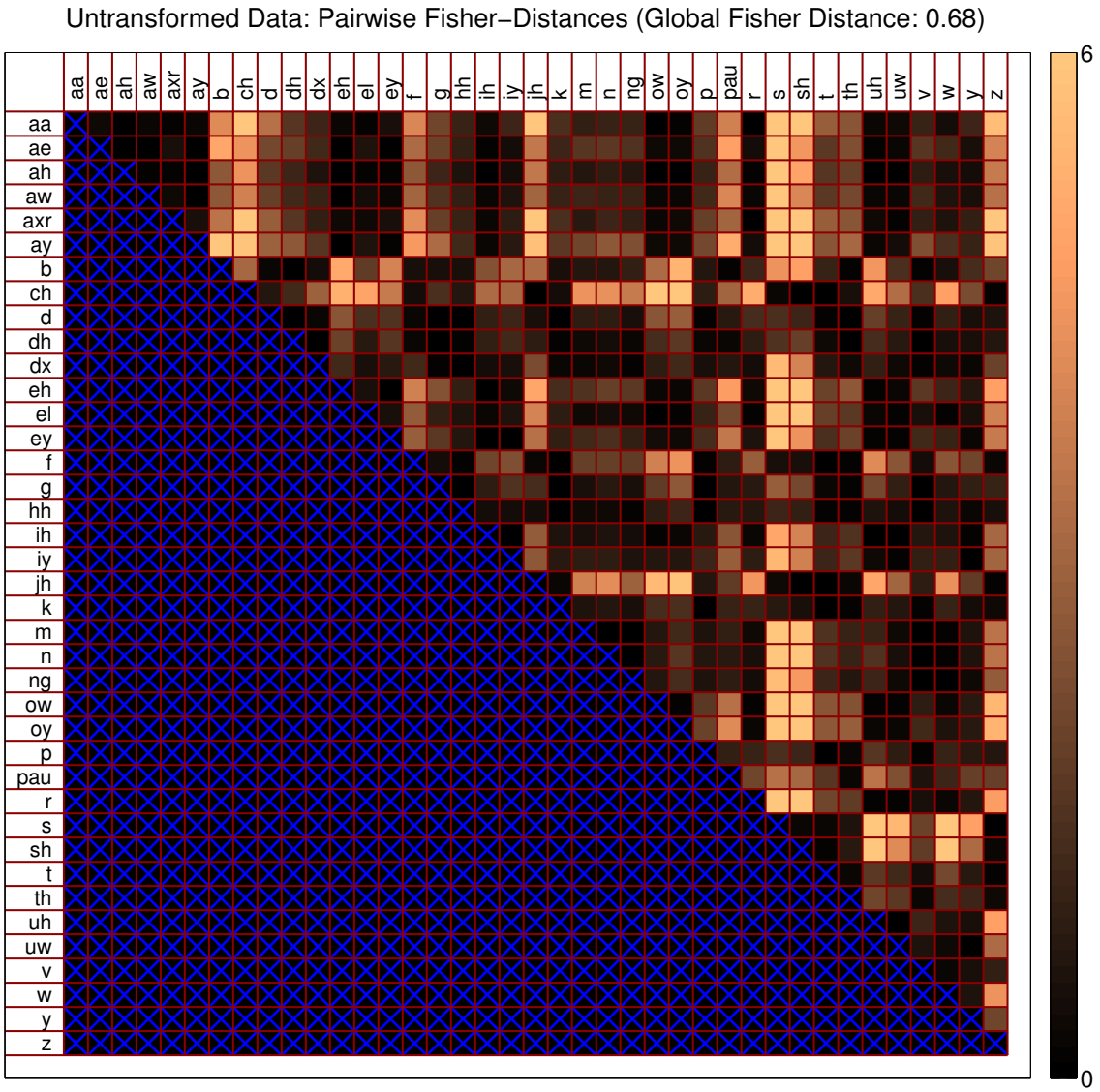


Figure 9: Pairwise Fisher distances obtained for untransformed speech data (averaged over different sets of test speakers)

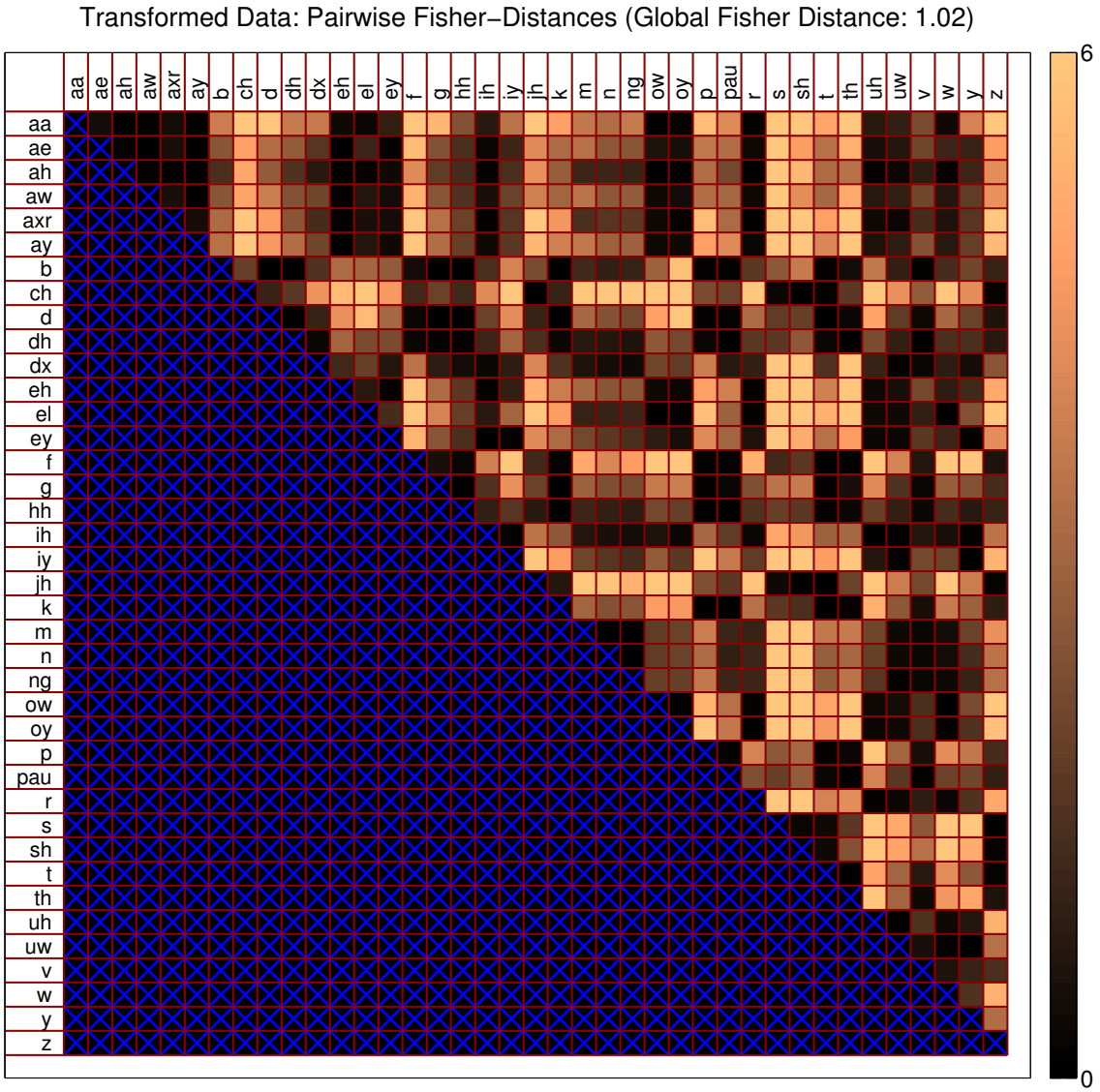


Figure 10: Pairwise Fisher distances obtained for transformed speech data using the ANN configuration **classifier-complex-samedim** (NN_D is the *classifier* block, NN_T is the *complex* transformation net which maps to the *same* dimensionality)

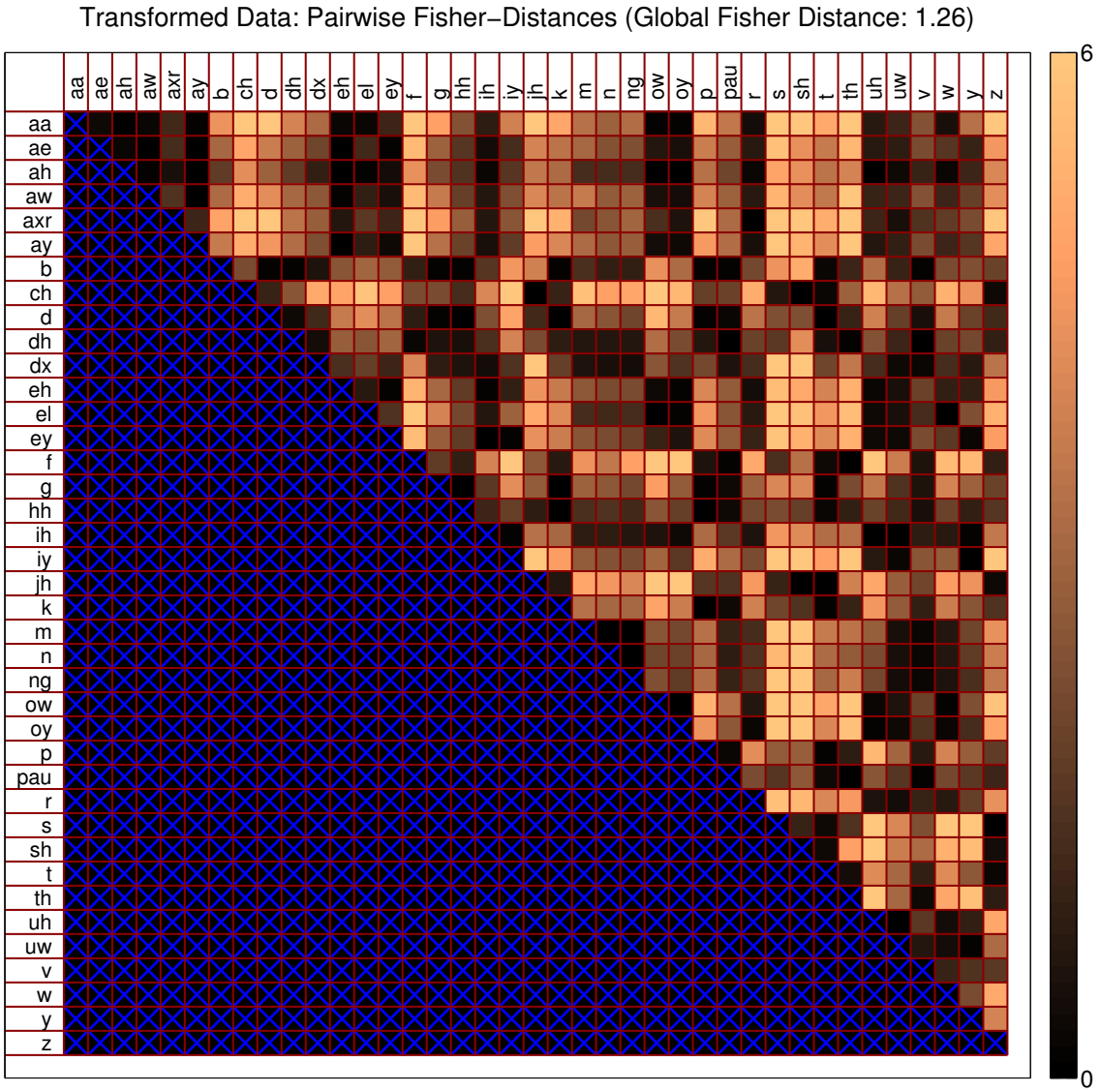


Figure 11: Pairwise Fisher distances obtained for transformed speech data using the ANN configuration **euclid-complex-samedim** (NN_D is the *euclidean distance* block, NN_T is the *complex* transformation net which maps to the *same* dimensionality)

distance. On the average, the value of 0.68 of the untransformed data is raised to approximately 1.0 to 1.05, which is an increase of 47-54%.

The *euclidean distance* setup manages to increase the global Fisher distance in all cases and also provides a larger improvement. The value after the transformation is in the range 1.24 to 1.26, which is an improvement of 82-85%.

Obviously both variants of the distance computation network can help to produce a transformation network which improves the discriminability of the speech features. However, the blocks NN_T obtained from training with a predefined NN_D perform better.

Finally, we take a short glimpse at the maps of the pairwise Fisher distances portrayed in figure 9, 10 and 11 (note that these maps were obtained from averaging over the different speaker sets).

The illustration for the untransformed data shows that most of the phoneme pairs are hard to be told apart (large dark areas). However, there are also quite a few cases which do not seem to be problematic (bright spots and stripes). The plots for the transformed data demonstrate what was already found in the above investigation: the general discriminability is improved (the corresponding Fisher distance maps are much brighter in color). However, there are also situations where the values stay the same. Fortunately, the phoneme pairs which already had a high pairwise Fisher distance remain on this good level. Unfortunately, there are also many phoneme pairs (mainly vowels) which stay constant on a bad level for both the networks obtained from the *classifier* and the *euclidean distance* configuration. However, the training with the *euclid* block still results in more improvement for many of these critical cases than the one with the *classifier* (some of the corresponding areas in the plot are brighter).

On the whole, we have learned the following from the analysis of the speech data

- the proposed ANN training setup obviously works for speech data
- both the training with *classifier* and *euclidean distance* blocks NN_D produces transformation networks NN_T which manage to increase the Fisher distance
- the *euclid* setup with an average improvement of 82-85% performs better than the *classifier* setup with an average improvement of 47-54%. This is probably due to the fact that the *euclid* setup restricts the learning task to discrimination whereas the *classifier* setup additionally tries to find an optimal distance measure. Apparently, the training is more successful for the simpler task.
- Both the dimensionality of the transformation (*higherdim*, *samedim* and *lowerdim* configuration) and the complexity of the transformation (*complex* and *simple* configuration) show no strong influence on the result of the training with a *euclidean distance* block (the average of the global Fisher distance over the speaker sets is practically the same for all variants).
- obviously, the achievable improvement is lower for speech data than for the artificially created data because vocal and phoneme characteristics are more complex than Gaussian distributions

5 Conclusion and Future Work

It could be verified that the proposed artificial neural network setup is indeed capable of learning a coordinate transformation which significantly improves the Fisher distance of the feature vectors.

For a proof of concept, data clusters were created using the multivariate Gaussian distribution. By teaching the overall neural network to decide whether two vectors belong to the same cluster, the transformation sub-block could learn the coordinate transformation. A closer investigation of the transformed data showed that the Fisher distance is improved by a factor of 2-3 for artificial data with a broad range of variances when the distance computation sub-block of used during training implements the (square) euclidean distance.

After verifying the functionality of training setup and the obtained transformation, the method was applied to MFCC features computed from the TIMIT database. Training, validation and test data for the neural network structure were taken from disjoint sets of speakers to optimally train the ANNs and investigate the effect of the learned transformation. The analysis revealed that even for speech data, a substantial increase of 82-85% can be achieved for the Fisher distance using the same euclidean distance block in the overall network. For the case of a trainable distance computation block, a smaller improvement of 47-54% was found.

Although the conducted experiments indicate that the phoneme discriminability is increased, there was unfortunately no time to integrate the obtained transformation network into a speech recognizer to determine the effect on the phoneme recognition rate. This implementation still needs to be done.

Moreover, more effort could be dedicated to the optimisation of the MFCCs prior to performing the transformation. The effects of filterbanks with different shapes and multitaping could not be looked at in more detail because the limited timeframe of a semester thesis did not allow to do so. It would certainly be interesting to investigate more sophisticated mel-filterbanks and other multitaper functions in detail.

Finally, the training setup could be adapted to include more speakers and therefore more instances of the different phonemes. It seems probable that given a larger amount of training data, further improvements can be achieved.

References

[1] Jahangir Alam, Patrick Kenny, and Douglas O'Shaughnessy. A Study of Low-Variance Multi-Taper Features for Distributed Speech Recognition. In *Advances in Nonlinear Speech Processing*, volume 7015 of *Lecture Notes in Computer Science*, pages 239–245. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-25020-0-31.

[2] M. Antal. Speaker Independent Phoneme Classification in Continuous Speech. *Studia Univ. Babes-Bolyai Informatica*, 49(2):55–64, 2004.

[3] A. Biem, S. Katagiri, E. McDermott, and Biing-Hwang Juang. An Application of Discriminative Feature Extraction to Filter-Bank-based Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 9(2):96–110, February 2001.

[4] S. Chakroborty, A. Roy, and G. Saha. Improved closed Set Text-Independent Speaker Identification by combining MFCC with Evidence from Flipped Filter Banks. *International Journal of Signal Processing*, 4(2):114–122, 2007.

[5] S. Chakroborty and G. Saha. Improved Text-Independent Speaker Identification using Fused MFCC & IMFCC Feature Sets based on Gaussian Filter. *International Journal of Signal Processing*, 5(1):11–19, 2009.

[6] M. Cutajar, E. Gatt, I. Grech, O. Casha, and J. Micallef. Neural Network Architectures for Speaker Independent Phoneme Recognition. In *7th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 90–94. IEEE, 2011.

[7] T. Kinnunen. Matlab Implementation of the SWCE Multitaper Method. [`http://cs.joensuu.fi/pages/tkinnu/webpage/`; accessed May 2012].

[8] T. Kinnunen, R. Saeidi, F. Sedlák, K.A. Lee, J. Sandberg, M. Hansson-Sandsten, and H. Li. Low-Variance Multitaper MFCC Features: a Case Study in Robust Speaker Verification. 2012.

[9] David Kriesel. Ein kleiner überblick über Neuronale Netze. *Download under* ***http://www.dkriesel.com/index.php***, 2008.

[10] Shang-Ming Lee, Shi-Hau Fang, Jeih weih Hung, and Lin-Shan Lee. Improved MFCC Feature Extraction by PCA-optimized Filter-Bank for Speech Recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding 2001 (ASRU '01)*, pages 49–52, 2001.

[11] B. Logan et al. Mel Frequency Cepstral Coefficients for Music Modeling. In *International Symposium on Music Information Retrieval*, volume 28, page 5, 2000.

[12] D.J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[13] S. Molau, M. Pitz, R. Schluter, and H. Ney. Computing Mel-Frequency Cepstral Coefficients on the Power Spectrum. In *IEEE International Converence on Acoustics, Speech and Signal Processing 2001 (ICASSP '01)*, volume 1, pages 73–76. IEEE, 2001.

[14] Heinrich Niemann. Lehrbuch zur Mustererkennung. *Download under* *http://www5.informatik.uni-erlangen.de/fileadmin/Persons/NiemannHeinrich/klassifikation-von-mustern/*, 2003.

[15] D. O'Shaughnessy. Interacting with Computers by Voice: Automatic Speech Recognition and Synthesis. *Proceedings of the IEEE*, 91(9):1272–1305, 2003.

[16] Beat Pfister and Tobias Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung (Springer-Lehrbuch)*. Springer Publishing Company, Incorporated, 2008.

[17] M.D. Skowronski and J.G. Harris. Increased MFCC Filter Bandwidth for Noise-robust Phoneme Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing 2002 (ICASSP '02)*, volume 1, pages I–801. IEEE, 2002.

[18] Nico Stromul. The NICO Toolkit. [`http://www.nico.nikkostrom.com/#FAQ_`; accessed May 2012].

[19] R. Vergin, D. O'Shaughnessy, and A. Farhat. Generalized Mel Frequency Cepstral Coefficients for Large-Vocabulary Speaker-Independent Continuous-Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 7(5):525–532, September 1999.

[20] R. Vergin, D. O'Shaughnessy, and V. Gupta. Compensated Mel Frequency Cepstrum Coefficients. In *IEEE International Conference on Acoustics, Speech and Signal Processing 1996 (ICASSP '96)*, volume 1, pages 323–326. IEEE, 1996.

A Task Description (Original Handout)

Institut für
Technische Informatik und
Kommunikationsnetze

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo

(SA-2012-04)

Aufgabenstellung für die Semesterarbeit

von

Herrn Andreas Kettner

Betreuer: Dr. B. Pfister, ETZ D97.6
S. Hoffmann, ETZ D97.5

Ausgabe: 20. Februar 2012
Abgabe: 1. Juni 2012

Optimale Sprachmerkmale zur Unterscheidung von Lauten

Einleitung

Im Zusammenhang mit der Spracherkennung muss detektiert werden, welche Laute in einem Sprachsignal nacheinander vorkommen. Im Sprachsignal gibt es jedoch nicht nur Information über die Laute, sondern auch über die Stimme der Person, von welcher das Sprachsignal stammt. Zudem wird das Sprachsignal durch Sprechweise, Dialekt, Emotionen, ev. Gesundheit etc. geprägt (vergl. [1], Kap. 2.1). Alle diese zusätzlichen Informationen sind jedoch für die Spracherkennung nicht nützlich, im Gegenteil, sie stören und erschweren die Spracherkennung.

Die wichtigste Voraussetzung für einen guten Spracherkenner ist deshalb, Merkmale aus dem Sprachsignal zu ermitteln, die zwar von den Lauten abhängig sind, aber durch die weiteren Informationen im Sprachsignal möglichst wenig beeinflusst werden. Anhand dieser Merkmale können dann die Laute statistisch beschrieben oder unterschieden werden (v.a. mittels Hidden-Markov-Modellen und neuronalen Netzen).

Merkmale für die Spracherkennung

Für die Spracherkennung werden heute hauptsächlich sogenannte Mel Frequency Cepstral Coefficients (MFCC) und davon abgeleitete Grössen eingesetzt (siehe [1], Kap. 10.7).

MFCC sind deshalb als Sprachmerkmale für die Spracherkennung geeignet, weil sie das Spektrum des Sprachsignals in einer Art beschreiben, welche unabhängig von der Phase und von der Lautstärke des Sprachsignals ist. Zudem weisen die MFCC eine geringe Sensitivität auf gegenüber Veränderungen der Sprachgrundfrequenz, zumindest für tiefe Stimmen. Für Frauen- und Kinderstimmen ist der Einfluss der Grundfrequenz auf die MFCC grösser.

Was jedoch im Zusammenhang mit der Spracherkennung im Allgemeinen am meisten stört ist der Einfluss der stimmlichen Eigenschaften, die für jede Person mehr oder weniger stark verschieden sind. Diese stimmlichen Eigenschaften bewirken eine grosse Varianz der Sprachmerkmale. Das bedeutet, dass die Merkmale eines Lautes, der von verschiedenen Personen gesprochen worden ist, sich viel mehr unterscheiden, als wenn der gleiche Laut von einer Person mehrmals gesprochen wird. In der Spracherkennung wirkt sich dies so aus, dass ein Spracherkenner, der für eine bestimmte Person trainiert worden ist (sprecherabhängiges Training), für Sprache von dieser Person eine signifikant höhere Erkennungsrate erreicht als von anderen Personen. Wird der Spracherkenner mit Sprachsignalen von vielen Personen trainiert (sprecherunabhängiges Training), dann ist die Erkennungsrate auch für am Training nicht beteiligte Personen zufriedenstellend, aber doch deutlich tiefer als dies bei sprecherabhängigem Training der Fall wäre.

Problemstellung

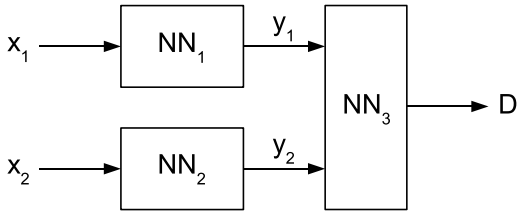
Da die stimmlichen Eigenschaften, die von Person zu Person verschieden sind, einen grossen Einfluss auf die in der Spracherkennung gebräuchlichen Sprachmerkmale haben, also die Varianz der Merkmale vergrössern, stellt sich die Frage, wie die Varianz verkleinert werden könnte. Grundsätzlich gibt es zwei Wege: entweder man sucht geeignetere Merkmale oder man transformiert die MFCC so, dass der Sprechereinfluss reduziert wird.

In dieser Semesterarbeit soll der zweite Weg eingeschlagen werden. Dabei wird von der Hypothese ausgegangen, dass es für Merkmalsvektoren $\mathbf{x}$ eine Transformation Ψ gibt, welche den Sprechereinfluss auf die Merkmalsvektoren reduziert. Transformierte Vektoren $\mathbf{y} = \Psi\{\mathbf{x}\}$ sollen also weniger stark durch die Sprecher geprägt sein. Gleichzeitig soll jedoch die Abhängigkeit vom Laut erhalten bleiben, sonst wäre das Merkmal nicht mehr für die Spracherkennung brauchbar.

Die Güte eines Merkmals zur Unterscheidung von Lauten kann wie folgt beurteilt werden: Ein Merkmal ist umso besser, je weniger es für denselben Laut variiert und je mehr es für verschiedene Laute variiert. Diese Güte lässt sich mit der Fisher-Distanz[1] ausdrücken (vergl. auch Übung 11 zur Vorlesung Sprachverarbeitung).

Es stellt sich nun die Frage, wie man die Transformation Ψ erhält. Die Idee ist, diese Transformation zu schätzen, indem man ein neuronales Netz so trainiert, dass es eine Näherung $\tilde{\Psi}$ dieser Transformation lernt. Ein neuronales Netz kann eine Transformation lernen (oder wenigstens einen Näherung davon), wenn eine genügend repräsentative Menge von Ein-/Ausgangspaaren dieser Transformation vorhanden sind (siehe [1], Anhang A.5).

[1] Die Fisher-Distanz ist ein Mass für die Verschiedenheit von zwei Wahrscheinlichkeitsverteilungen. Sie ist definiert als $d_f = (m_0 - m_1)^2 \,/\, (\sigma_0^2 + \sigma_1^2)$, wobei m_0 und m_1 die Mittelwerte und σ_1^2 und σ_0^2 die Varianzen der beiden Verteilungen sind.



Figur 1: System zum Trainieren des neuronalen Netzes für die Transformation Ψ

Dies ist hier jedoch nicht der Fall. Für das Training muss deshalb das neuronale Netz in System von Netzen eingebaut werden, wie es in Figur 1 dargestellt ist.

In diesem System ist das zu trainierende neuronale Netz zweimal vorhanden, nämlich NN_1 und NN_2. Sie sind identisch. Zusätzlich ist NN_3 vorhanden, das eine Distanz D zwischen den transformierten Vektoren $\mathbf{y}_1$ und $\mathbf{y}_2$ ermittelt. Für zwei Eingangsvektoren $\mathbf{x}_1$ und $\mathbf{x}_2$ soll nun die Distanz zwischen den transformierten Vektoren $\mathbf{y}_1$ und $\mathbf{y}_2$ klein sei, nämlich d_0, und zwar dann, wenn die Vektoren $\mathbf{x}_1$ und $\mathbf{x}_2$ aus Sprachabschnitten mit demselben Laut stammen. Andernfalls soll die Distanz gross, also d_1 sein.

Das aus drei neuronalen Netzen bestehende System ist auch wieder ein neuronales Netz. Es wird mit NN_S bezeichnet. Für NN_S können nun Ein-/Ausgangspaare wie folgt angegeben werden: Aus Sprachsignalen von vielen Sprechern kann man Beispiele für die Vektoren $\mathbf{x}_1$ und $\mathbf{x}_2$ auswählen und angeben, ob der zugehörige Zielwert d_0 oder d_1 ist.

Grundsätzlich kann somit das Netz NN_S trainiert werden.[2] Wie die Teile NN_1, NN_2 und NN_3 zu konfigurieren sind (wieviele Schichten und Neuronen) und wie das Training durchzuführen ist (unter der Bedingung, dass NN_1 und NN_2 identisch sind), ist jedoch noch zu untersuchen.

Interessant wird schliesslich sein, ob die mit NN_1 transformierten Merkmalsvektoren wirklich besser sind, also beispielsweise eine grössere Fisher-Distanz aufweisen als die untransformierten. Am meisten interessiert selbstverständlich um wieviel die Spracherkennungsrate durch die Transformation verbessert werden kann.

Vorgehen

Für diese Semesterarbeit wird das folgende Vorgehen empfohlen:

1. Arbeiten Sie sich zuerst in die Thematik ein. Machen Sie sich mit der Merkmalsextraktion (Ermitteln der MFCC aus dem Sprachsignal) und der Beurteilung von Sprachmerkmalen anhand der Fisher-Distanz (siehe SPV-Übung 11) vertraut.

2. Stellen Sie analog zur SPV-Übung 11 ein System zusammen, mit dem Sie die Güte von Sprachmerkmalen testen und mit den Standardmerkmalen vergleichen können. Für dieses Testsystem sollen Sprachsignale aus der TIMIT-Datenbank (englische Sprache; wird zur Verfügung gestellt) verwendet werden und es sind alle Laute

[2]Das Netz NN_S hat grundsätzlich dieselbe Struktur wie die neuronalen Netze in [2]. NN_S ist jedoch nicht voll verbunden.

einzubeziehen. Ergänzen Sie das System so, dass es aufgrund von Testdaten eine Lautverwechslungsmatrix ermitteln und ausgeben kann.

3. Optimieren Sie die Parameter der Berechnung der MFCC so, dass Lautunterscheidung (ermittelt mit dem obigen Testsystem) optimal wird. Überlegen Sie zuerst, welche Parameter zu variieren sind und besprechen Sie Ihre Überlegungen mit den Betreuern. Führen Sie dann die Optimierung durch.

4. Arbeiten Sie sich in die neuronalen Netze und in das Trainings-Tool NICO (siehe [3]) ein. Legen Sie die Konfiguration der Netze NN_1 , NN_2 und NN_3 fest und überlegen Sie, wie das gesamte System NN_s mit Nico trainiert werden kann.

5. Stellen Sie aus der TIMIT-Datenbank Trainings-, Validierungs und Testdaten zusammen. Trainieren und testen Sie NN_s. Testen Sie auch das erhaltene NN_1 mit erstellten Testsystem (siehe Punkt 2).

6. Fakultativ: Wenden Sie die verbesserten Merkmale (Punkt 3) und die Merkmalstransformation in einem HMM-basierten Spracherkenner an und ermitteln Sie, ob und um wieviel sich die Erkennungsrate erhöht hat.

Die ausgeführten Arbeiten und die erhaltenen Resultate sind in einem Bericht zu dokumentieren (siehe dazu [4]), der in elektronischer Form (als PDF-Datei) abzugeben ist. Zusätzlich sind im Rahmen eines Kolloquiums zwei Präsentationen vorgesehen: etwa zwei Wochen nach Beginn soll der Arbeitsplan und am Ende der Arbeit die Resultate vorgestellt werden. Die Termine werden später bekannt gegeben.

Literaturverzeichnis

[1] B. Pfister und T. Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer Verlag (ISBN: 978-3-540-75909-6), 2008.

[2] M. Gerber, T. Kaufmann, and B. Pfister. Perceptron-based class verification. In *Proceedings of NOLISP (ISCA Workshop on non linear speech processing)*, Paris, May 2007.

[3] KTH Stockholm. The NICO Toolkit for Artificial Neural Networks, 1996. (online manual `http://nico.nikkostrom.com/doc/`).

[4] B. Pfister. *Richtlinien für das Verfassen des Berichtes zu einer Semester- oder Diplomarbeit*. Institut TIK, ETH Zürich, Februar 2009.
(http://www.tik.ee.ethz.ch/spr/SADA/richtlinien_bericht.pdf).

[5] B. Pfister. *Hinweise für die Präsentation der Semester- oder Diplomarbeit*. Institut TIK, ETH Zürich, März 2004.
(http://www.tik.ee.ethz.ch/spr/SADA/hinweise_praesentation.pdf).

Zürich, den 20. Februar 2012