Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Disertační práce

k získání akademického titulu doktor

v oboru **Kybernetika**

Využití prostoro-časové struktury příznakových vektorů pro adaptaci neuronových sítí

Ing. Jan Trmal

Školitel: Doc. Ing. Müller Luděk, Ph.D.
Katedra kybernetiky

Plzeň, 2011

University of West Bohemia

Faculty of Applied Sciences

DOCTORAL THESIS

submitted in partial fulfillment of the requirements
for the degree Doctor of Philosophy

in the field of

Cybernetics

SPATIO-TEMPORAL STRUCTURE OF FEATURE VECTORS IN NEURAL NETWORK ADAPTATION

Ing. Jan Trmal

Advisor: Doc. Ing. Müller Luděk, Ph.D.
Department of Cybernetics

Pilsen, 2011

Abstract

This doctoral thesis aims at research in the field of neural networks adaptation and in the field of speaker adaptive training, with special attention to the application of both in the field of automatic speech recognition.

Both these technologies, i.e. adaptation and speaker adaptive training are often used in the area of speech recognition in the context of GMM/HMM modeling framework. In that context, they pose one possible approach to improving recognition accuracy, often at a cost of an insignificant increase of computational complexity. The crucial assumptions of both these techniques, i.e. of the speaker adaptation and of the speaker adaptive training, are realistic and can be ensured relatively easily. Therefore, it is desirable to have similar techniques developed even for hybrid (i.e. non-GMM/HMM) speech recognition systems.

The goal of this thesis was to develop such a method and to experimentally evaluate its influence on the accuracy of the speech recognition system.

Keywords: acoustic modeling, neural networks, adaptation, speech recognition, speaker adaptive training

Abstrakt

Tato práce se zabývá metodikou adaptace neuronových sítí a na řečníku adaptivním trénováním neuronových sítí pro systémy automatického rozpoznávání řeči.

Obě tyto technologie, tedy jak adaptace, tak na řečníku adaptivní trénování jsou v oboru rozpoznávání řeči často využívány v rámci GMM/HMM modelovacího frameworku. Zde představují jednu z dalších přístupů k zlepšování přesnosti rozpoznávání, často za cenu pouze zanedbatelného navýšení výpočetních nároků. Zásadní předpoklad těchto dvou technik, tedy znalost identity řečníka jak během trénování, tak i během rozpoznávání je poměrně realistický a poměrně snadno zajistitelný. Je tedy žádoucí, aby byly vyvinuty ekvivalentní techniky i pro hybridní systémy rozpoznávání řeči.

Cílem této práce je vyvinout a otestovat metodu adaptace a metodu na řečníku adaptivního tréningu a experimentálně ohodnotit její vliv na přesnost rozpoznávače.

Klíčová slova: akustické modelování, neuronové sítě, adaptace, rozpoznávání řeči, na řečníku adaptivní trénování

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně, s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne Jan Trmal

Acknowledgements

First of all, I wish to express my thanks to my advisor, Doc. Ing. Müller Luděk, Ph.D., for all the support, patience and consultations he gave me and to the head of my supervisory department, Prof. Ing. Josef Psutka CSc., for tolerance and for alleviating me from my other duties during the time I was working on this thesis.

It is appropriate that I record my heartfelt thanks to my friends who have, at all times, encouraged me to carry this project through to a successful conclusion. Thank you one and all.

The financial assistance of the Ministry of Education, Youth and Sport of the Czech Republic under the projects LC536 (*Integrated Center for Natural Language Processing*) and project 2C06020 (*Elimination of the Language Barriers Faced by the Handicapped Watchers of the Czech Television*) is fully acknowledged. Moreover, the access to the MetaCentrum computing facilities provided under the project LM2010005 (*Projects of Large Infrastructure for Research, Development, and Innovations*) funded by the Ministry of Education, Youth, and Sports of the Czech Republic is appreciated.

Contents

List of Figures

List of Tables

List of Abbreviations

1CTX	Single Context Neural Network, 83
ANN	Artificial Neural Network, 22
BNKS	Bottleneck Features, 10
BPTT	Backpropagation through time, 26
BUT	Brno University of Technology, 83
CMLLR	Constrained Maximum Likelihood Linear Regression, 47
CMN	Cepstral Mean Normalization, 42
CSR	Continuous Speech Recognition System, 78
CVN	Cepstral Variance Normalization, 42
DARPA	Defense Advanced Research Projects Agency, 77
DCT	Discrete Cosine Transform, 7
FMLLR	Feature-level MLLR, 47
GMM	Gaussian Mixture Model, 13
GPU	Graphic Processing Unit, 3
HATS	Hidden Activations TRAPS, 10
HMM	Hidden Markov Models, 5
HPC	High Performance Computing, 3
ICA	Independent Component Analysis, 58
IIR	Infinite Impulse Response, 42
iRPROP	Improved Resilient Propagation, 35
IVR	Interactive Voice Response, 76
LCRC	Left Context – Right Context, 83

LDA	Linear Discriminant Analysis, 58
LSTM	Long short term memory, 26
LTSP	Long Temporal Spectral Patterns, 9
LVCSR	Large Vocabulary Continuous Speech Recognition, 5
MFC	Mel-frequency Coefficients, 7
MFCC	Mel-frequency Cepstral Coefficients, 6
ML	Maximum Likelihood, 51
MLP	Multi-layer Perceptron, 22
MSE	Mean Square Error, 35
OSHM	One Step Hessian Manipulation, 53
PCA	Principal Component Analysis, 11
PCA	Principal Component Analysis, 58
PCM	Pulse-code modulation, 77
PHN	Parallel Hidden Network, 54
RPROP	Resilient Propagation, 35
SD	Speaker Dependent, 42
SI	Speaker Independent, 42
SRILM	The SRI Language Modeling Toolkit, 77
SVD	Singular Value Decomposition, 58
TRAPS	Temporal Patterns, 8
UWB	University of West Bohemia, 83
XENT	Cross Entropy, 35

1 Introduction

Adaptation is a profound process. It means you figure out how to thrive in the world.

Charlie Kaufman – Adaptation

The most prevalent approach to speech recognition (and especially to the large vocabulary speech recognition) is via the Hidden Markov Models (HMM) framework. The HMM framework allows its user to capture the underlying temporal structure of speech in a conceptually simple way.

The use of HMMs dates back to late 60's and early 70's of the 20th century([59]). The use of HMMs is tied to Gaussian Mixture Models (GMM). Nowadays, these two approaches form a virtual pair so firmly that these terms are used almost interchangeable.

Despite our knowledge about speech, speech production process, human physiology, etc., the ASR is still regarded as an opened (unsolved) problem. The deficiency of the current paradigm manifests acridly especially when comparing the performance of the ASR systems to the performance of human beings. The crucial question is, what the root cause of this deficiency is. One of the commonly held opinions says that the cause lies in the inability of the current paradigm to exploit fully the complete information available in the training data, mainly because of the fundamental assumptions about the speech recognition problem being implicitly made when employing the HMM/GMM approach. Therefore, new methods and algorithms should be worked on, which will relax the inherent limitations of the current approach.

Because of the modularity and the statistical foundations of ASR, the interaction between the individual components of the ASR system is done by means of providing probabilities of "interesting" phenomena. The consequence is that it is relatively easy to substitute the possibly suboptimal subsystem by an improved subsystem. An improvement can be achieved either by better engineering work or by using a different paradigm for modeling and computing the probabilities of the phenomena of interest. One of the promising approaches is to use the Artificial Neural Networks (ANNs) in the module of acoustic modeling. This is referred to as a hybrid approach or as a connectionist approach. Historically, neural networks have been successfully used in dealing with many complicated problems. The advantage of the neural networks is that only a limited knowledge about the problem structure is required, an awareness of the problem difficulty is sufficient. As with other data-driven approaches, the structure is determined automatically during the training process.

During the 50 years of the use of the HMM/GMM speech recognition paradigm, a tremendous scientific and engineering effort has been put into development, extending,

improvement and polishing of both the theory and the systems of speech recognition, based on these approaches. It is interesting not only from the scientific perspective but also from the economical perspective to research how many of the advanced techniques developed primarily for the HMM/GMM framework can be applied in the hybrid approach.

One of the interesting and certainly beneficial aspects that has not been paid sufficient attention to yet, is the possibility of adaptation of neural networks. In the broadest sense, adaptation means an adjustment of the coefficients (i.e. behavior) of the neural network in such way that improves the performance when the operating conditions change. In the context of speech recognition, adaptation is usually used to improve the performance of the ASR system, when some additional information is obtained. It can be information about the speaker, the environment, the recording channel and so on. The aim of this work is to develop an adaptation technique usable for these situations.

2 Scope and Goals of the Thesis

As has been said in the introduction, the adaptation is an advanced technique that helps to bridge the performance drop between speaker specific and speaker independent acoustic models. The adaptation process is relatively easy in the context of HMM/GMM framework, because models (or more precisely models' parameters) of the individual word units (triphones) are largely decoupled. Because of this decoupling, i.e. relative independence, only a limited portion of parameters is manipulated, when a single speech unit is being adapted. In the context of ANNs, the parameters of the individual units are not isolated. Instead, the knowledge about the units and their relationship is coded deep into the structure and the weights of the neural network. Therefore, a large portion of the weights must be adjusted even if only one speech unit is to be adapted. This is a serious issue when the amount of data is not sufficient (which is often the case in the field of ASR).

The approach this work adopts is to exploit the inner frequency-temporal structure of the input feature vectors to devise an adaptation method. Because of the underlying structure, the number of free variables can be reduced significantly during the adaptation process.

2.1 Goals of the Thesis

1. To develop a baseline hybrid ASR system to allow further research in the area.

2. To develop and optimize an MLP-ANN training software suitable for use in the environment of grid-computing. The software should use existing HPC libraries optimized by hardware vendor and make use of GPU devices, when available.

3. Using the available literature to describe the current approaches to ANN adaptation, identify the strong and weak points of the individual methods.

4. To develop a supervised MLP-ANN adaptation technique suitable for use on large and very large networks. Current approaches exhibit poor behavior or do not work at all.

5. To verify the proposed adaptation technique on publicly available speech corpora of sufficient sizes. At least one of the used speech corpora should be in the English language to allow an objective comparison with the state-of-the-art systems.

6. To develop a two-pass unsupervised adaptation technique suitable for use on large and very large neural networks and to verify the proposed adaptation technique on publicly available speech corpora.

3 Continuous Speech Recognition

The modern LVCSR systems are based on statistical approach almost exclusively. The approach most systems use is based on *Hidden Markov Models* (HMM). The block diagram of such system is depicted on the Fig. 3.1.

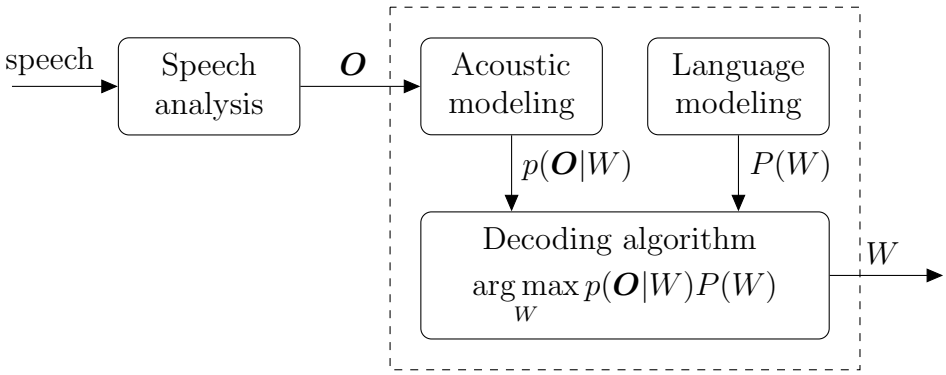


Figure 3.1: A block diagram of the statistical approach to speech recognition.

The block diagram contains several modules. In the module of *Speech analysis*, the speech signal is transformed into a series of observations $\boldsymbol{O} = \boldsymbol{o}_0, \boldsymbol{o}_1, \ldots$. The observation is usually represented by a *feature vector*. The assumption is that the observation sequence carries as much information needed for recognizing the speech as possible. The module of *Acoustic modeling* converts the series of observation into $p(\boldsymbol{O}|W)$, i.e. into likelihoods of the sequence of the observation given the word sequence W. The module of *Language modeling* evaluates the a priori probability $P(W)$ of the given word sequence W. The task of the *Decoding algorithm* is then to find the most probable word sequence $W^\star$ given the $p(\boldsymbol{O}|W)$ and $P(W)$.

More detailed description of the individual modules follows.

3.1 Speech Signal Analysis

The human speech contains an immense amount of information about the speaker, his/her emotional state, the environment the speaker is in, etc. It is impractical to use all the information for speech recognition. The task of speech signal analysis is to convert the acoustic signal (sampled at sampling frequency f, usually 8 kHz $\leq f \leq$ 44.1 kHz) into a series of observation vectors containing as much information about the utterance as possible while reducing the amount of the information unimportant for the speech recognition task.

Because the speech signal is non-stationary (i.e. the parameters of the system (vocal tract) generating the speech change through time), short-time analysis is usually applied. The assumption is that the parameters of vocal tract change only slowly and therefore the speech can be treated as a stationary signal for small time spans. Usually the time span in which the signal is treated as stationary is about 15 to 40 ms. To ensure sufficient resolution in time, these signal segments are gathered every 10 to 15 ms and therefore the segments overlay each other. The process of conversion the speech signal into a series of observation vectors $\boldsymbol{O}$ is called the *feature extraction*.

Feature extraction cannot create any new information; it just removes information not required for the given task. It is quite obvious that different information is needed for speech recognition tasks and different for the other tasks (such as speaker verification). There are three basic objectives:

- reduction of the feature vector dimensionality
- concentration and filtering of information
- coding of information

In an ideal case, all the three objectives are accomplished. However, in the real world, the reduction of dimensionality results necessarily in loss of information; it is impossible to filter out the unnecessary information without losing the useful information. Sometimes, the encoding of information even increases the dimensionality, and in most cases, it is impossible to know what the optimal representation of information is for the given classifier. So rather than striving for a complete fulfillment of these objectives, the goal is to find their optimal trade-off.

In the last few years, the features obtained using short-time frequency analysis have started to be complemented by features using longer temporal contexts to enable the classifier not only to work in the frequency domain but also take the temporal structure of the feature series into account. This new paradigm has brought additional computational, storage and data demands. Sadly, these demands could not have been fulfilled earlier, as the computation performance and storage capacities of the computers had not been sufficient.

Moreover, besides the usual model-driven approach, where during the feature extraction the parameters of a given model are to be determined (thus the name *parametrization* process), there is a tendency to use advanced machine learning methods to determine both the model structure and its parameters during the training process from the data (data-driven approach). The resulting model is then used to generate features having specific properties (probabilistic features, discriminative features, etc.). This is obviously much more computationally demanding. However, it relaxes the sometimes strong assumptions about the signal made commonly when using the model-driven approach.

3.1.1 Mel-frequency Cepstral Coefficients

Mel-frequency Cepstral Coefficients (MFCC) parametrization method is one of the most commonly used methods today. This method respects the laws of critical bands of hearing

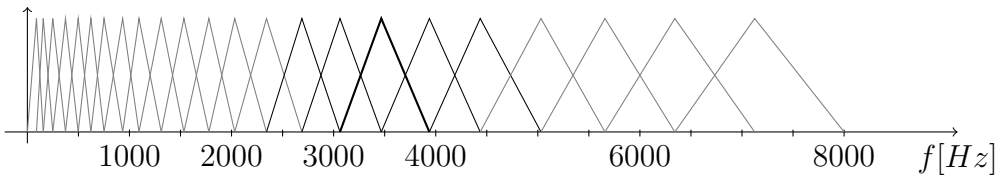


Figure 3.2: An example of placement of mel-filters in frequency

and the masking phenomenon. The relationship between subjectively perceived pitch (measured in units called *mels*) and (objective) frequency is described by means of the *mel-scale* using the following transform

$$f_{\text{mel}} = 2595 \log_{10} \left(1 + \frac{f_{\text{Hz}}}{700}\right). \tag{3.1}$$

The MFCC parametrization approach uses the transform from the Eq. (3.1) to place triangle-shape bandpass filters equidistantly in the mel-scale (and therefore non-linearly in the frequency). See the Fig. 3.2 for an example.

The log-energies of M these bandpass filters are called *Mel-frequency Coefficients* and are either used without any further processing or are decorelated using the discrete cosine transform (DCT) to obtain the *cepstral coefficients*.

The DCT transform is described as follows:

$$c(j) = \sum_{i=1} M \log_{10} \left[y(i)\right] \cos \left[\frac{\pi j}{M}(i - 0.5)\right], \qquad \text{for } j = 0, 1, \ldots, N, \tag{3.2}$$

where N is the number of target cepstral coefficients to compute. Usually, it holds $N < M$.

The cepstral coefficients paradigm is motivated by its favorable properties in relation to the recording channel influence. In the time domain, the channel influence is described as a convolution of the original signal and the impulse response of the channel. In frequency domain, the same effect can be modeled by multiplication of the spectrum of the signal and the spectrum of the impulse response of the channel. When using the cepstral coefficients, the channel influence is represented only as addition of the cepstrum of the channel to the cepstrum of the original signal. The cepstrum concept expresses only the approach to isolating the channel influence and its application is not limited to mel-frequencies filter banks.

3.1.2 Delta and Acceleration Coefficients

In general, the delta and the acceleration coefficients[1] capture the change of the individual features in feature vector through time. These changes (or trajectories) have a direct relation to the temporal structure of speech. In principle, the delta coefficients are

[1]The delta and the acceleration coefficients are often referred to as *dynamical coefficients*.

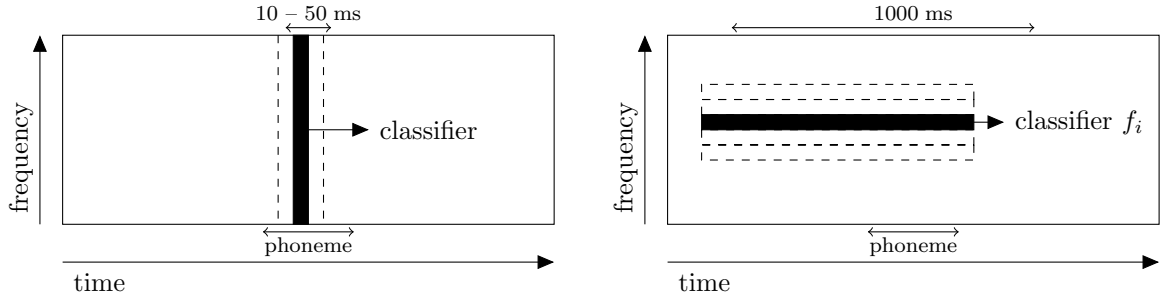


Figure 3.3: Comparison between the conventional (left) and the TRAP (right) approach, redrawn from [104].

computed using a numerical differentiation. Usually, the following formula (M-point secant method, $M = 2K + 1$) is used([124])

$$\Delta_t = \frac{\sum_{k=1}^{K} k(x_{t+k} - x_{t-k})}{2\sum_{k=1}^{K} k^2}, \tag{3.3}$$

where Δ_t is the delta feature computed at time t from the corresponding static features $\{x_{t-k}, \ldots, x_{t+k}\}$, k is the number of points from which the estimate is computed. Usually, $2 \leq K \leq 4$.

The acceleration coefficients are computed using the same equation, i.e. Eq. (3.3), however instead of the original static coefficients, the delta coefficients are used in place of the input features $\{x_{t-k}, \ldots, x_{t+k}\}$.

Since computation of Δ_t using the Eq. (3.3) relies on up to K points both in the past and the future, some care must be paid at the beginning and at the end of the input stream of features. Usually, the first or last feature vector is replicated as needed to fill the analysis window.

The resulting delta and acceleration coefficients are merged with the original feature vectors, increasing the original number of features twice or three times.

3.1.3 Temporal Patterns (TRAPS)

The Temporal Patterns (TRAPS) were introduced in [104]. The key motivation was the fact that the contemporary methods often perform speech signal analysis over a short time window owing to the non-stationary characteristics of the speech. Therefore, albeit these short-term parametrization techniques successfully capture frequency characteristics of the speech segments, they mostly fail to capture the temporal characteristics. The fact that the temporal characteristics carry a complementary information beneficial for speech recognition is strongly hinted by the positive influence of dynamic coefficients, mean and variance normalization techniques or RASTA([48]) on recognizer accuracy. Without going into specific details, all these methods modify or augment the original feature vectors using statistics computed on a larger time span.

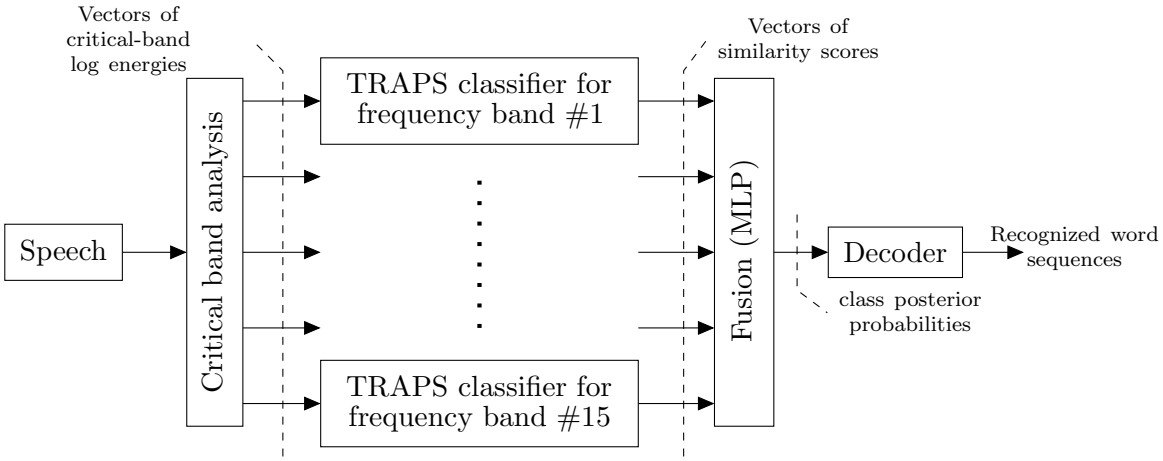


Figure 3.4: TRAP classification process, redrawn from [104].

The original TRAPS technique is based on short term mel-frequency critical bands analysis. However, compared to the conventional parametrization techniques, where the feature vectors are formed from the log-energies of all the critical bands at one time, the TRAPS parametrization technique is more elaborate. First of all, for each critical band, an auxiliary feature vector is created. The auxiliary feature vector of the i-th critical band captures the temporal trajectory of the log-energy values of the given critical band. The captured trajectory is relatively long and represents up to 1 second time span. See the Fig. 3.3. Then, a local (i.e. frequency band specific) phoneme classifier f_i is used to obtain a vector of target phone similarity scores. The phone similarity score vectors for all the critical bands are then fused together using a previously trained neural network. See the Fig. 3.4 for the complete scheme of the decision process.

The same work ([104]) experimented with different choices of the phoneme classifier used for production of the similarity scores. First, a simple correlation analysis is used to compare the trajectory with template phone trajectories. The correlation coefficients are then used as similarity scores. This technique is called *Mean TRAPS*, because of the way the phone templates are obtained. The templates are obtained by computing the mean of all possible phone realizations found in the training data. The other approach, called *Neural TRAPS* uses a neural network in the place of a probability estimator. The outputs of the neural network are estimates of the posterior probability of the individual phone classes.

3.1.4 Long Temporal Spectral Patterns (LTSP)

Although the TRAPS often bring a substantial improvement, the spread of the technique was hampered by a relatively high complexity of the approach. For example, in a situation where a mel-filterbank with 23 critical bands is used, the Neural TRAPS require 24 neural networks to be trained, which is a time consuming process. Moreover, as has been noted in [19], the multi-stage approach fails to model the spectral correlations between

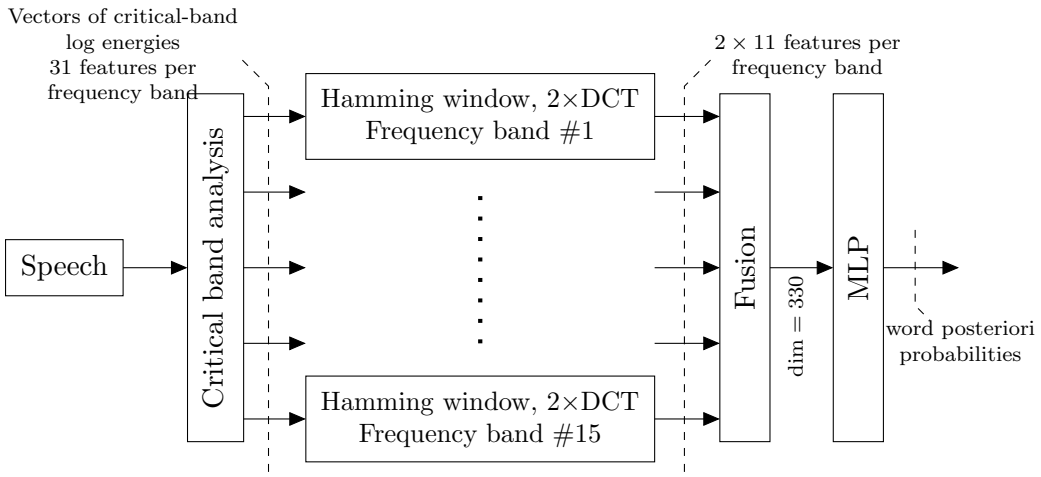


Figure 3.5: LTSP classification process, redrawn from [99].

the individual bands.

The author of [99] simplified the process significantly. With the process being only one-stage, the previously mentioned drawbacks have been alleviated. After performing the log-energy mel filterbank analysis, the auxiliary feature vector elements representing trajectories of the associated critical band are extracted in the same way as during the TRAPS analysis. These vectors/trajectories are then individually weighted using the Hamming window and then the number of coefficients in each trajectory is reduced using the DCT. Moreover, the author explicitly separates the "past" and "future" halves, using the current value of the critical band in both contexts. Then, all the individual sub-vectors obtained in the previous step are joined together and processed as one (long) feature vector. See the Fig. 3.5 for illustration of the process.

3.1.5 Hidden Activation TRAPS (HATS)

The concept of HATS builds on the Neural TRAPS technique. After performing a series of experiments the authors of the paper [19] concluded that the merger ANN should be trained on the activation values of the hidden layer instead of on the activation values of the output layer. Hence the name *Hidden Activations TRAPS*. The experimental system using the HATS and devised by the authors of the paper outperformed systematically the system using the original TRAPS setup.

3.1.6 Bottleneck Features (BNKS)

As the experiments with HATS suggested, the activation values from the hidden layer might be more suitable for further processing in the LVCSR. However, the HATS feature vector has a quite large dimension to use it directly in the conventional LVCSR system. Therefore, a dimensionality reduction method must be applied. The authors of [42, 44, 43] realized that these two stages can actually be performed together, using a neural

network of a special structure. They used a four layer MLP-ANN trained to give the estimates of phoneme posterior probabilities. Moreover, the second hidden layer had a really small number of neurons.

This setup is beneficial for several reasons. First, the hidden layer activation values can be used as they are, without any explicit dimensionality reduction transform. During the process of the training of the MLP-ANN, the MLP-ANN itself has learned the dimensionality reduction transform that minimizes the training criterion value. Second, because the training criterion led to inherently discriminative training, the hidden activation values (named *bottleneck features* by the authors) have a remarkable property of being discriminative, i.e. they are bearers of the information needed for discriminating between the target classes. This can be contrasted to the DCT or PCA methods, where the dimensionality reduction criterion guarantees only a minimal reconstruction error.

Moreover, according to the findings of the authors, the bottleneck features are complementary to the commonly used cepstral features and the fusion or combination of both types of features often leads to a significant improvement of the recognition accuracy.

3.2 Acoustic Modeling

The task of the Acoustic Modeling module is to provide an estimation of the likelihood $p(W|\boldsymbol{O})$ for an arbitrary sequence of observation $\boldsymbol{O}$ and an arbitrary sequence of words W. For this task, the HMMs are used almost exclusively.

The HMM is a model of a stochastic process (or stochastic automaton) ([87]) that in discrete time instances generates two tied time-aligned sequences of random variables. The first sequence is the sequence of observation $\boldsymbol{O} = \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots$, the second sequence is the sequence of states of the Markov model between which the process transitioned generating the sequence $\boldsymbol{O}$. The sequence of the states corresponds directly to the recognized utterance. The speech recognition using HMM can be formalized as a task to determine the most probable sequence of states, between which the automaton transitioned, given the fact that the observation sequence was $\boldsymbol{O}$.

For the task of the speech modeling, the left-to-right hidden Markov models are used prevalently. This class of models is suitable for modeling of stochastic processes, the hidden state of which evolves through time. As the observation vectors arrive, the automaton starts in the first state and either makes a transition gradually from states with lower index into states with higher index or stays in the same state. The automaton ends in the last state when the last observation vector arrives.

The actual HMM topology depends on the intended use. In the cases of small vocabulary isolated words recognition, one model can be created for each word. The number of states can differ from model to model([6]). However, a fixed number of states was also used([88]), with almost negligible loss of accuracy.

In the field of the LVCSR (thousands and more words), however, the previous approach is unfeasible. Instead, sub-word units are used. Such units can be syllables, phonemes or even smaller units. The most common choice is a special case of context-dependent phoneme units called triphone units (or triphones). The triphone units are able to model

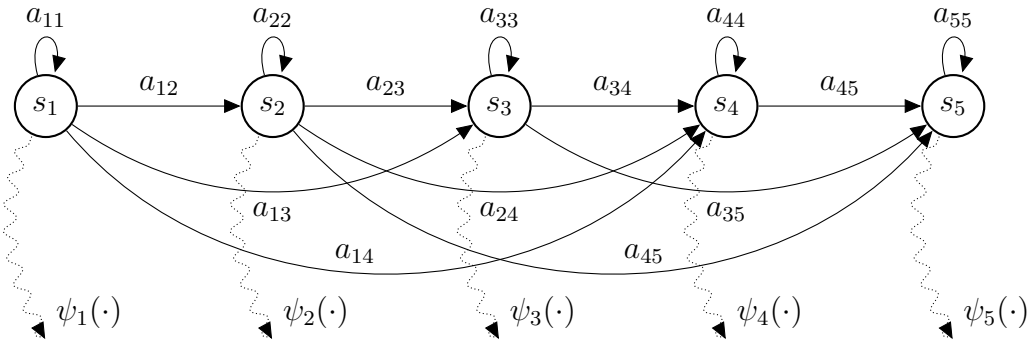


Figure 3.6: An example of a 5-state HMM of a word

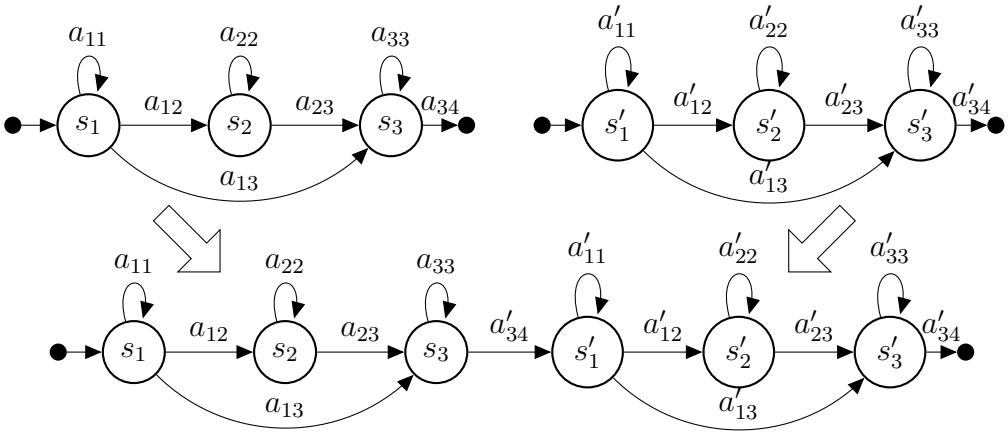


Figure 3.7: An example of linking of two triphones.

the inter-word and intra-word coarticulation context.

When using subword units, the word models are created by linking the subword units. Each subword unit has several emitting states and two non-emitting states. The non-emitting states are then used to link the subword-units together. An example of such process is depicted on the Fig. 3.7.

There are two classes of parameters in HMM. The first set is called *transition probabilities*. Its elements a_{ij} define the probability of transition from state i to state j. The transition probabilities are often represented as a (possibly sparse) matrix $\boldsymbol{A}$, $\boldsymbol{A} = |a|_{ij}$. The second class of parameters is referred to as *observation probability* functions. Nowadays, the observation probability function associated with state i is usually represented as a continuous probability density function $\psi_i(\boldsymbol{o}|\boldsymbol{\lambda})$, where $\boldsymbol{o}$ is the observation vector and $\boldsymbol{\lambda}$ is the vector of parameters of the function. The parameters of the function are determined during the training phase; the structure of the function is usually chosen a priori. There are two common choices of the architecture – Gaussian Mixture Models and Neural Network Densities Functions.

3.2.1 Gaussian Mixture Model (GMM)

In GMMs, the structure of the function ψ is given as

$$\psi(\boldsymbol{o}|\boldsymbol{\lambda}) = \sum_{i=1}^{M} \gamma_i p_i(\boldsymbol{o}|\boldsymbol{\lambda_i}), \tag{3.4}$$

where $p_i(\cdot)$, $i = 1, \ldots, M$ are density functions of individual mixtures, $\boldsymbol{\lambda_i}$ is the portion of $\boldsymbol{\lambda}$ that parametrizes $p_i(\cdot)$ and γ_i is the weight of the i-th mixture. Each mixture is then modeled as an N-dimensional normal probability distribution

$$p_i(\boldsymbol{o}) = \frac{1}{(2\pi)^{\frac{N}{2}}\sqrt{\det \boldsymbol{C}_i}} \exp\left[-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu}_i)^{\mathrm{T}}\boldsymbol{C}_i^{-1}(\boldsymbol{o}-\boldsymbol{\mu}_i)\right], \tag{3.5}$$

where $\boldsymbol{C}_i$ is the $N \times N$ covariance matrix, $\boldsymbol{\mu}_i$ is an N-dimensional vector of mean values and $\boldsymbol{o}$ is the observation vector. For the weights γ_i, $i = 1, \ldots, M$ holds

$$\sum_{i=1}^{M} \gamma_i = 1. \tag{3.6}$$

The parameters of individual mixture are $\boldsymbol{\lambda}_i = (\boldsymbol{C}_i, \boldsymbol{\mu}_i, \gamma_i)$ and the set of parameters $\boldsymbol{\lambda}$ is then $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N)$.

It can be proved that the GMMs are universal approximators (see [72]). The optimal number of mixtures, however, must be determined experimentally. Usually, for technical reasons, several assumptions is being made when using the GMMs. First, the matrices $\boldsymbol{C}_i$ are usually assumed to be diagonal and, second, the number of mixtures is often chosen to be a power of two.

3.2.2 Neural Network Densities Functions

Instead of training the GMM (which is prevalent nowadays), a neural network can be trained [113, 14]). The neural networks are universal approximators as well ([52, 24]). Unfortunately, this fact is not helpful in determining the optimal topology of the neural network.

The neural network can be trained in a way, which guarantees that asymptotically, the outputs of the network will be approximations of $p(\boldsymbol{w}|\boldsymbol{o})$ with $\boldsymbol{o}$ being the input observation and $\boldsymbol{w}$ being the vector of probabilities of all admissible (known to the system) elementary word or subword units (syllables, demi-syllables, phonemes, alophones, ...).

Using the Bayessian formula,

$$p(\boldsymbol{w}|\boldsymbol{o}) = \frac{p(\boldsymbol{o}|\boldsymbol{w})P(\boldsymbol{w})}{P(\boldsymbol{o})} \tag{3.7}$$

the likelihoods $p(\boldsymbol{o}|\boldsymbol{w})$ can be computed as

$$\frac{p(\boldsymbol{w}|\boldsymbol{o})P(\boldsymbol{o})}{P(\boldsymbol{w})} = p(\boldsymbol{o}|\boldsymbol{w}), \tag{3.8}$$

where the quantities $P(\boldsymbol{w})$ can be computed from the training data (one should be aware that $P(\boldsymbol{w})$ is different from the language model probabilities $P(W)$, because $\boldsymbol{w}$ is the set of the basic units, not necessarily the words from vocabulary) and $P(\boldsymbol{o})$ is a constant term that can be neglected.

3.3 Language Modeling

The module of Language Modeling contains a list of all words (or more generally units, since the recognizer can recognize not just words but also other units, e.g. phonemes) that are "known" to the speech recognition system together with the associated pronunciation. The pronunciation is represented as a sequence of the elementary (basic) units that have to be linked together to form the respective word. Each word has one or possibly more pronunciation alternatives.

Moreover, some words share the same pronunciation. Given the pronunciation, the decoding process must be able to select the correct word. The decision is based on the fact that different words have different probabilities in different contexts. The task of language modeling module is for any given sequence of words (phonemes) to provide the estimate of $P(W)$ for the decoding algorithm.

In practice, the volume of training data is always limited. There are four corner cases that can occur under this circumstance:

1. Not all words have been seen in the training data.
2. Some words haven't been seen enough times to compute the statistics reliably.
3. Not all word sequences have been seen in the training data.
4. Some word sequences haven't been seen enough times to compute the statistics reliably.

The language modeling module (called usually simply the *language model*) should be able to handle all these cases gracefully. While the cases 2 – 4 can be alleviated by approximating the probability of the problematic word or word sequence, the case 1 is more complex. A word that has not been seen during preparation of a language model is referred to as an *Out-of-Vocabulary* (OOV) word. There certainly are methods of language modeling that allow approximation of the probability of an unseen word and to detect the fact that a previously unseen word has been encountered during speech recognition. However, the problem lies in the process of conversion of the acoustic observations to the phonetic form of the word and the conversion of the phonetic form into the orthographic (i.e. written) form.

The usual approach to the OOV problem is to ignore the problem. In such a case, the decoder simply chooses some other — most fitting — word during the recognition process. If the vocabulary coverage is sufficient for the given task, the impact on the recognition accuracy is negligible. Fortunately, this is usually the case, especially when the recognition is limited to some specific domain.

The quality of a language model is usually measured by means of perplexity. The perplexity measure is defined as

$$\mathrm{PPL} = 2^{\mathrm{H}\{P\}} = 2^{-\sum_{i=1}^{N} \frac{1}{N} \log_2(P(w_i))}, \tag{3.9}$$

where $\mathrm{H}\{P\}$ is the entropy of language model P evaluated on the text W of length N, $W = w_1, w_2, \ldots, w_N$. The perplexity measure can be interpreted as the average number of words between which it has to be decided in the given time instance. The perplexity measure has problems with OOV words and unseen sequences in general, because of the $\log_2(\cdot)$ computation. There are two possible approaches of how to deal with it. The first approach is to add the OOV words from the testing data directly into the dictionary. The second approach is to ignore such event, which might be completely legitimate when different modeling methods are compared on the same data. In the latter case, it is, however, a good practice to publish the number of OOV words together with the achieved perplexity.

3.3.1 Language Models Based on N-grams

In general, the language model should be able to give the probability of a sequence of words for any length of the sequence. This is a very difficult task not only from the statistical point of view (with increased length of sequence, the number of observation drops to one), but also from the technical point of view (memory and computational requirements). The n-gram language models simplify this task significantly by imposing a limit on the length of the sequence to n words at most. The direct consequence is that word sequences identical in the first $n-1$ words will be treated as the same sequence.

The prominent n-gram models are as follows.

- Zerogram language models ($n = 0$), in which all the units have the same probability. This language model is often used for testing the acoustics models to remove the "bias" of the language.
- Unigram language models ($n = 1$), in which the probability of the unit is independent of the context and only the a-priori probability of the word is taken into account.
- Bigram language models ($n = 2$), in which the probability of the unit depends only on the previous word.
- Trigram language models ($n = 3$), in which the probability of the unit depends on a sequence of the two previous words. Trigram language models are commonly used for English.

For language modeling on words, $n = 2$ or $n = 3$ is usually used. Higher order language models on words do not usually bring significant improvement. In general, it is beneficial to use the more complex n-gram models in some specific cases, where the amount of units is relatively small. One such case is for example phoneme recognition, where 8-gram and even higher order language models are not an exception([86]).

3.3.2 Training of the N-gram Language Models

The probability of a seen n-gram $(w_{-n+1}, \ldots, w_{-1}, w_0)$ is easily determinable from the training text data using the following formula

$$P(w_0|w_{-n+1}, \ldots, w_{-1}) \approx \frac{N(w_{-n+1}, \ldots, w_{-1}, w_0)}{N(w_{-n+1}, \ldots, w_{-1})}, \tag{3.10}$$

where $N(w_{-n+1}, \ldots, w_{-1}, w_0)$ is the occurrences count of the sequence $w_{-n+1}, \ldots, w_{-1}, w_0$ and $N(w_{-n+1}, \ldots, w_{-1})$ is the number of occurrences of the sequence $w_{-n+1}, \ldots, w_{-1}$). In case of a unigram language model ($n = 1$), the formula is

$$P(w_0) \approx \frac{N(w_0)}{N}, \tag{3.11}$$

where N is the total length of the training text (number of tokens). The hitch of the straightforward application of the Eq. (3.10) or the Eq. (3.11) is that when any of the four corner cases discussed on the page 14 occurs, the obtained probability estimate can be highly biased. The solution to this problem is to apply *smoothing*. Smoothing assigns a non-zero probability to unseen n-grams (that would receive zero probability which would effectively prohibit recognition of the word sequence) and — to keep the equality $\sum_w P(w|w_{-n+1}, \ldots, w_{-1}) = 1$ valid — the probability of the seen n-grams is lowered accordingly. One of the most common approaches is referred to as the *back-off smoothing*, where the probability of an unseen n-gram is computed as

$$P(w_0|w_{-n+1}, \ldots, w_{-1}) \approx P(w_0|w_{-n+2}, \ldots, w_{-1})B(w_{-n+1}, \ldots, w_{-1}), \tag{3.12}$$

where the choice of function $B(\cdot)$ depends on the used backing-off technique([115]).

3.3.3 Neural Network Based Language Models

In recent years, the methods based on the statistical approach have been being complemented by the methods based on neural networks.

The training of the neural network has two phases ([8]).

1. Learning a mapping of the dictionary word w_i to a continuous-values vector representation $C(w_i)$ for all words in the dictionary V. This is usually done in the following way. The word w_i is represented by its dictionary index q. Then, the input feature vectors are created from the word indices by so called called 1-of-N mapping, i.e. the input vector representing a word w_i is obtained by setting the q-th element of the input vector to 1 and all other elements are set to 0.

   Then, a two layer neural network is trained. The topology of the neural network is $|V| \times H \times |V|$, where $|V|$ represents the size of the dictionary and H, $H \ll |V|$, represents the dimension of the hidden layer. The neural network input is the feature vector representing the $(i-1)$-th word in the training corpus (coded using the previously mentioned scheme) and the neural network target output is the i-th

word in the training corpus. During the process of training, the neural network learns to estimate the posteriori probability $P(r|w_j)$, which is the probability of the r-th dictionary word, given the preceding word w_j, for every valid r. In other words, upon convergence, the neural network represents a bigram language model.

The continuous mapping $C(w_i)$ is then obtained simply by considering the hidden layer outputs instead of the output layer outputs[74].

2. The continuous-values vector representation $C(w_i)$ is then used in the second stage to train the n-gram language model of the chosen complexity by joining the activation values vectors $C(w_{i-n+1}), \ldots, C(w_{i-1})$ of the individual words $w_{i-n+1}, \ldots, w_{i-1}$ into one large vector and training another network to provide the probabilities estimates as the outputs.

The drawback of this approach is the high computational complexity both during the training phase and during the recognition phase that scales quadratically with the size of dictionary. While for weakly inflected languages(such as English), a dictionary with 100k words is sufficient even for a general purpose LVCSR, for highly inflective languages the dictionary with the same OOV rate must be $5\times$ – $10\times$ larger.

The question if the neural network representation of the language model is superior to the common n-gram representation is still open, since the experiments are usually performed on small-to-medium size text corpora (because of the computational complexity of the training process). One notable exception is the work [101], where a sampling approach was used to train language models on a very large text corpus (600M of tokens). The authors report significant improvement in the cross-entropy measure as well as when the language model was used in the state-of-the art speech recognition system.

3.4 Speech Decoding Techniques

The model of speech decoder finds the most probable sequence of words $W^\star$

$$W^\star = \arg\max_W p(\boldsymbol{O}|W)P(W), \tag{3.13}$$

where the values of $p(\boldsymbol{O}|W)$ are provided by the Acoustic model module and the values of $P(W)$ are provided by the Language model module.

Direct (exhaustive) optimization of the formula Eq. (3.13) is impossible, because of the prohibitive number of all hypotheses on the sequence W. In reality, only a very limited fraction of hypotheses can be evaluated. Therefore only the most promising hypotheses are evaluated. The problem of an identification of the most promising hypotheses is one of the fundamental problems in the field of LVCSR, especially under the real-time constraints.

In the real world application, it is necessary to modify the criterion Eq. (3.13), because the probability densities provided by the language and the acoustic model are not

necessarily commensurable. The solution is to modify the search criterion to

$$\begin{aligned} W^\star &= \arg\max_W p(\boldsymbol{O}|W)P(W)^\beta \gamma^L \\ &= \arg\max_W \log p(\boldsymbol{O}|W) + \beta \log P(W) + L \log \gamma, \end{aligned} \tag{3.14}$$

where the value β is called the *language model weight*(LMW), γ is called the *word insertion penalty*(WIP) and L is the number of words in the actual hypothesis W.

The values of γ and β are determined experimentally during the development of the speech recognition system. The language model weight β constitutes a factor balancing the influences of the language model and the acoustic model, the word insertion penalty β is used for tuning of the ratio between the insertion and the deletion errors (see the next chapter).

3.5 Speech Recognition Accuracy Evaluation

For quantitative evaluation of a speech recognition system, the Levenshtein (or edit) distance[69] is used predominantly. The Levenshtein distance between two texts is defined as a minimum number of edit operations necessary to transform the recognized text W into the reference text W_{ref}. Three types of edit operations are allowed – an insertion, a deletion and a substitution. The number of insertions I, the number of deletions D and the number of substitutions S is then used to compute the *Correctness*(Corr) of the recognized text using the following formula

$$\text{Corr} = \frac{N - D - S}{N} \cdot 100\%, \tag{3.15}$$

where N is the length of W_{ref} and the *Accuracy*(Acc) of the recognized text defined as

$$\text{Acc} = \frac{N - D - S - I}{N} \cdot 100\%. \tag{3.16}$$

Sometimes, instead of the Acc measure, the *Word Error Rate*(WER) [2] is given. The WER measure is defined as

$$\text{WER} = 100 - \text{Acc} = \frac{D + S + I}{N} \cdot 100\%. \tag{3.17}$$

3.5.1 Absolute and Relative Improvement

Sometimes, when evaluating a new method or a new method implementation, the obtained Acc and Corr measures are compared against the baseline performance Acc_{ref} and Corr_{ref}

[2] or Phone Error Rate (PER) or Character Error Rate (CER), the interpretation differs, the formula is the same

and the absolute and relative improvements are computed. The absolute improvement of WER, Δ_{WER} is computed as

$$\Delta_{\mathrm{WER}} = \mathrm{WER}_{\mathrm{ref}} - \mathrm{WER} \qquad [\%]\,. \tag{3.18}$$

and similarly for the Corr and Acc measures. It can be seen by substituting the Eq. (3.17) into the Eq. (3.18) that actually $\Delta_{\mathrm{Acc}} = \Delta_{\mathrm{WER}}$.

The relative accuracy improvement Λ_{WER} is computed as

$$\Lambda_{\mathrm{WER}} = \frac{\Delta_{\mathrm{WER}}}{\mathrm{WER}_{\mathrm{ref}}} = \frac{\mathrm{WER}_{\mathrm{ref}} - \mathrm{WER}}{\mathrm{WER}_{\mathrm{ref}}} \qquad [\%] \tag{3.19}$$

and similarly for the Corr and Acc measures. Customarily, the relative improvement is reported relatively to the WER reduction, though.

3.5.2 Statistical Significance Tests

The above mentioned measures are often used to indicate an improvement above the baseline; however this is not a completely correct approach. To ensure validity of the result and to support the claim about the achieved improvement, the significance of the result should be performed to show how likely it is that the discussed results had occurred by chance.

The significance tests are usually performed by means of testing a *null hypothesis* H_0 against the alternative hypothesis. Based on the evidence (data), the null hypothesis either must be rejected or cannot be rejected. The important point is that the null hypothesis can never be proven.

The *significance level* α is the probability of rejecting the null hypothesis when the hypothesis actually holds. In other words, the significance level α corresponds to the type-I error. Popular choices of the α value are $\alpha = 0.05$ (5% chance) or $\alpha = 0.001$ (1% chance).

For the field of speech recognition analysis, the paired version of the Wilcoxon signed rank test ([105]) is suitable. It is a non-parametrical two-sided test for assessing whether the differences between the paired samples of observations come from a distribution having median equal zero. Moreover, it is assumed that the distribution of the differences is continuous and symmetrical about the median.

The testing process computes the differences between the matched values, discards zeroes and ranks the differences in an ascending order according the the absolute size of the difference. Then it computes the value $S = min(\kappa_+, \kappa_-)$, where κ_+ is the sum of ranks of the differences higher than 0 and κ_- is the sum of ranks of the differences lower than 0. Then, the value of S is compared to the critical value to determine whether H_0 can be rejected or not. For small sample sizes, the critical value is tabulated according to the sample size and the needed confidence level; for higher sample sizes, the distribution of S tends to normal distribution and therefore a normal approximation is usually used.

Nowadays, instead of comparing the value of S against critical values of the chosen significance level, a different approach is usually used. This approach is based on

computing the probability of obtaining the value of S at least as extreme as the one that was observed, assuming H_0 holds. This probability is called the p-value of the test. The hypothesis H_0 can be rejected, when p-value is less than the significance level α.

There are important philosophical and methodological differences between p-values and α-values, as the first comes from the Fisherian statistical significance testing and the second from the Neyman-Pearson frequentist views on hypothesis testing ([56]).

3.5.3 Confidence Intervals

Let's consider the situation when evaluating the Acc measure on testing data. The obtained score certainly is important, however the hitch is in the fact that it does not tell much about the stability of the performance. One might be for example interested in an interval of accuracy which the recognizer will provide in 95 % of the time. Or, because the accuracy of the recognizer depends on the speaker, in what interval the Acc will be for 95 % of speakers.

The *confidence intervals* (CI) in statistics are used to provide the answers to these questions. Similarly to the significance tests, the CI are tied to a probability value, in this case called the *confidence level*. The confidence interval with the given confidence level α is used to give an estimate of how probable it is, given the procedure used for getting the sample set, that the observed value will fall in the given confidence interval.

There are a wide variety of methods used for estimation of confidence intervals, differing in the assumption of the data distribution and/or computational complexity. Sometimes, instead of computing the CI values, the variance (or standard deviation) is used to describe the variability of the results. Assume that the result set have normal distribution $\mathcal{N}(\mu, \sigma^2)$. Then about 68 % of the result set lies in the interval $(\mu - \sigma, \mu + \sigma)$, about 95 % within the interval $(\mu - 2\sigma, \mu + 2\sigma)$ and 99.7 % within the interval $(\mu - 3\sigma, \mu + 3\sigma)$. The well known *three sigma rule*, used as a rule-of-thumb for outlier detection, stems from this observation.

This approach is however not suitable for the analysis of the recognizer accuracy. The distribution of the accuracy measure is highly non-gaussian, leptokurtic and skewed to the right. In such a situation, i.e. where the the distribution of the random variable is complicated or even unknown; the *bootstrapping* method is often used instead.

Bootstrapping is a computer-based method for measuring the accuracy of a statistical point estimate ([9]). The procedure is based on random sampling of the data-set with replacement. The data set is divided into K independent samples and sampled randomly R-times. From each of these samples, the point estimate is computed. In the end the sampling procedure ends with a set of R point estimates. Then it is trivial to determine the value $P_{-0.05}$ which is the 2.5 percentile and the value $P_{+0.05}$, which is the 97.5 percentile. These percentiles then constitute the 95% confidence interval $(P_{-0.05}, P_{+0.05})$. Generally, to determine α% confidence intervals, the $\frac{\alpha}{2}$ and $100 - \frac{\alpha}{2}$ percentiles must be evaluated.

The mentioned percentile method, despite the simplicity, works surprisingly well in most cases. It assumes that the distribution of the bootstrap point-estimate is symmetrical and centered around the observed point-estimate. In cases, when these assumptions are

not guaranteed, the *Accelerated Boostrap* (BCa) [28] method is usually suggested. The BCa method adjusts for bias (i.e centers the distribution on the observed point-estimate) and for skewness (i.e. symmetrizes bootstrap point-estimate distribution). A wide variety of alternative methods exists, which can be used in specific circumstances, where the previously described methods are not sufficient ([17]).

3.6 Conclusion

In this chapter, the basic introduction into the task of speech recognition has been given. The description is given from the probabilistic point of view. This approach, in combination with GMM/HMM modeling is the most prevalent nowadays. In this approach, the speech recognition task can be broken down into three distinctive tasks – language modeling, acoustic modeling and decoding techniques. Despite the fact that this decomposition leads to a significant reduction of the complexity of this task, the construction of a speech recognition engine is a formidable task.

Special attention has been paid to possibilities of use of the neural networks. Because of the relative versatility of neural networks, they can be used in acoustic modeling for discriminative features extraction or as posteriori probabilities estimators and in language modeling as n-gram probability estimators. Of course, the application of the neural network is not limited to these three tasks – these three task represent the main points of scientific and research interest.

4 Artificial Neural Networks

4.1 Biological Neural Networks

A neuron (or a neuron cell) is a cell that plays a key role in the processing of information in all living organisms. The simplified function of a neuron cell is as follows. By its dendrites, through so called synapses, the neuron receives stimuli (either inhibitory or excitatory) from other neurons. These stimuli are then combined (summed) together. Of course, stimuli received through different dendrites have different importance. When the combined potential of these stimuli reaches a certain threshold, the neuron itself sends a signal (action potential spike) through its axon (with other neurons' synapses connected to that axon) to the connected neurons. This process is called the *neuron firing*.

Of course the description of this process is highly simplified and does not cover all the possible information flows in the human brain, however it helps to emphasize the very important fact that has become the ground paradigm of the ANNs. The brain power is not specifically rooted in low amount of some finely tuned highly complex unit. On the contrary, the brain power emerges as a result of an extreme amount of "simple" computational units by virtue of their massive interconnection. The human brain contains approximately 10^{11} neurons and about 10^{14} connections[102].

A generally accepted fact is that the number of neurons decreases as the organism grows old. The latest findings suggest, however, that new neurons can actually be born during the lifetime of an organism[1]. Nevertheless, the ability to learn and to absorb new knowledge does not lean on the number of neurons specifically. Instead, during the process of learning, i.e. adoption of a new ability or storing a new knowledge into long-term memory, new connections (dendrite spines) between neurons are created and the weights of these interconnections are set accordingly.

A notable feature of biological neurons is that the action potentials are created on an all-or-nothing principle. The output action potential spike parameters do not depend on the intensity of the stimulus. Instead, the intensity of the stimulus is coded as a faster or a slower firing of the neuron.

[1] This is a subject to an intensive discussion. The evidence suggests that although the neurogenesis is possible and in no way rare in many evolutionary older parts of brain, the neurogenesis in neocortex, i.e. the part of mammal brain involved in higher cognitive functions — sensory perception, motor commands generation, spatial orientation, social and emotional processing, etc. — does not occur.

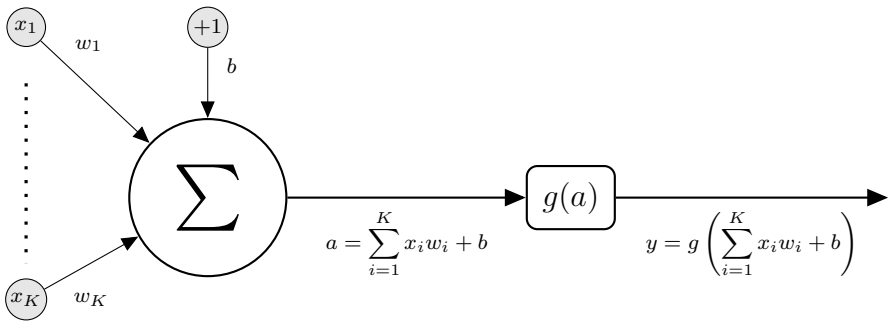


Figure 4.1: Perceptron: A computational model of a neuron.

4.2 Perceptron Unit

A single perceptron is an idealized computational unit inspired by the biological neuron. The perceptron unit activation potential a is defined as

$$a = \sum_{k=1}^{K} w_k \cdot x_k + b \tag{4.1}$$

eventually, using an vector dot-product, the Eq. (4.1) can be written as

$$a = \boldsymbol{w}^{\mathrm{T}} \cdot \boldsymbol{x} + b,$$

where x_k is the k-th input potential (*input*) of the neuron, w_k is the importance factor (*weight*) of the k-th input and b is the activation threshold (*bias*) of the perceptron. Moreover, the previous equation can be written as

$$a = \hat{\boldsymbol{w}}^{\mathrm{T}} \cdot \hat{\boldsymbol{x}}, \tag{4.2}$$

where $\hat{\boldsymbol{w}}$ is an augmented version of $\boldsymbol{w}$, $\hat{\boldsymbol{w}} = [w_1, w_2, \ldots, w_K, b]^{\mathrm{T}}$ and $\hat{\boldsymbol{x}}$ is an augmented version of $\boldsymbol{x}$, $\hat{\boldsymbol{x}} = [x_1, x_2, \ldots, x_K, 1]^{\mathrm{T}}$. In other words, the bias can be represented as an additional weight $\hat{w}_{K+1}$ connected to a virtual input whose value is fixed to $+1$. For clarity, the second notation will be preferred when possible. The action potential y is defined as

$$y = g\left(a\right), \tag{4.3}$$

where the function g is generally a non-linear function of the activation potential a representing the action potential signal propagation through the axon (*transfer function*).

4.3 Feedforward Multi-layer Perceptron

Perceptron units are usually combined together to form much more complicated structures. One class of these structures is called the Multi-layer Perceptron (MLP-ANN). A notable

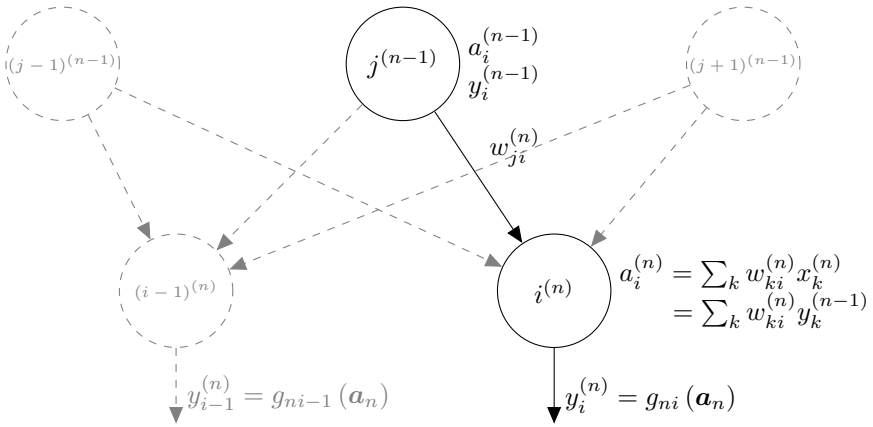


Figure 4.2: A Feedforward Multilayer Perceptron (MLP Network)

feature of the MLP-ANN is that the structure is strictly layered; works in a time discrete fashion and the signal is propagated only forwards, i.e. the outputs from one layer are fed as inputs to the consecutive layer.

An n-th layer of an L-layer MLP-ANN, $n \in \{1, \ldots, L\}$ contains K_n neurons. The scalar outputs y_{nk}, $1 \leq k \leq K_n$, of individual perceptrons are combined to form a vector $\boldsymbol{y}_n$, $\boldsymbol{y}_n = \left[y_1^{(n)}, y_2^{(n)}, \ldots, y_{K_n}^{(n)}\right]^{\mathrm{T}}$[2]. For an MLP-ANN with more layers it holds that $\boldsymbol{x}_n \equiv \boldsymbol{y}_{n-1}$, where $\boldsymbol{x}_n$ denotes inputs to layer n and $\boldsymbol{y}_{n-1}$ denotes the output of the $(n-1)$-th layer. Moreover, the $\boldsymbol{y}_0 \equiv \boldsymbol{x}_1$ is the input feature vector and $\boldsymbol{y}_L$ is the output of the MLP-ANN. The layers associated with $\boldsymbol{y}_n$ other than $\boldsymbol{y}_0$ or $\boldsymbol{y}_L$ are called hidden layers.

A matrix notation can be used to write the n-th layer of MLP-ANN as

$$\boldsymbol{a}_n = \boldsymbol{y}_{n-1}^{\mathrm{T}} \cdot \boldsymbol{W}_n + \boldsymbol{b}_n \tag{4.4}$$

$$\boldsymbol{y}_n = \boldsymbol{g}_n(\boldsymbol{a}_n), \tag{4.5}$$

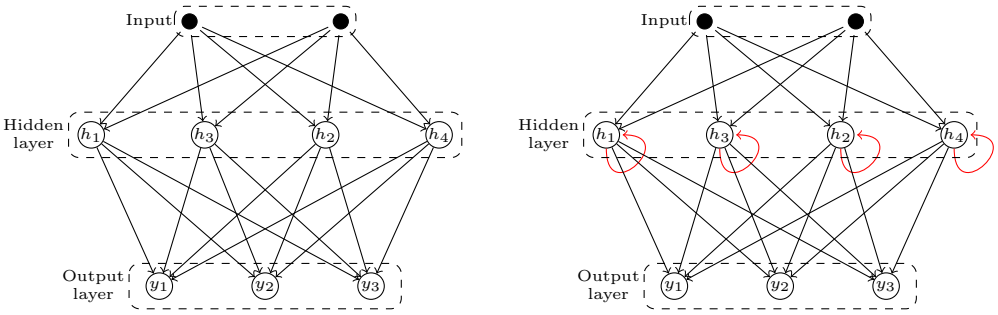
where the $K_{n-1} \times K_n$ matrix $\boldsymbol{W}_n$ whose i-th column is constituted by the weight vector $\boldsymbol{w}_i$ associated to the i-th neuron in the n-th layer is called the *weight matrix*, the vector $\boldsymbol{b}_n$, $\boldsymbol{b}_n = \left[b_1^{(n)}, \ldots, b_{K_n}^{(n)}\right]^{\mathrm{T}}$ is called the *bias vector*, the vector $\boldsymbol{a}_n$, $\boldsymbol{a}_n = \left[a_1^{(n)}, a_2^{(n)}, \ldots, a_{K_n}^{(n)}\right]^{\mathrm{T}}$ is the vector of activation potentials, and the vector function $\boldsymbol{g}_n : \Re^{K_l} \to \Re^{K_l}$ is called the *transfer function*.

The transfer function is sometimes called an *activation function*. The function $\boldsymbol{g}_l(\cdot)$ from Eq. (4.5) is in practical applications usually represented as a vector of scalar function of vector argument

$$\boldsymbol{g}_n(\boldsymbol{a}_n) = \left[g_{n1}(\boldsymbol{a}_n), g_{n2}(\boldsymbol{a}_n), \ldots, g_{nK_n}(\boldsymbol{a}_n)\right]^{\mathrm{T}} \tag{4.6}$$

and, furthermore, it usually holds $g_{ni}(\cdot) \equiv g_{nj}(\cdot)$ for every $i, j \in \{1, \ldots, K_n\}$. In that case, for the sake of legibility, $g_{ni}(\boldsymbol{a}_n)$ will be written as $g_n(\boldsymbol{a}_n)$. The Fig. 4.2 demonstrates

[2]Please note that the superscript (n) does not mean exponentiation nor differentiation, it is only used to signify the associated layer of the neural network



(a) A feedforward neural network, a multi-layer perceptron

(b) A recurrent neural network, a recursive multilayer perceptron

Figure 4.3: An example of two different architectures of an ANN

the computation of the activation potentials of the n-th layer given the weights and activation potentials of the $(n-1)$-th layer.

There exists a slight confusion regarding the description of the topology of a MLP-ANN. The topology of any generally fully connected MLP-ANN can be given by means of a set D, $D = \{K_0, K_1, \ldots, K_L\}$ and a set G, $G = \{\boldsymbol{g}_1(\cdot), \ldots, \boldsymbol{g}_L(\cdot)\}$, where K_0 is the dimension of $\boldsymbol{y}_0$ (dimension of the input vector) and $|D| = L + 1$ and $|G| = L$. The same network with $L = 2$ is sometimes referred to as a *three-layer* network (because $|D| = 3$) and sometimes (possibly more correctly) as a *two-layer* network (because $L = 2$). Sometimes, to prevent misunderstanding, the network is referred to as a *two-layer network with one hidden layer*.

4.4 Recurrent Multi-layer Perceptron

Recurrent multi-layer perceptron (RMLP, RNN) relaxes the strictly feed-forward layer ordering as found in the feed-forward MLP architecture. When the assumption of ordering is relaxed, an internal memory can be realized by virtue of directed cycles in the network – see the Fig. 4.3(b). This memory can be used for storing of an internal state of the network. The immediate consequence is that this class of networks can analyze and model temporal dependencies.

The topologies can vary. The basic architecture is called a *fully recurrent network*. The neurons in this network are fully connected, i.e. each neuron is connected to every other neuron. Some of the neurons receive the input signal in addition to the signal from the other neurons and the RNN output is again obtained from a part of the non-input neurons. The non-input and non-output neurons are referred to as hidden neurons.

To train the weights of the fully connected topology, however, is a quite complicated task. Moreover, the network is usually highly sensitive to the input and can often behave chaotically. Usually, simplified architectures are used, such as the Elman([29]) or the Jordan networks. Both these architectures build up on the original feed-forward architecture. The internal state is represented by the activation values in the hidden layer.

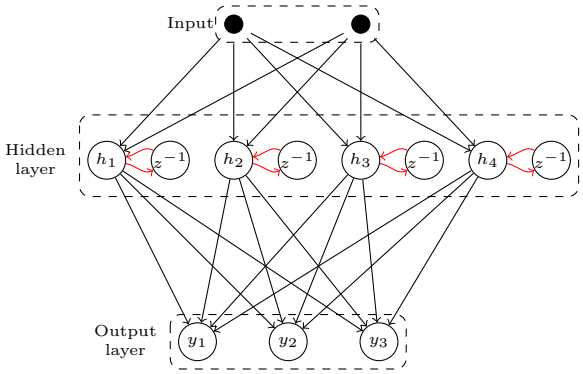


Figure 4.4: A scheme of an Elman network.

The activation values from the previous time instance are kept in a set of fixed-weight units. At the successive time instance, the input is propagated in the feed-forward fashion, the previous activation values are added to the activation potential a and the action potential is evaluated. The activation values are then stored to be used in the next time instance.

The training of the architectures mentioned above is done either through global optimization methods, notably genetic algorithms or through a modification of the steepest gradient descent modification called *Backpropagation through time* (BPTT). The drawback of the global optimization methods is their heuristic nature and slow convergence rate. The problem with the BPTT training is that it usually fails to discover long temporal dependencies. The root cause is because of the effect of vanishing gradients. An detailed analysis of this phenomenon was performed in [49, 50].

The Long short term memory (LSTM) [51] architecture was developed to alleviate the previously mentioned drawbacks. An LSTM network contains a special kind of nodes called the LSTM nodes. The speciality of LSTM nodes lies in their ability to remember the given value for an arbitrary length of time. Each LSTM node consists of two or three gates – an input gate, an output gate and an optional forgetting gate. The input gate decides, if the input value is significant enough to be remembered, the output gate decides, if the remembered value should be presented at the output of the LSTM node and the forgetting gate controls the forgetting of the remembered value. The training of an LSTM network is done via the BPTT.

4.5 Activation Functions

The transfer functions are non-linear. Let's consider case, when $\boldsymbol{g}_l$ is linear, i.e. $\boldsymbol{y}_l = \boldsymbol{a}_l$. Using Eq. (4.4) and substituting $\boldsymbol{y}_l$ into the $l+1$-th layer leads to

$$\boldsymbol{a}_{l+1} = \left(\boldsymbol{y}_{l-1}^{\mathrm{T}} \cdot \boldsymbol{W}_l + \boldsymbol{b}_l\right)^{\mathrm{T}} \cdot \boldsymbol{W}_{l+1} + \boldsymbol{b}_{l+1} \tag{4.7}$$

$$\boldsymbol{y}_{l+1} = \boldsymbol{g}_{l+1}(\boldsymbol{a}_{l+1}) \tag{4.8}$$

and then Eq. (4.7) can be easily converted to

$$\boldsymbol{a}_{i+1} = \boldsymbol{y}_{l-1}^{\mathrm{T}} \underbrace{\boldsymbol{W}_l \boldsymbol{W}_{l+1}}_{\boldsymbol{W}'_{l+1}} + \underbrace{\boldsymbol{b}_l^{\mathrm{T}} \cdot \boldsymbol{W}_{l+1} + \boldsymbol{b}_{l+1}}_{\boldsymbol{b}'_{l+1}}. \tag{4.9}$$

Therefore, for any MLP-ANN with Q consecutive linear layers, an equivalent MLP-ANN with $L - Q + 1$ layers exists.

A wide variety of functions exists that can be used in place of $g_{li}(\boldsymbol{a}_l)$. The activation function used in the original work by Minsky and Papert [75] is called the *hard-step* function

$$g_{li}(\boldsymbol{a_l}) = \begin{cases} 0 & \text{when } a_{li} < 0 \\ 1 & \text{when } a_{li} \geq 0 \end{cases}. \tag{4.10}$$

The drawback in using the hard-step function is that its derivative is not continuous, which poses a problem for gradient training methods.

The two following functions are used instead of the hard-step function.

$$g_{li}(\boldsymbol{a_l}) = \frac{1}{1 + \exp(-a_{li})} \tag{4.11}$$

$$g_{li}(\boldsymbol{a_l}) = \frac{\exp(a_{li}) - \exp(-a_{li})}{\exp(a_{li}) + \exp(-a_{li})} \tag{4.12}$$

They both belong to the family of sigmoidal functions, the latter is called the *hyperbolic tangent* and the former is usually denoted as the *logistic sigmoid* function. Sometimes, when no confusion is possible, the logistic sigmoid function is denoted simply as *the sigmoid* function. The $\tanh(\cdot)$ function has the same shape as the logistic sigmoid function, they differ only in their range (co-domain). The relationship between these two functions can be formalized as

$$\frac{1}{2} \cdot \tanh\left(\frac{a_{li}}{2}\right) + \frac{1}{2} \equiv \frac{1}{1 + \exp(-a_{li})}. \tag{4.13}$$

Thus, an MLP-ANN with the hidden layer(s) employing the $\tanh(\cdot)$ function is equivalent to an MLP-ANN with logistic sigmoid activation function with weights and biases adapted appropriately. The empirical evidence suggests, however, that using the $\tanh(\cdot)$ function leads to faster convergence during training.

The last function introduced here, in Eq. (4.14), is called the *softmax* function.

$$g_{li}(\boldsymbol{a_l}) = \frac{\exp(a_{li})}{\sum_{k=1}^{K_l} \exp(a_{lk})} \tag{4.14}$$

The name originates from the fact that the softmax function represents a differentiable version of the $\max(\cdot)$ operator, where the largest element of the input vector $\boldsymbol{a}_l$ will receive a value close to 1 and the other elements will receive values close to 0. Moreover, it holds that $\sum_{i=1}^{K_l} g_{li}(\boldsymbol{a_l}) = 1$. The *softmax* function can be thought of as a generalization of the *logistic sigmoid* function. Moreover, it has some remarkable properties in the sense of interpreting the outputs as posterior probabilities. Because of these properties, it is often used as the *activation* function of the output layer. These properties will be discussed later in more detail.

4.6 Training of Multi-layer Perceptron Networks

As it has already been mentioned, the architecture of an MLP-ANN is defined by pair (D, G). However, the domain knowledge of the MLP-ANN is encoded as a parameter set $\boldsymbol{W} = \left\{\hat{\boldsymbol{W}}_1, \ldots, \hat{\boldsymbol{W}}_L, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_L\right\} = \left\{\hat{\boldsymbol{W}}_1, \ldots, \hat{\boldsymbol{W}}_L\right\}$, i.e. as a set of all weights, represented either as a set of weight matrices and the associated bias vectors or as a set of the augmented weight matrices.

In the scope of this work, the training process is *supervised*. It means that for each presented input $\boldsymbol{x} \equiv \boldsymbol{y}_0$ and the associated network-computed response (output) $\boldsymbol{o} \equiv \boldsymbol{y}_L(\boldsymbol{x})$, the teacher information vector $\boldsymbol{t}$ exists. The vector $\boldsymbol{t}$ carries the meaning of the expected (correct) output.

Therefore, for a set of T training pairs $\Psi = \{(\boldsymbol{x}_0, \boldsymbol{t}_0), \ldots, (\boldsymbol{x}_{T-1}, \boldsymbol{t}_{T-1})\}$, the task is to find $\boldsymbol{W}^\star$, i.e. to find such a set $\boldsymbol{W}$ that minimizes an a priori chosen compound loss function $E(\Psi|\boldsymbol{W})$.

$$\boldsymbol{W}^\star = \arg\min_{\boldsymbol{W}} E(\Psi|\boldsymbol{W}) \tag{4.15}$$

Usually, the function $E(\Psi|\boldsymbol{W})$ has the form

$$E(\Psi|\boldsymbol{W}) = \sum_{q=0}^{T-1} E^q, \tag{4.16}$$

where T is the number of training samples and E^q is the error contribution of the q-th training pattern.

4.6.1 The Most Frequent Error Functions

Two common forms of function E^q are the *mean square error* function

$$E^q = \frac{1}{2}||\boldsymbol{o}(\boldsymbol{x}_q) - \boldsymbol{t}_q||^2, \tag{4.17}$$

and the *cross-entropy error* function

$$E^q = -\sum_k t_{qk} \ln o_k(\boldsymbol{x}_q), \tag{4.18}$$

where t_{qk} denotes k-th element of vector $\boldsymbol{t}_q$, analogously $o_k(\boldsymbol{x}_q)$ denotes k-th element of vector $\boldsymbol{o}(\boldsymbol{x}_q)$. The minimum of this function occurs when $o_k(\boldsymbol{x}_q) = t_{qk}$ and its value is $E^q = -\sum_k t_{qk} \ln t_{qk}$, which is a non-zero value in a general case. This minimal value can be subtracted from the error function to guarantee that the minimum error will be always zero. The normalized cross-entropy function is defined as

$$E^q = -\left(\sum_k t_{qk} \ln o_k(\boldsymbol{x}_q) - \sum_k t_{qk} \ln t_{qk}\right) = -\sum_k t_{qk} \ln \frac{o_k(\boldsymbol{x}_q)}{t_{qk}}. \tag{4.19}$$

It is worth noticing that the function defined by the Eq. (4.19) is actually the *Kullback–Leibler divergence* between the probability distributions from which the samples t_{qk} and $o_k(\boldsymbol{x}_q)$, for all admissible values of k, are drawn. In reality however, the Eq. (4.19) is often referred to as the cross-entropy function. Additionally, some other criterions based on cross-entropy error function exist[107].

4.6.2 Backpropagation of an Error in the Multi-layer Perceptron

When functions $g_i(\cdot)$ are differentiable, iterative gradient descent optimization methods can be used to determine the value $\boldsymbol{W}^\star$. This class of methods solves the following set of equations

$$\frac{\partial E(\Psi|\boldsymbol{W})}{\partial \boldsymbol{W}} = 0, \tag{4.20}$$

and each iteration t can be divided into four consecutive stages

1. compute the gradient $\nabla\boldsymbol{W}(t)$, $\nabla\boldsymbol{W}(t) = \frac{\partial E(\Psi|\boldsymbol{W})}{\partial \boldsymbol{W}(t)}$
2. compute the weight update Δ_t, $\Delta_t \approx \nabla\boldsymbol{W}(t)$
3. compute the new weights $\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \Delta_t$
4. evaluate the stopping criterion and either finish the optimization or continue again with the stage 1.

Evaluating the Gradient $\nabla\boldsymbol{W}(t)$ The algorithm for evaluating the gradient $\nabla\boldsymbol{W}(t)$ is called the *backpropagation*. It has been discovered, forgotten and discovered again several times, however it finally gained wide recognition after 1974, when it was published in [122].

Suppose the augmented formulation of a neural network as given by the Eq. (4.2). Now consider evaluation of the derivative E^q with respect to some weight $w_{ji}^{(n)}$ between the node i in the n-th layer and the node j in the $(n-1)$-th layer. Since the E^q depends on $w_{ji}^{(n)}$ only through the value of $a_i^{(n)}$ (activation value of the i-th neuron), the chain rule can be applied, which leads to

$$\frac{\partial E^q}{\partial w_{ji}^{(n)}} = \frac{\partial E^q}{\partial a_i^{(n)}} \frac{\partial a_i^{(n)}}{\partial w_{ji}^{(n)}}. \tag{4.21}$$

Using the Eq. (4.1) it can be seen instantly that the derivative $\frac{\partial a_i^{(n)}}{\partial w_{ji}^{(n)}}$ is

$$\frac{\partial a_i^{(n)}}{\partial w_{ji}^{(n)}} = x_j^{(n)} = y_j^{(n-1)}. \tag{4.22}$$

Usually, an additional substitution

$$\sigma_i^{(n)} \equiv \frac{\partial E^q}{\partial a_i^{(n)}} \tag{4.23}$$

is used and the quantities $\sigma_i^{(n)}$ are referred to as *errors*. Substituting both the Eq. (4.22) and the Eq. (4.23) into the Eq. (4.21) leads to

$$\frac{\partial E^q}{\partial w_{ji}^{(n)}} = \sigma_i^{(n)} y_j^{(n-1)}, \tag{4.24}$$

which tells us that the required derivative can be simply obtained by multiplication of the value of $\sigma_i^{(n)}$ by the value $y_j^{(n-1)}$. The values of $y_j^{(n-1)}$ are known from the forward propagation, i.e. from calculation of the output of the neural network. Thus, in order to evaluate the complete set of derivatives, only the quantities $\sigma_i^{(n)}$ for every (hidden or output) neuron in the network has to be evaluated and then the Eq. (4.21) has to be applied.

For the output units ($n = L$), where y_i does depend only on the a_i, the formula describing evaluation of $\sigma_i^{(n)}$ is

$$\sigma_i^{(n)} = \frac{\partial E^q}{\partial a_i^{(n)}} = \frac{\partial E^q}{\partial y_i^{(n)}} \frac{\partial y_i^{(n)}}{\partial a_i^{(n)}} = \frac{\partial E^q}{\partial y_i^{(n)}} \underbrace{\frac{\partial}{\partial a_i^{(n)}} g_{ni}\left(\boldsymbol{a}_n\right)}_{g'_{ni}\left(a_i^{(n)}\right)} \tag{4.25}$$

by using the chain rule on the Eq. (4.23). In cases, where y_i does depend also on some other a_k, $k \neq i$, (as in the case of the softmax transfer function), the correct formula is

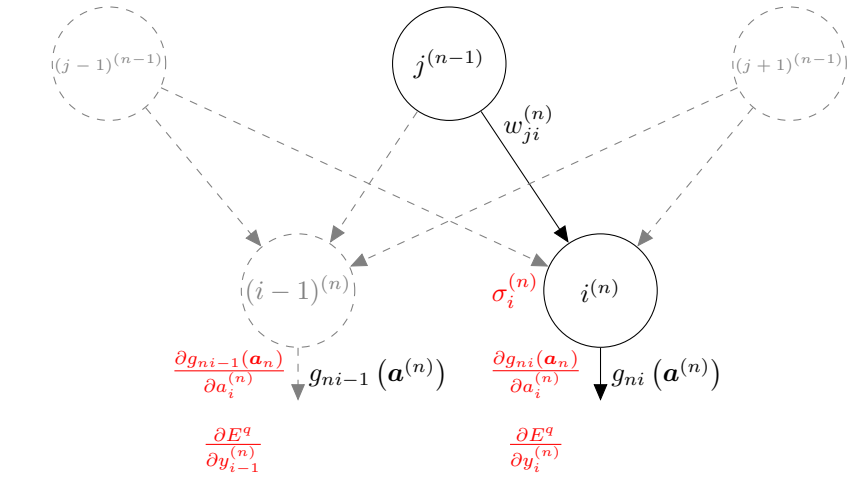
$$\sigma_i^{(n)} = \frac{\partial E^q}{\partial a_i^{(n)}} = \sum_k \frac{\partial E^q}{\partial y_k^{(n)}} \frac{\partial y_k^{(n)}}{\partial a_i^{(n)}} = \sum_k \frac{\partial E^q}{\partial y_k^{(n)}} \underbrace{\frac{\partial}{\partial a_i^{(n)}} g_{nk}\left(\boldsymbol{a}_n\right)}_{g'_{nk}\left(a_i^{(n)}\right)} \tag{4.26}$$

The symbol $g'_{ni}\left(a_i^{(n)}\right)$ denotes the first derivative of the transfer functions $g_{ni}\left(\boldsymbol{a}_n\right)$ with respect to $a_i^{(n)}$ and, similarly, the symbol $g'_{nk}\left(a_i^{(n)}\right)$ denotes the first derivative of the transfer function function $g_{nk}\left(\boldsymbol{a}_n\right)$ with respect to $a_i^{(n)}$. The Fig. 4.5(a) depicts the calculation scheme.

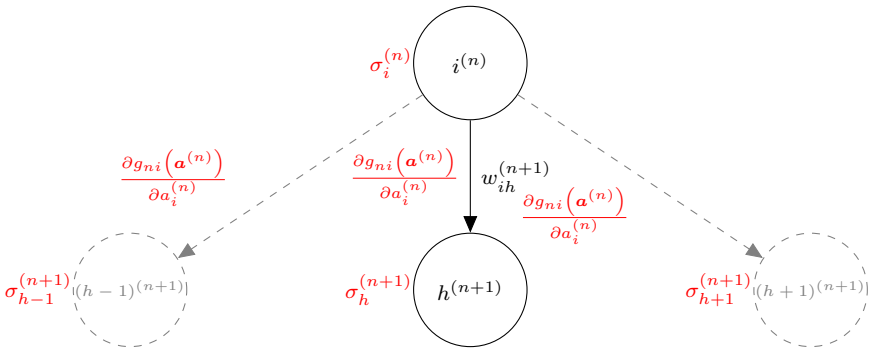
For the hidden layer units, using the chain rule leads to

$$\begin{aligned} \sigma_i^{(n)} &= \sum_h \frac{\partial E^q}{\partial a_h^{(n+1)}} \frac{\partial a_h^{(n+1)}}{\partial a_i^{(n)}} \\ &= \sum_h \frac{\partial E^q}{\partial a_h^{(n+1)}} \frac{\partial a_h^{(n+1)}}{\partial y_i^{(n)}} \frac{\partial y_i^{(n)}}{\partial a_i^{(n)}}, \end{aligned} \tag{4.27}$$

where the sum runs over all units h to which the unit i is connected. The units h can belong into another hidden layer or they can belong into the output layer. Applying the identity Eq. (4.23) on $\frac{\partial E^q}{\partial a_h^{(n+1)}}$ and evaluating the gradient $\frac{\partial a_h^{(n+1)}}{\partial y_i^{(n)}}$ using the Eq. (4.1), the



(a) Output layer neurons



(b) Hidden layer neurons

Figure 4.5: A scheme of backpropagation computation process of σ_i.

following recursive formula is obtained

$$\sigma_i^{(n)} = \underbrace{\frac{\partial}{\partial a_i^{(n)}} g_{ni}\left(\boldsymbol{a}_n\right)}_{g'_{ni}\left(a_i^{(n)}\right)} \sum_h w_{ih}^{(n+1)} \sigma_h^{(n+1)}. \tag{4.28}$$

The calculation scheme is depicted in the Fig. 4.5(b).

The recursive application of this formula is the key for computing the derivatives of the neural network parameters. For computation of all the quantities $\sigma_i^{(n)}$, it is necessary to determine the values $\frac{\partial}{\partial a_i^{(n)}} g_{ni}\left(\boldsymbol{a}_n\right)$ in the Eq. (4.25) and in the Eq. (4.28) and also the values $\frac{\partial E^q}{\partial y_i^{(L)}}$ in the Eq. (4.25). The quantities $\frac{\partial E^q}{\partial w_{ji}^{(n)}}$ can be obtained by virtue of using the formula Eq. (4.24).

The derivative of the total error E with respect to the given weight w_{ij} is simply obtained by virtue of evaluating the error E^q for every training sample and summing the

individual contributions as

$$\frac{\partial E}{\partial w_{ji}^{(n)}} = \sum_q \frac{\partial E^q}{\partial w_{ji}^{(n)}}. \tag{4.29}$$

A Case of a Two-layer Network In the field of speech recognition, two-layer networks with sigmoidal activation functions defined by the Eq. (4.11) in the hidden layer and the softmax activation function defined by the Eq. (4.14) in the output layer are used predominantly. This network is then trained using the cross-entropy criterion given by the Eq. (4.19). The derivative $g'_{ni}(\boldsymbol{a_n})$ of the sigmoidal function can be obtained using the quotient rule as

$$g'_{ni}(\boldsymbol{a}_n) = \frac{1}{\left(1 + \exp\left(-a_i^{(n)}\right)\right)^2} \exp\left(-a_i^{(n)}\right), \tag{4.30}$$

which can be transformed using the Eq. (4.11) into

$$g'_{ni}(\boldsymbol{a}_n) = g_{ni}(\boldsymbol{a}_n)(1 - g_{ni}(\boldsymbol{a}_n)) = y_i^{(n)}\left(1 - y_i^{(n)}\right). \tag{4.31}$$

The formula given by the Eq. (4.31) provides a way how to compute the derivative using only elementary algebraic operations (one multiplication and one subtraction) from the original activation function value, which is a very useful feature used in many software implementations.

Let's evaluate the quantities σ_i in the output layer. Because of the softmax transfer function, there exists a dependence between a_i and y_j even for $i \neq j$. The quantities σ_i can be obtained by virtue of the Eq. (4.26).

$$\sigma_i^{(n)} = \sum_j \frac{\partial E^q}{\partial g_{nj}(\boldsymbol{a}_n)} \frac{\partial g_{nj}(\boldsymbol{a}_n)}{\partial a_i^{(n)}}. \tag{4.32}$$

Using the definition Eq. (4.14), the derivative $\frac{\partial g_{lj}(\boldsymbol{a}_l)}{\partial a_i^{(n)}}$ function is

$$\frac{\partial g_{nj}(\boldsymbol{a}_n)}{\partial a_i^{(n)}} = \begin{cases} \frac{\exp\left(a_i^{(n)}\right)}{\sum_{j=1}^{K_n} \exp\left(a_j^{(n)}\right)} - \frac{\exp\left(2a_i^{(n)}\right)}{\left(\sum_{j=1}^{K_n} \exp\left(a_j^{(n)}\right)\right)^2} & \text{if} \quad i = j, \\ -\frac{\exp\left(a_i^{(n)} + a_j^{(n)}\right)}{\left(\sum_{j=1}^{K_n} \exp\left(a_j^{(n)}\right)\right)^2} & \text{else,} \end{cases} \tag{4.33}$$

which can be written as

$$\frac{\partial g_{nj}(\boldsymbol{a}_n)}{\partial a_i^{(n)}} = (g_{nj}(\boldsymbol{a}_n)\delta_{ij} - g_{ni}(\boldsymbol{a}_n)g_{nj}(\boldsymbol{a}_n)) = \left(y_j^{(n)}\delta_{ij} - y_i^{(n)}y_j^{(n)}\right), \tag{4.34}$$

where the symbol δ_{ij} denotes the Dirac delta. The partial derivative $\frac{\partial E^q}{\partial g_{nj}(\boldsymbol{a}_n)}$, given the fact that the cross-entropy criterion, which is used, is given by the Eq. (4.19) is defined by the formula

$$\frac{\partial E^q}{\partial g_{nj}(\boldsymbol{a}_n)} = -\frac{t_{qj}}{g_{nj}(\boldsymbol{a}_n)} = -\frac{t_{qj}}{y_j^{(n)}}. \tag{4.35}$$

Substituting the equations Eq. (4.34) and Eq. (4.35) into the formula Eq. (4.25) will result in the following formula

$$\begin{aligned}\sigma_i^{(n)} &= \sum_j -\frac{t_{qj}}{y_j^{(n)}}\left(y_j^{(n)}\delta_{ij} - y_i^{(n)}y_j^{(n)}\right)\\ &= \sum_j -t_{qj}\delta_{ij} + \sum_j t_{qj}y_i^{(n)}\\ &= -t_{qi} + \sum_j t_{qj}y_i^{(n)}.\end{aligned} \tag{4.36}$$

Considering that $\sum_j t_{qj} = 1$, the previous formula can be transformed to

$$\sigma_i^{(n)} = y_i^{(n)} - t_{qi}. \tag{4.37}$$

For the hidden layer, substituting the expression Eq. (4.31) into the Eq. (4.28) will give us

$$\sigma_i^{(n)} = \left(y_i^{(n)}\left(1 - y_i^{(n)}\right)\right)\sum_h \sigma_h^{(n+1)}w_{ih}^{(n+1)}. \tag{4.38}$$

For the real world applications however, when performance is of the essence, it is usually more practical to use the matrix-vector formulation of the computation. The formulae for the output layer are

$$\boldsymbol{\sigma}_L = \boldsymbol{t} - \boldsymbol{y}_L, \tag{4.39}$$

$$\nabla\hat{\boldsymbol{W}}_L = \boldsymbol{\sigma}_L^{\mathrm{T}}\boldsymbol{y}_{L-1}, \tag{4.40}$$

and for the hidden layers

$$\boldsymbol{\sigma}_{L-i} = \boldsymbol{y}_{L-i} \otimes (1 - \boldsymbol{y}_{L-i}) \otimes (\boldsymbol{\sigma}_{L-i+1}\hat{\boldsymbol{W}}_{L-i+1}) \tag{4.41}$$

$$\nabla\hat{\boldsymbol{W}}_{L-i} = \boldsymbol{\sigma}_{L-i}^{\mathrm{T}}\boldsymbol{y}_{L-i-1} \tag{4.42}$$

where operation $\otimes$ denotes element-wise multiplication and for $\nabla\boldsymbol{W}$ intuitively holds $\nabla\boldsymbol{W} = \left\{\nabla\hat{\boldsymbol{W}}_1, \ldots, \nabla\hat{\boldsymbol{W}}_L\right\}$

Computing the Weight Update $\boldsymbol{\Delta}_{t+1}$ For the simplest gradient descent algorithm, the update is computed as

$$\boldsymbol{\Delta}_{t+1} = -\lambda\nabla\boldsymbol{W} \tag{4.43}$$

Unfortunately, because of the simplicity of this algorithm, the convergence is often slow and sometimes even stops altogether, because the gradients further away from the output layer are zero or almost-zero. This occurs quite often when training an MLP-ANN with three or more layers with increasing occurrence as the number of layers of the MLP-ANN increases. This phenomenon is called *effect of vanishing gradients*. It is usually a difficult issue in connection with training of *recurrent neural networks* by means of so-called *backpropagation through time*, but it can be lethal even for training the MLP-ANN [7].

Computing the New Weights $\boldsymbol{W}(t+1)$ After the proper weight update $\boldsymbol{\Delta}_{t+1}$ is computed, the new weights are commonly determined as

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \boldsymbol{\Delta}_{t+1}. \tag{4.44}$$

The new weights are then used subsequently in the next iteration of training. The stopping criterions are varied. Usually, one or a combination of some of the following is used.

- The maximum number of iteration has been reached.
- The gradient is close to zero.
- The error value E do not improve significantly anymore.

4.6.3 Speeding Up the Training Process

A lot of attention has been paid to speeding up the convergence of the training process[103]. One possible approach is to preprocess the training data. The preprocessing usually includes normalization of the input variables to have the mean close to zero and a constant variance. Usually, not only the variance normalization, but also the decorelation is performed. Other accelerating heuristics include the careful initialization of the weights[65], choosing the learning rate (or adaptation scheme of the learning rate) properly and choosing the proper target class mapping[67].

Another class of faster algorithms uses the Hessian matrix information. Employing the Hessian matrix often leads to a significantly faster convergence. However, besides the obvious requirement of twice continuously differentiable activation functions, these methods become impractical as the number of trained weights increases, because the storage requirements depends quadratically on the number of parameters of the neural network being trained. The memory demands can be relaxed and the computational complexity reduced furthermore, when the Hessian matrix is of blocked shape[21].

Newton Method Because of the problems associated with usage of the simple gradient descent algorithm, it is beneficial to consider the curvature of the optimized function in the optimization algorithm. The curvature information allows for choosing of a more direct step towards the minimum. The Newton method constructs a local quadratic approximation of the error function and then performs a direct step into the point of expected minimum of the approximation. For the local quadratic approximation, not only the gradient but also the Hessian must be evaluated, which can be both time consuming and memory demanding. Especially the memory is a concern for neural networks with more than a few thousand weights, because the Hessian matrix storage need depends quadratically on the number of parameters.

Levenberg-Marquardt (L-M) Method This algorithm is designed specifically for the mean square error function from the Eq. (4.17). Under this assumption, the Hessian matrix can be approximated using only the gradient $\nabla \boldsymbol{W}(t+1)$. L-M method is an example of model trust region approach in which a model of the original function is constructed; however the trust in this model is limited only to a limited neighborhood of the original point. The size of the neighborhood is governed by an automatically tuned parameter λ. For small values of λ, the L-M behaves almost like real Newton method (with quadratic convergence rate) and for large values of λ, the algorithm falls back to the standard gradient descent algorithm.

BFGS Method The BFGS belongs into the class of *quasi-Newton methods*. Instead of computing the whole Hessian matrix, only an approximation is computed and updated iteratively as the algorithm progresses. This alleviates the computational demands of the Newton method, however it has the same asymptotic memory requirement. Because of the storage demands, the next method is usually preferred.

Limited Memory BFGS (L-BFGS) Unlike the original "full" BFGS method, the L-BFGS method does not store $n \times n$ (where n is the number of parameters) elements of the Hessian matrix but only a few (let's say m) vectors of dimension n. This set of vectors represents the approximation implicitly. In most of applications, $4 \leq m \leq 10$, which leads to significantly reduced memory demands, since $m \ll n$.

Resilient Propagation (RPROP) and Improved Resilient Propagation (iRPROP) The RPROP method seeks to improve the convergence rate in cases where the phenomenon of vanishing gradient occurs. Instead of using the size of the gradient to compute the step size, the algorithm uses merely the sign of the gradient to determine the direction towards the minimum. The step size is determined adaptively. The original approach ([91]) has been somewhat modified in [58], using the total previous error value to improve the overall convergence rate.

Quickprop The Quickprop method ([30]) assumes the error surface is locally quadratic and computes the step size leading directly into the minimum of the assumed parabola. The problem with this method is tied to the inherent assumption of the local shape of the error function. For some error surfaces, this can lead to a wild behavior and slow convergence or no convergence at all. These problems have been alleviated in [118], where a globally convergent modification of the Quickprop algorithm has been devised including proofs of the global convergence.

4.6.4 Natural Pairing of an Error Function and Transfer Functions

As has been noted in some publications ([26, 15]), a special pairing between some types of the output layer activation functions $g_{Lk}(\boldsymbol{a}_L)$, $1 \leq k \leq K_L$, and the error functions

E^q exists. The distinctive property of such pairs is the interaction between the error function and the transfer function during training.

Consider a neural network with the softmax transfer function from the Eq. (4.14) in the output layer and the cross-entropy error function given by the Eq. (4.19) used for training. Recall that the error quantity $\sigma_i^{(n)} \equiv \frac{\partial E^n}{\partial a_i^{(n)}}$ for the output layer (n) is

$$\frac{\partial E^q}{\partial a_i^{(n)}} = \sum_j \frac{\partial E^n}{\partial y_j^{(n)}} \frac{\partial y_j^{(n)}}{\partial a_i^{(n)}}. \tag{4.45}$$

The combination of softmax transfer function and the cross-entropy error function can be simplified to

$$\frac{\partial E^q}{\partial a_i^{(n)}} = \left(y_i^{(n)} - t_{qi} \right). \tag{4.46}$$

For complete derivation of this formula refer to the page 32. As has been noted (see [12]), this is the same result as if the combination of the linear transfer function and the Mean Square Error criterion defined in the Eq. (4.17) was used. The important observation is that the error signal is not attenuated just right in the beginning of the training process. When using the sigmoid transfer function in combination with the mean-square error function, the formula is

$$\frac{\partial E^q}{\partial a_i^{(n)}} = y_i^{(n)} \left(1 - y_i^{(n)} \right) \left(y_i^{(n)} - t_{qi} \right). \tag{4.47}$$

Taking the fact that $0 \leq y_i \leq 1$ and $0 \leq t_i \leq 1$ into account, it is easy to see that the error signal is attenuated, which can have a negative impact on the convergence speed [12]. Indeed, suppose the expected (wanted) classification decision $t_i = 0$ and the obtained (real) classification $y_i = \varepsilon$. Substituting into the Eq. (4.47) yields $\frac{\partial E^q}{\partial a_i^{(n)}} = \varepsilon^2(1-\varepsilon)$. Now, suppose the opposite case, i.e. $y_i = 1 - \varepsilon$ and $t_i = 1$. Substituting into the Eq. (4.47) yields $\frac{\partial E^q}{\partial a_i^{(n)}} = -\varepsilon^2(1-\varepsilon)$. It is easy to see that solving the problem $\frac{\partial E^q}{\partial a_i^{(n)}} = 0$ is equivalent to finding such an ε for which the equality

$$\left| \varepsilon^2(1-\varepsilon) \right| = 0 \tag{4.48}$$

holds true. The equality has two solutions, see the Fig. 4.6. The first, $\varepsilon = 0$, is the expected solution of the optimization problem. However, the second valid solution $\varepsilon = 1$ does not coincide with finding the optimal weights. This property may lead not only to a slow convergence, but the optimization process may fail completely – especially in a case, where only a simple first-order optimization method (i.e. gradient descent) is used.

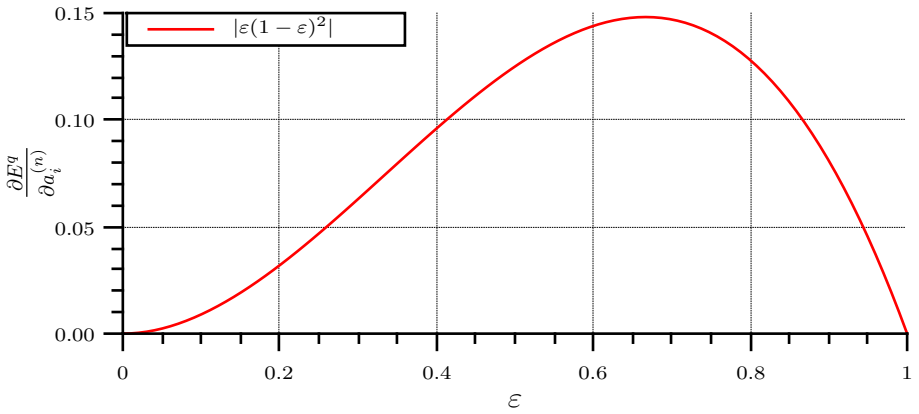


Figure 4.6: The shape of the derivative of the combination MSE and cross-entropy

The combination of softmax error function and the MSE criterion yields

$$\begin{aligned}\frac{\partial E^q}{\partial a_i^{(n)}} &= \sum_j \frac{\partial E^n}{\partial y_j^{(n)}} \frac{\partial y_j^{(n)}}{\partial a_i^{(n)}} \\ &= \sum_j \left(y_j^{(n)} - t_j\right) \left(y_i^{(n)} \delta_{ij} - y_i^{(n)} y_j^{(n)}\right) \\ &= \sum_j y_i^{(n)} \delta_{ij} \left(y_j^{(n)} - t_j\right) - \sum_j y_i^{(n)} y_j^{(n)} \left(y_j^{(n)} - t_j\right) \\ &= y_i^{(n)} \left(y_i^{(n)} - t_i\right) + y_i^{(n)} \sum_j y_j^{(n)} \left(y_j^{(n)} - t_j\right),\end{aligned} \tag{4.49}$$

and finally, the combination of sigmoid transfer function and the cross-entropy criterion, when used on a two class problem, yields

$$\begin{aligned}\frac{\partial E^q}{\partial a_i^{(n)}} &= \sum_j \frac{\partial E^n}{\partial y_j^{(n)}} \frac{\partial y_j^{(n)}}{\partial a_i^{(n)}} \\ &= \sum_j \frac{t_j}{y_j^{(n)}} y_j^{(n)} \left(1 - y_j^{(n)}\right) \\ &= t_i \left(1 - y_i^{(n)}\right) + y_i^{(n)} \left(1 - t_i\right) = y_i^{(n)} - t_i,\end{aligned} \tag{4.50}$$

which is the same result as for the combination of linear transfer function and the MSE criterion or softmax transfer function and the cross-entropy criterion. The choice of the sigmoid transfer function in combination with the cross-entropy criterion is not sensible for problems with more than two classes, though. The minimum of the error surface can be reached easily, irrespective of the value of t_j, by simply setting the output $y_j^{(n)} = 1$. This means that a neural network producing constantly $y_j^{(n)} = 1$ without any respect to the input vector poses a mathematically correct solution that is, however, completely useless for real applications.

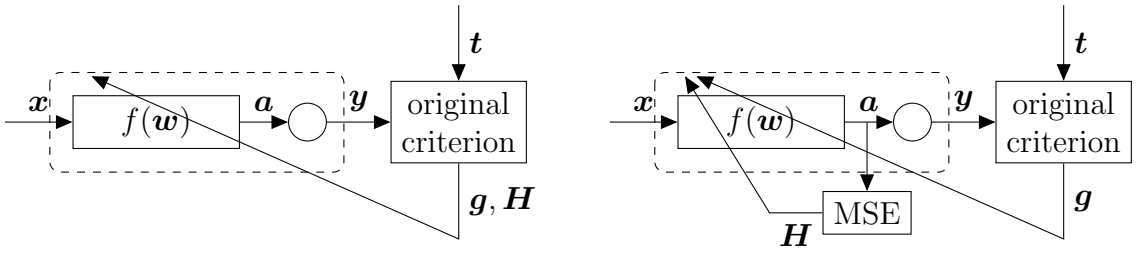


Figure 4.7: An example of search in companion space, adopted from [32]

As has been shown, there exist a special pairing between the type of the last layer transfer function and the error function. Choosing one of these pairs leads to faster converging optimization process with low chance of converging to a degenerate solution.

Moreover, it has been proven theoretically as well as practically that this pairing property can be used for a development of novel efficient training algorithms [32]. In these algorithms, the optimization is performed on so called *companion error surface* that is implicitly described using the original criterion gradient $\boldsymbol{g}$ and using the Hessian matrix $\boldsymbol{H}$ obtained using a different criterion, exploiting the aforementioned pairing property. The shape of the hybrid error function then allows for much faster convergence of the optimization process or the optimization process is computationally less expensive, see Fig. 4.7 for an illustration of such process.

4.6.5 Incremental, Batch and Bunch Mode Training

As has already been mentioned, the total gradient for all training samples is gathered from a complete set of training samples, i.e.

$$\frac{\partial E}{\partial w_{ij}} = \sum_q \frac{\partial E^q}{\partial w_{ij}} \tag{4.51}$$

and the weights are then updated using this gradient. This approach is called the *offline training* or *batch mode training*. There are, however, different strategies, which differ in the amount of the training samples that are processed to accumulate the gradient that is used consequently for the weights update [109].

- Batch mode training – the process gathers gradient using all the training samples. The batch mode training is sometimes referred to as the *epoch learning*.

  The training process can be done as follows.

  while the network is not trained sufficiently **do**
  Using the actual set of weights, compute the gradient $\frac{\partial E}{\partial w_{ij}} = \sum_q \frac{\partial E^q}{\partial w_{ij}}$,
  Using $\frac{\partial E}{\partial w_{ij}}$, compute the weight update ΔW
  Using ΔW, compute the new weights
  Update the weights
  end while

- Incremental mode training – the training process performs weight update after evaluation of each training sample. The training process converges faster, however it is computationally more expensive and the parallelization of the computation is problematic.

  The training process can be done as follows.

  while the network is not trained sufficiently **do**
    Randomize the training set
    for each training vector q from the training set **do**
      Using the current set of weights, compute the gradient $\frac{\partial E^q}{\partial w_{ij}}$;
      Using $\frac{\partial E^q}{\partial w_{ij}}$, compute the weight update ΔW
      Using ΔW, compute the new weights W
      Update the weights
    end for
  end while

- Bunch mode training – a compromise between the batch and the incremental approaches. The training set is split (usually randomly) into a set of B bunches (sets) such that $\sum_{b=1}^{B} q_b = Q$, where Q is the total number of training samples, q_b is the number of training samples of the b-th bunch (hundreds or thousands of samples), and the gradient is gathered only for samples in the current bunch. Then, the weight update is performed and the gradient computation is started on the next bunch.

  The training process can be done as follows.

  while the network is not trained sufficiently **do**
    Partition the training vectors randomly into B sets (bunches)
    for each bunch b from the set of B bunches **do**
      Using the current set of weights, compute the gradient $\frac{\partial E_b}{\partial w_{ij}} = \sum_{q_b} \frac{\partial E^q}{\partial w_{ij}}$;
      Using $\frac{\partial E_b}{\partial w_{ij}}$, compute the weight update ΔW
      Using ΔW, compute the new weights W
      Update the weights
    end for
  end while

The training methods using the conventional numerical optimization approaches usually require the batch mode gradient evaluation. The reason for this is that the classical optimization methods accelerate the convergence by exploiting gradients from several consecutive steps together with the dynamics of the gradient changes. If the gradients are computed from different data, this information is no longer useful nor valid.

For the incremental mode training, the *stochastic* methods are usually used. The gradient computed using only a limited portion of samples can be conceptualized as a gradient that was noise-corrupted. The only assumption is that the intensity of the noise is sufficiently small and therefore the noise-corrupted gradient points approximately the

correct direction[3]. In general, the stochastic methods do not converge to minimum. This has two consequences. First, the stopping criterion must be based on different metrics than the value of the error function. Second, to allow the training process to slowly set to a local minimum it is necessary to slowly reduce the influence of the newly computed gradients. Usually, some kind of gradual decrease of the learning rate is practiced. This approach is similar to *simulated annealing*. Moreover, because of the intrinsic training with noise, the chance of overtraining phenomenon is limited[13].

4.6.6 Probabilistic Interpretation of Network Outputs

Since the introduction of neural networks, a lot of attention ([78, 46, 15, 23, 90]) has been paid to conditions on possibility of interpretation of the outputs as probabilities or likelihoods. This is a challenging task because of the interaction between the output layer units and the training algorithm.

Membership Problem, Independent Classes Consider a task of assigning several probabilities of a different Boolean event (have, doesn't have) to a input vector. In other words, the output vector $\boldsymbol{z}$ is expected to be

$$\boldsymbol{z} = [P(c_1|\boldsymbol{x}), \ldots, P(c_N|\boldsymbol{x})] \tag{4.52}$$

Since the elements of the vector $\boldsymbol{z}$ are independent, we can concentrate on a single output unit without the loss of generality. Using Bayes' theorem, the posterior probability $P(c_1|\boldsymbol{x})$ can be written as

$$P(c_1|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|c_1)P(c_1)}{P(\boldsymbol{x}|c_1)P(c_1) + (1 - P(\boldsymbol{x}|c_1))\,(1 - P(c_1))}, \tag{4.53}$$

which can be written as

$$P(c_1|\boldsymbol{x}) = \frac{1}{1 + \exp(-a)}, \tag{4.54}$$

where the substitution

$$a = \ln \frac{P(\boldsymbol{x}|c_1)P(c_1)}{(1 - P(\boldsymbol{x}|c_1))\,(P(c_1))} \tag{4.55}$$

was used, in which a is the input into the output layer. It can be shown ([12]) that if a follows a distribution from the family of exponential distributions parametrized by $\boldsymbol{\varphi}$ and $\boldsymbol{\phi}$

$$a \sim \exp\left\{A(\boldsymbol{\varphi}) + B(\boldsymbol{q}, \boldsymbol{\phi}) + \boldsymbol{\varphi}_k^{\mathrm{T}} \boldsymbol{z}\right\}, \tag{4.56}$$

then the vector $\boldsymbol{z}$ can be interpreted as a posterior probability.

[3]The definitions of "sufficiently small" and "approximately correct" depend on the chosen method.

Membership Problem, Tied Classes In situations, where the output

$$\boldsymbol{z} = [P(c_1|\boldsymbol{x}), \ldots, P(c_N|\boldsymbol{x})]\,, \tag{4.57}$$

must fulfill an additional condition

$$\sum_{n=1}^{N} P(c_n|\boldsymbol{x}) = 1, \tag{4.58}$$

the previous approach is unusable, because the sigmoid activation function guarantees only $0 \leq g(a) \leq 1$ but not the unity sum. Given a set of unconstrained values a_j, both conditions (limitation and unity) can be ensured using the softmax activation function. Moreover, if the cross-entropy is used for training, the neural network can be interpreted as the *Maximum Likelihood* estimator of the probabilities[90].

4.7 Conclusion

In this chapter, the neural networks were introduced. A special attention has been paid to neural networks applied in speech recognition tasks. In this field, the recurrent and feedforward topologies are used prevalently.

Moreover, the most common training algorithm based on the backpropagation of the error was derived. Because the straightforward use of the steepest descent optimization algorithm often leads to slow convergence, alternative optimization algorithms, usually yielding much better convergence, were briefly introduced.

5 Training of the Speech Recognition Systems

The current speech recognition systems are usually constructed to be *speaker independent* (SI). The notion of speaker independence in the context of speech recognition systems is used to emphasize the fact that the recognition accuracy does not depend on knowledge of speaker's individual voice characteristics. In lesser degree, it represents the insensitivity to changes in the acoustical environment.

The speaker independence is usually achieved by means of estimating the parameters on huge amount of acoustic data collected from a wide variety of speakers. The currently used speech corpora contain hundreds or thousands of hours ([20]) of acoustic data and hundreds of speakers. Using such amount of data is beneficial for several reasons. Even if the amount of per-speaker data is small, the overall amount of data is sufficient to estimate the statistics robustly. However, it is a known fact that for a given speaker, the performance of a SI system is inferior to the *speaker dependent* (SD) model.

5.1 Speaker Normalization

The goal of speaker normalization techniques is to obtain a sequence of observation vectors with the speaker-dependent information removed. Usually, the methods are applied during the signal parametrization phase and are based on our understanding of human physiology.

5.1.1 Cepstral Mean Normalization (CMN)

The Cepstral Mean Normalization (or Cepstral Mean Substraction) is a common normalization technique. The original motivation was to rectify the convolutional distortion by means of removing the constant (time-invariant) influence of the recording channel. However, it can can be used for the speaker normalization as well. Together with the normalization of the recording channel transfer function, it also removes the constant (time-invariant) portion of the individual's vocal tract transfer function.

The algorithm of the CMN computes a long-term mean of the observation vectors. The observation vectors mean is then used to zero-mean normalize the observation vectors. The computation is usually performed in a per-sentence or a per-segment fashion. For the application of the CMN in real-time systems, the mean value is usually computed on-the-fly using an IIR (Infinite Impulse Response) filter providing a running average of the cepstral coefficients. The time constant (i.e. the forgetting factor) of the IIR filter

must be tuned a-priori to allow for accommodation of changes of speaker/environment and yet to be able to provide sufficiently robust statistics.

Additionally, the Cepstral Variance Normalization (CVN) is often used in robust speech recognizers. The interpretation of the CVN is not as straightforward as the interpretation of the CMN. From the statistical point of view, the variance normalization reduces the sample variability and improves the coherence amongst the distribution functions of features extracted from individual speaker-specific data sets.

While the CMN application mostly brings satisfactory improvements, the situation about the CVN is not as simple. When the operating conditions are not very diverse, the CVN application usually results in a slight decrease of the recognition accuracy.

5.1.2 Statistical Moments Normalization

The motivation for statistical moments normalization follows the statistical reasoning that reduction of the mismatches between statistical properties (moments) gathered on the speaker-specific data correlates with the reduction of the amount of the speaker-dependent information. Besides the mean and the variance normalization, higher moments normalization has been experimented with. It is necessary to differentiate between the odd and the even moments. The N-th moment is computed as

$$E\left[O^N\right] \triangleq \frac{1}{T}\sum_k \left[o_k\right]^N \tag{5.1}$$

and the purpose of the normalization is to obtain a new sequence of observation vectors $\tilde{O}_{[N]}$, for which

$$E\left[\tilde{O}^N_{[N]}\right] = 0, \tag{5.2}$$

if N is odd, and

$$E\left[\tilde{O}^N_{[N]}\right] = M_N, \tag{5.3}$$

when N is even. In the previous equation, the value M_N is the N-th moment of the Gaussian distribution $\mathcal{N}(0,1)$. The N-th even moment for the Gaussian distribution is defined as

$$M_N = \frac{(N-1)!}{k!2^k}, \quad \text{with} \quad k = \frac{N-1}{2}. \tag{5.4}$$

Experiments published in [112] suggest that with dropping signal-to-noise ratio (SNR), even the $N = 3$ and $N = 5$ moments improve the robustness of the recognition system. For the moments higher than $N = 5$, no evidence of positive influence has been found. On the other hand, the authors of [53] performed similar experiments with the higher degree statistical moments normalization on the AURORA system and they report consistent improvements across all SNR ranges and for all tested types of noise.

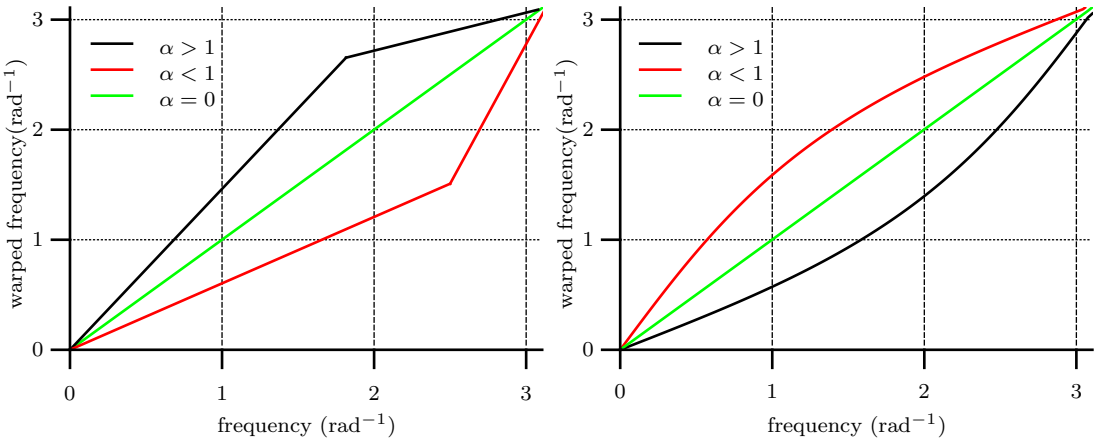


Figure 5.1: An example of non-linear warping functions: Piecewise linear (left) and Bilinear (right)

5.1.3 Vocal Tract Normalization (VTN)

As has been said, the sources of speaker variability are diverse. The general agreement is that one of the significant sources of the variability is the variability in the length of speakers' vocal tract. The length of the vocal tract varies significantly amongst speakers (from 13 cm in case of adult women to more than 18 cm in case of adult men). The length affects the resonance frequencies of the vocal tract, i.e. the locations of formant frequencies. The formant frequencies are crucial for distinguishing between vowels.

Therefore, a lot of attention has been paid to the development of techniques suitable for the *vocal tract normalization* (VTN) or the *vocal tract length normalization* (VTLN).

According to [119], the relationship between the vocal tract length L_{VT} and the i-th formant frequency F_i is approximately

$$L_{VT} \approx \frac{(2i-1)c}{4F_i}, \tag{5.5}$$

where c is the speed of sound. Therefore, by means of manipulation with frequency spectra, the inter-speaker discrepancies caused by different lengths of the vocal tract can be reduced. The transform function $\nu_\alpha(\omega)$ is required to have specific properties. When shifting the spectra, the domain and co-domain are required to retain the complete information available in the original spectrum and not to change the frequency range. In other words, the domain and co-domain of the function $\nu_\alpha(\omega)$ must have the same range, $\nu_\alpha(\omega) : (0, \omega_{\text{mez}}) \rightarrow (0, \omega_{\text{mez}})$. Moreover, the function is usually assumed to be strictly monotonic and continuous. Such a function is referred to as a *warping function*. Most often, one of the following warping functions is used.

Piecewise Linear Warping Function The Piecewise Linear Warping Function is defined as

$$\nu_\alpha(\omega) = \begin{cases} \alpha\omega & \text{for} \quad 0 \leq \omega \leq \omega_0 \\ \alpha\omega + \frac{\pi - \alpha\omega_0}{\pi - \omega_0}(\omega - \omega_0) & \text{for} \quad \omega_0 < \omega \leq \omega_{mez}, \end{cases} \tag{5.6}$$

where ω_0 is the "break point" frequency, where the steepness of the function changes. Usually, ω_0 is chosen so that for the third format frequency w_{F_3} holds $w_{F_3} < w_0$ for the majority of the population. The *warping coefficient* α is determined during the process of speaker normalization. Usually, $0.88 \leq \alpha \leq 1.12$.

Bilinear Warping Function The Bilinear Warping Function[2] is a smooth version of the piecewise linear warping function. It is defined as

$$\nu_\alpha(\omega) = \omega + 2\arctan\left(\frac{(1-\alpha)\sin\omega}{1-(1-\alpha)\cos(\omega)}\right), \tag{5.7}$$

where α is again the *warping factor* and usually is of the same range as in the case of the piecewise linear warping function.

The warping factor is then used during the signal analysis. Sometimes, when possible, instead of transforming the frequency spectrum of the signal, the filterbank filters coefficients are transformed. This approach is of significantly lower computational complexity, since the coefficients of the filters are transformed only once, during the initialization of the filterbanks.

The estimation of warping factors is usually done in the maximum likelihood fashion. Either the likelihood of the recognized utterance is maximized repeatedly to find out the value of the factor α maximizing the utterance's likelihood (which can be time consuming task) or a special simplified approach is used. In the simplified approach, instead of the acoustic model used for recognition, a relatively simple GMM model used solely for the task of warping factor computation is used. In both these approaches, because of high computational complexity, usually only a relatively small number of predefined values of α are evaluated and the resulting α is chosen from this set.

5.2 Acoustic Model Adaptation

While the task of the speaker normalization is to remove the speaker-dependent information from the observation vectors, the task of *model space adaptation* is somewhat different. The process of adaptation modifies the parameters of the model to improve the fit(match) of the model on the given speaker. In other words, the adaptation converts the speaker independent model into a speaker dependent model. As has been said, the speaker dependent model usually performs significantly better than the speaker independent models. The most common approach to adaptation in the context of GMM/HMM framework is the *maximum a posteriori*(MAP) adaptation or *maximum likelihood linear regression* (MLLR) adaptation.

Maximum A-Posteriori (MAP) Adaptation The MAP approach to adaptation strives to maximize the posterior probability of an utterance. This approach assumes that an informative prior exists. In the case of a GMM/HMM system adaptation, the role of the informative prior is played by the speaker-independent model. Let's consider the case, when the prior is not informative (or is not informative enough). In the case when the prior is not informative, the MAP approach degenerates to ML approach. This is not a problem when the amount of data is huge, but one of the premises of adaptation is the small amount of adaptation data.

Maximum Likelihood Linear Regression (MLLR) Adaptation The MLLR approach tries to circumvent the main problem of the MAP adaptation – requirements of a relatively large amount of data. Since the amount of the data available for the speaker adaptation is usually limited, a considerable amount of research effort has been invested into development of less data-hungry adaptation methods.

The key observation behind this method is that the most important parameters of the GMM/HMM system are the parameters of GMM components – the vectors of mean values and the covariance matrices. The MLLR method tries to find a linear transform, which maximizes the likelihood that the adaptation data are generated by the given acoustic model.

Moreover, the MLLR method uses pooling (clustering) to limit the amount of free parameters. The transformation parameters are then shared amongst all the members of the given cluster as opposed to the MAP approach, where the parameters are determined for every mixture component individually.

For mixtures in one pool (cluster), a parameters set $(\boldsymbol{A}, \boldsymbol{Q}, \boldsymbol{b})$ is computed and all the mixtures in the given pool are then transformed using the following formulae

$$\begin{aligned} \tilde{\boldsymbol{\mu}} &= \boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{b} \\ \tilde{\boldsymbol{\Sigma}} &= \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Q}^{\mathrm{T}}, \end{aligned} \tag{5.8}$$

where $\boldsymbol{\mu}$ and $\tilde{\boldsymbol{\mu}}$ are the original (unadapted) and adapted mean vector respectively, similarly $\boldsymbol{\Sigma}$ and $\tilde{\boldsymbol{\Sigma}}$ are the original and adapted covariance matrix respectively. From the engineering point of view the adaptation of covariance matrix is not of great importance, because the reported improvement is usually reported to be less than 2 % relatively ([54]).

The use of the hierarchical clustering approach referred to as a *regression tree* can be advantageous for the grouping of the mixtures components. Leafs of the regression tree represent the elementary classes, i.e. mixture components. The topmost root of the tree represents the global class of all possible components of all mixtures. Each inner node in the tree represents a class that results from merging the child nodes/groups together.

During the adaptation, the tree is walked in the direction from the root to leafs and for node the amount of data available for the given node is evaluated. The node is split, if the amount of the data is sufficient and all the resulting child nodes have a sufficient amount of data. If the node is a leaf or if the split would produce a child node with insufficient amount of data the node is marked as *closed* and the traversal is finished there. In the end, the adaptation parameters are computed on the obtained groups.

The benefit is in the guarantee of a sufficient amount of data for the clusters obtained during the process. Moreover, during the on-line adaptation, the transforms can be refined when enough data is gathered to allow further splitting of clusters.

Constrained Maximum Likelihood Linear Regression (CMLLR) is (as the name suggests) a special case of the MLLR. However, the CMLLR has very special properties that deserve to be discussed in more detail. The special feature of the CMLLR ([25]) is that the same matrix is used for adaptation of both means and covariances, i.e. for one mixture component the adapted parameters are

$$\begin{aligned} \tilde{\boldsymbol{\mu}} &= \boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{b} \\ \tilde{\boldsymbol{\Sigma}} &= \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^{\mathrm{T}}, \end{aligned} \tag{5.9}$$

where $\boldsymbol{\mu}$ and $\tilde{\boldsymbol{\mu}}$ are the original (unadapted) and the adapted mean vector respectively, and similarly $\boldsymbol{\Sigma}$ and $\tilde{\boldsymbol{\Sigma}}$ are the original and the adapted covariance matrix respectively and the adaptation matrix $\boldsymbol{A}$ and the adaptation vector $\boldsymbol{b}$ are to be determined during the adaptation process.

The interesting property of the CMLLR is that in the case where only one set of matrices has been computed (i.e. the parameters were computed for the root of the regression tree), the transform can be applied either on the model parameters (as described) or at the feature level as

$$\tilde{o}_i = \boldsymbol{A}^{-1} o_i + \boldsymbol{A}^{-1}\boldsymbol{b}, \tag{5.10}$$

where o_i is the i-th observation vector. In that case the CMLLR can be referred to as the FMLLR (feature-level MLLR).

The possibility of transforming the features instead of the model means that FMLLR can be seen as a speaker normalization procedure or as an acoustic model adaptation procedure. Moreover, the paper [85] proved that in the case of using cepstral features, the FMLLR is equivalent to VTLN. The equivalence between these two approaches holds generally; it does not depend on the warping function choice or the type of the cepstral representation.

5.3 Speaker Adaptive Training

As has been said, a significant performance gap between the speaker-specific (speaker-dependent) acoustic models and speaker-independent acoustic models exists. The reason is that the SI model must account for the inter-speaker variances during training. Adding more parameters (increasing the model size) helps to lower the difference, but only to a limited extent. Because of higher variability in the statistical properties of features, the clusters of features belonging to different speech units overlap more, resulting in the impossibility to discriminate correctly between the overlapped clusters.

Both the speaker adaptation and the acoustic model adaptation approaches aim to improve the coherence between the SI model and the speaker. The important thing is

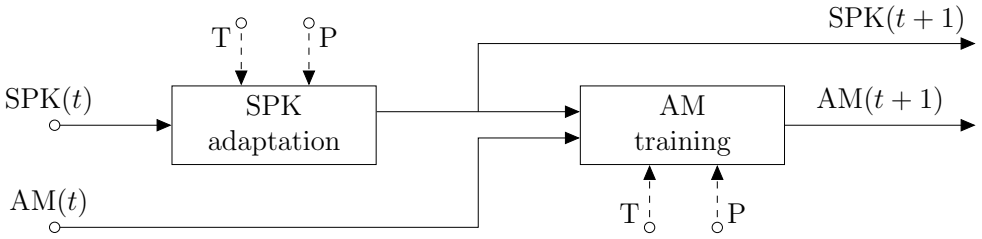


Figure 5.2: A diagram of a single SAT epoch

that this remedy is done only before or during the recognition phase, the training phase produces a common SI model.

In [4], an alternative approach, called *speaker adaptive training* (SAT) was proposed. The SAT approach introduces the removal of the speaker-induced discrepancies during the training phase. The assumption is that, if the inter-speaker variability is reduced and the homogenized data are used during training instead of the original data, the resulting acoustic model will perform better. The model obtained using the SAT approach is called the *SAT model* or the *canonical model* to help differentiate it from SI or SD models.

The process of SAT is iterative. Each iteration/epoch has two consecutive stages.

1. Use the present model AM(t) (obtained in the previous iteration) to estimate the speaker-dependent normalization parameters SPK($t + 1$).

2. Use the speaker-dependent normalization parameters SPK($t + 1$) estimated in the previous step to reduce the inter-speaker mismatch and train a new acoustic model AM($t + 1$).

Usually, the SI model is trained first of all. Then several epochs of SAT are performed. The SI model is used as the starting model in the first iteration of SAT. It has been observed ([73]) that 3–6 epochs of SAT are sufficient. The diagram of one training iteration is depicted in the Fig. 5.2. In the figure, the symbol T denotes the referential text and P denotes the training feature vectors.

The recognition phase in the SAT recognition system closely follows the recognition phase of a speech recognition system employing speaker adaptation. The only exception is that instead the SI trained acoustic model, the SAT trained acoustic model is used for speaker adaptation. The enrollment (i.e. introduction of a new speaker) can be either *supervised* or *unsupervised*.

Supervised Enrollment In this case, the speaker's utterances are available together with the correct transcript. The reference transcript T is used to calculate the speaker adaptation parameters.

Unsupervised Enrollment In this case, the speaker's utterance is available, however the reference transcription is not. Usually, the SI model is used to recognize the utterance and the resulting recognized text is then used in place of the reference text. Since no

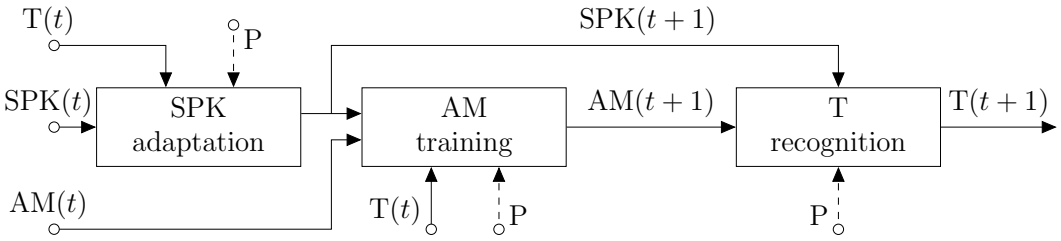


Figure 5.3: Unsupervised (twopass) Speaker Adaptive Recognition scheme

recognizer works with 100% accuracy, some words may be recognized incorrectly. Luckily, in cases when sufficient amount of data is available, this poses no serious problem, because the bias of the incorrectly recognized words will be compensated by the amount of the correctly recognized words. What amount of data is sufficient depends on the accuracy of SI recognizer and the number of parameters to be estimated during the speaker normalization phase. The diagram of one training iteration is depicted in the Fig. 5.3. In the figure, the symbol $T(t)$ denotes the recognized text transcript obtained using the acoustic model $AM(t)$ and P denotes the training feature vectors.

Another approach is to evaluate a confidence factor (CF) for each of the recognized words and then use these factors to eliminate the words identified to be recognized incorrectly (e.g. [3], [41] and many others). The possible hitch of this approach is that the confidence factors are essentially computed by means of evaluating the fit of the recognized word on the underlaying acoustic observation. The incorrectly recognized words will fit less than the correctly words and therefore will receive lower CF. However, a low CF ranking can be assigned to some correctly recognized words as well. In the context of speaker adaptation, the more profound speaker-specific information the acoustic observations bear, the smaller will be the match between the recognized word and the observation sequence. Therefore, by removing the words with low CF ranking, the words bearing the speaker-specific information will be removed as well. In an extreme case, the speaker adaptation mechanism may be rendered completely inoperative.

5.4 Conclusion

This chapter discussed the problems arising during training of speech recognition systems, especially the problems connected with inter-speaker variability.

The natural variations between speakers increase inner variability of the data, which usually results in a more complex model (a portion of the model parameters is necessary to compensate for the speaker-induced data variability) and/or possibly worse recognition accuracy, because the inner variability lowers the separability of the classes (phonemes, context dependent phones).

The techniques of speaker adaptation and speaker normalization (in combination with SAT) aim to reduce this variability and thus to produce a less complex, yet often a better performing acoustic model.

6 Current Approaches to Adaptation of a Neural Network

In this chapter, the current approaches to speaker adaptation or speaker normalization are described and discussed. Not all methods presented here were developed specifically for the speaker normalization or adaptation. The methods will be discussed from the point of suitability for speaker normalization or adaptation.

6.1 Retraining of the Network

The most popular and very simple approach is to retrain the SI neural network on the speaker specific data. During the process of the training, all parameters of the SI NN are adapted. Because of the large number of parameters being adapted and a relatively small amount of data, attention must be paid to avoid overtraining. This is usually done by early-stopping, where a portion of the adaptation data is used only for cross-validation of the training criterion ([77]). It is vital not to use the cross-validation portion of the data for training directly, thus the total amount of speaker-specific data available to train on, is reduced furthermore.

Another disadvantage of this approach is that no speaker-specific information is isolated. The result of the training is a new standalone speaker-adapted network [16].

Moreover, the retraining approach has another disadvantage. This disadvantage is tied to the stability/plasticity dilemma and is usually referred to as *catastrophical forgetting*.

6.1.1 Catastrophical Forgetting

Ideally, the ANN's internal representation of the problem should be plastic enough to allow for adaptation in the presence of an additional knowledge (training vectors) and, at the same time, stable enough to retain the important bits of the original knowledge, even during the adaptation. Quite obviously, these demands are in contradiction. This problem is of high importance in the field of speaker adaptation, because during the adaptation, only a limited amount of data is available and, furthermore, the original training data is usually not present.

It should be noted that this problem is not specific to neural networks, but it exhibits here more noticeably, because of the discriminative nature of the training. The direct consequence of the restricted data amount is that some target classes presented originally in the training data are not present in the adaptation data. The associated nodes are then forced to have zero value for all training pairs, which leads to a corresponding

change of the associated weights. The change is directed towards producing zero output no matter what input is presented and therefore, the original knowledge is lost (forgotten) during the adaptation.

This can be contrasted to a case when GMM models trained in the ML-sense are adapted. In that case, the units with little or no associated data are either usually simply left unadapted or a more general transform is determined for a cluster of acoustically similar units by virtue of considering the units as a one (more general) unit. The size of the cluster is chosen in such way that provides enough training material for computation of the adaptation transform.

6.1.2 Rehearsal and Pseudo-rehearsal Techniques

A significant amount of attention has been paid to development of approaches enabling alleviation of the above mentioned problem. One group of the proposed techniques is referred to as *rehearsal techniques*[89]. Although the rehearsal training techniques were developed more specifically for inclusion of new knowledge into neural networks, they can be used directly for adaptation as well. These techniques rely on the presence of a substantial amount of the training data being present even during the adaptation phase. The adaptation is then performed sequentially. In each step, a training pair is chosen and used for the so-called rehearsal training and when the network is trained (i.e. produces correct output for this training pair), the training pair is included into the training set. The rehearsal training itself is based on the well-known backpropagation technique. During the training phase, the single pair from the adaptation set is complemented with a small subset of training pairs drawn from the training set. The way the training pairs are drawn from the training set is the main differentiating feature between the methods belonging to the group of rehearsal training.

Although the rehearsal techniques make possible to learn new knowledge while retaining the old, the fact that access to the training data must be kept even during the adaptation is inconvenient and in many cases this necessity renders the rehearsal techniques unapplicable. A similar technique, called *pseudo-rehearsal*, has been devised[93] that mitigates the need for access to the training data.

Pseudo-rehearsal avoids the use of the training data by constructing a population of pseudo-training data (called pseudo-items) dynamically. Pseudo-items are generated as follows. First, an input feature vector $\boldsymbol{x}_k$ is generated randomly. Then, the input feature vector is forwarded through the neural network and an output vector $\boldsymbol{o}_k$ is obtained as

$$\boldsymbol{o}_k = \mathrm{ANN}(\boldsymbol{x}_k), \tag{6.1}$$

where the symbol $\mathrm{ANN}(\cdot)$ is used to denote the forward pass of the vector $\boldsymbol{x}_k$ through the neural network. The resulting pair $(\boldsymbol{x}_k, \boldsymbol{o}_k)$ is then used as a real "training" pair $(\boldsymbol{x}_k, \boldsymbol{t}_k)$.

In [94], both rehearsal and pseudo-rehearsal techniques are analyzed from the point of view of function approximation. The process of learning the neural network can be formalized as a process of fitting a function to the training data. The rehearsal mechanism during adaptation then leads to incorporation of the new data points into the set instead

of finding a new fitting function. The pseudo-rehearsal mechanism can be interpreted in the same way, with the exception that the constraints are enforced by means of sampling randomly the original function instead of sampling randomly the training data.

6.1.3 Conservative Training

The conservative training[39, 40] is an alternative approach to rehearsal and pseudo-rehearsal techniques. Instead of keeping the access to the training data or generating random vectors, it tries to alleviate the forgetting phenomenon by avoidance to enforcing the zero value with the unseen target classes. The idea is very straightforward. Let F_p be the set of units present as targets in the training set and let F_m be the set of units not present as targets in the training set. First, for the k-th vector $\boldsymbol{x}_k$ from the adaptation set, the output value $\boldsymbol{o}_k$ is generated as

$$\boldsymbol{o}_k = \mathrm{ANN}(\boldsymbol{x}_k). \tag{6.2}$$

Then, the vector $\hat{\boldsymbol{t}}_k$ is computed using the output $\boldsymbol{o}_k$ and the correct output $\boldsymbol{t}_k$ as

$$\hat{t}_{kj} = \begin{cases} o_{kj} & \text{if} \quad j \in F_m, \\ 1 - \sum_{l \in F_m} o_{kl} & \text{if} \quad \mathrm{correct}(j, \boldsymbol{t}_k), \\ 0 & \text{else}, \end{cases} \tag{6.3}$$

where o_{kj} and $\hat{t}_{kj}$ are the j-the elements of the vectors $\boldsymbol{o}_k$ and $\hat{\boldsymbol{t}}_k$ respectively, and $\mathrm{correct}(j, \boldsymbol{t}_k)$ is a predicate that holds only when j refers to the correct classification of the vector $\boldsymbol{x}_k$.

6.1.4 Partial Retraining

Despite the aforementioned drawbacks, the retraining approach is very attractive and it brings significant improvements when performed correctly[77]. Another possible approach to facilitation of the use of the retraining approach is introduced in [108]. The key idea is to adapt only a subset of weights.

The authors argue that the weights that will benefit most from adaptation can be identified by the activity of the corresponding hidden layer units. Moreover, they propose that the activity can be measured by the variance of the hidden layer outputs. Two possible approaches to the choice of the units for adaptation exist. Either only a predefined number of units with the highest variance is chosen or all the units whose output variance will reach over a certain threshold are chosen.

In this approach, the measure of suitability is of critical importance and should be explored more thoroughly. One possible extension is to identify the important weights directly. For this approach, the *saliency* measures used for network pruning can be used[66, 47]. The network pruning is a family of techniques used to reduce computational complexity and to improve the generalization of the network by means of selective deletion of low-importance weights. For partial network retraining, the saliency measure could be used to select the most important weights instead of the least important weights.

6.2 One Step Hessian Manipulation

The One Step Hessian Manipulation (OSHM) approach was introduced in [10] as a way how to adapt a neural network in one step. Given the fact that the SI-network weights are already minimizing the criterion function, the OSHM method uses the inverse of the Hessian matrix to compute the optimal weights of the speaker-specific network.

Suppose the network has been trained to minimize the error E on the training set $\Psi = \{(\boldsymbol{x}_0, \boldsymbol{t}_0), \ldots, (\boldsymbol{x}_{T-1}, \boldsymbol{t}_{T-1})\}$, where T is the number of training pairs. The task of the OSHM is to find a new set of weights $\tilde{\boldsymbol{W}}$ that minimizes the error $\tilde{E}$ corresponding to the adaptation set $\tilde{\Psi} = \{(\tilde{\boldsymbol{x}}_0, \tilde{\boldsymbol{t}}_0), \ldots, (\tilde{\boldsymbol{x}}_{T-1}, \tilde{\boldsymbol{t}}_{T-1})\}$, where the k-th training pair is given as

$$\tilde{x}_{nk} = x_{nk} + \Delta x_{nk} \qquad 0 \leq n < K_0 \tag{6.4}$$

$$\tilde{t}_{mk} = t_{mk} + \Delta t_{mk} \qquad 0 \leq m < K_{L-1}, \tag{6.5}$$

where Δx_{nk} and Δt_{mk} are small known shifts in the input and in the output respectively for the k-th training vector and K_0 and K_{L-1} are dimensions of the input and of the output layer respectively. Suppose that the change of a weight $w_{ij}^{(r)}$, i.e. a weight between the neurons i and j located in the r-th layer can be written as

$$\tilde{w}_{ij}^{(r)} = w_{ij}^{(r)} + \Delta w_{ij}^{(r)}. \tag{6.6}$$

Then the set of adapted weights $\tilde{\boldsymbol{W}}$ can be determined by means of determining $\Delta w_{ij}^{(r)}$ for every valid combination of r, i and j. With some bookkeeping (e.g. properly chosen numbering of neurons), the r index can be omitted.

First, the error function for the q-th pair E^q is Taylor expanded in Δx_{nk}, Δt_{mk} and Δw_{ij}. Using this approach, the weight change Δw_{ij} is obtained by means of solving an nonhomogeneous set of K linear equations, K being the total number of weights (and biases) in the neural network,

$$\sum_{ij} A_{ij,kl} \Delta w_{ij} = -\Delta T_{kl}, \tag{6.7}$$

where $A_{ij,kl}$ is an element of the Hessian matrix $\boldsymbol{A}$ defined as

$$A_{ij,kl} \equiv \sum_q \frac{\partial^2 E^q}{\partial w_{ij} \partial w_{kl}}, \tag{6.8}$$

and ΔT_{kl} is a substitution term evaluated as

$$\Delta T_{kl} \equiv \sum_q \sum_n \frac{\partial^2 E^q}{\partial x_{nq} \partial w_{kl}} \Delta x_{nq} + \sum_q \sum_m \frac{\partial^2 E^q}{\partial w_{kl} \partial t_{mq}} \Delta t_{mp}. \tag{6.9}$$

The Hessian matrix (or its inverse) can be evaluated using an algorithm similar to the backpropagation algorithm[11].

There are two main obstacles preventing the OSHM method to be used for adaptation in the field of speech recognition.

First, the Hessian scales with $O(K^2)$, K being the number of the ANN parameters. Since the networks used in the field of ASR have hundreds of thousands or millions of parameters, this approach is unfeasible. Moreover, to obtain a sufficiently robust estimate of the Hessian, a large amount of data is necessary. The complexity of the task can be reduced by exploiting the fact that the Hessian matrix is highly blocked when using the cross-entropy criterion[21].

Second, the fundamental assumption of the known pairing (i.e. the ability to compute the shift Δx_{nk} in the Eq. (6.4) and the shift Δt_{mk} in the Eq. (6.5) between the training and adaptation data is not easily fulfilled in the area of speech recognition.

6.3 Topology Manipulation

Some approaches took the inspiration in the Cascade correlation learning algorithm[31] and modify the internal structure of the network. The most common modifications include adding a new layer (either linear or a non-linear) and adding new units into the hidden layer of the network.

The positive aspect of these approaches is that the complexity of the adaptation task can be finely tuned taking the amount of the available adaptation data into account. The negative aspect is that the adaptation process usually changes the topology of the network being adapted, which can result in an increased computational complexity or may pose technical problems when incorporating these techniques into real-world software solutions.

6.3.1 Parallel Hidden Layer

In this approach, additional units are added into the hidden layer of the SI network. This process is conceptually equivalent to adding a new hidden layer $\hat{\boldsymbol{h}}$ into the network to work in parallel with the original hidden layer $\boldsymbol{h}$. In [77], this process was named Parallel Hidden Network (PHN), because a parallel hidden network sharing the input and the output layer is created. See the Fig. 6.1 for demonstration of this concept.

The weights of connections between the input $\boldsymbol{x}$ and $\hat{\boldsymbol{h}}$ and the weights between $\hat{\boldsymbol{h}}$ and the output layer $\boldsymbol{o}$ are to be determined by a common ANN training algorithm. During the adaptation process, the weights of the original SI ANN are held fixed and only the parameters of the PHN are modified. That means that the PHN is trained to compensate for the differences between the original SI system and the new speaker-specific system.

6.3.2 Linear Adaptation Layer

This technique modifies the topology of an HMM by inserting an additional linear layer either before the output layer[27, 97, 1] or before the hidden layer[39, 40]. The linear layer is then trained using any common neural network training algorithm.

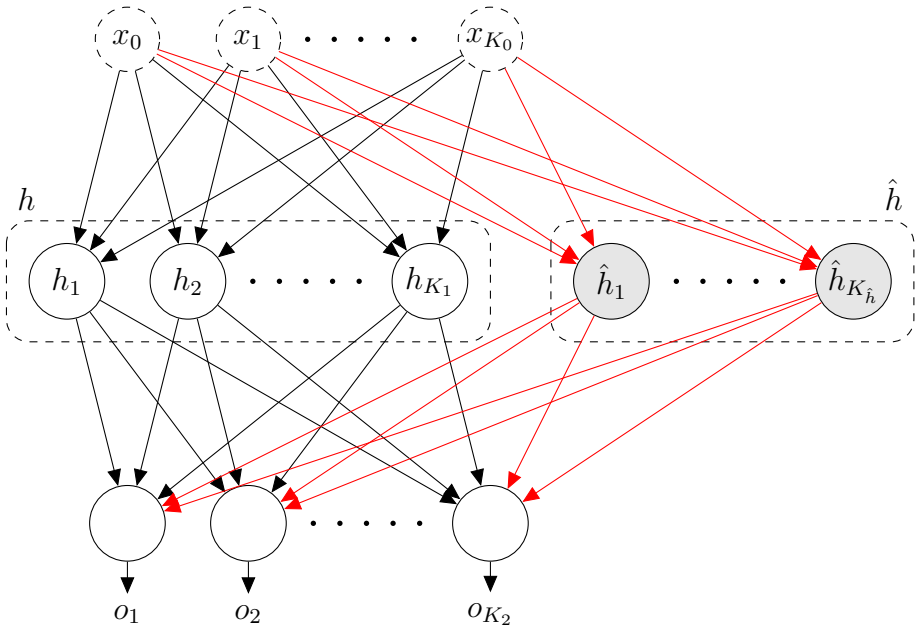


Figure 6.1: Parallel Hidden Network Adaptation

The reason, why a linear adaptation layer is used prevalently, is that when the weight matrix $\tilde{\boldsymbol{W}}$ is determined, the linear layer can be merged with the consecutive (non-linear) layer using the Eq. (4.9). Therefore, the computational and memory requirements stay constant, which is a favorable property.

In the field of hybrid speech recognition, the input to ANN usually consists of several adjacent frames of short-term low dimensional features[70, 81]. These adjacent frames form together an input vector of length n, $n = d \times k$, where d is the dimensionality of the short-term feature vector and k is the number of consecutive features. The input adaptation layer then captures the intra-frames dependencies as well as the inter-frames dependencies. Sometimes, as a way to mitigate the adaptation data sparsity problem, only the intra-frames dependencies are estimated. Instead of estimation of an $n \times n$ adaptation matrix $\tilde{\boldsymbol{W}}_A$,

$$\tilde{\boldsymbol{W}}_A = \begin{bmatrix} \tilde{w}_{11} & \cdots & \tilde{w}_{1n} \\ \vdots & \ddots & \vdots \\ \tilde{w}_{n1} & \cdots & \tilde{w}_{nn} \end{bmatrix}, \tag{6.10}$$

which requires n^2 coefficients to be estimated, the matrix $\tilde{\boldsymbol{W}}_A$ is estimated as a block-diagonal matrix

$$\tilde{\boldsymbol{W}}_A = \begin{bmatrix} \tilde{\boldsymbol{W}}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \tilde{\boldsymbol{W}}_k \end{bmatrix}, \tag{6.11}$$

where the $d \times d$ matrices $\tilde{\boldsymbol{W}}_1, \ldots, \tilde{\boldsymbol{W}}_k$ model the intra-frames dependencies in the individual short-term feature vectors. In this case, only $d^2 k$ coefficients has to be estimated.

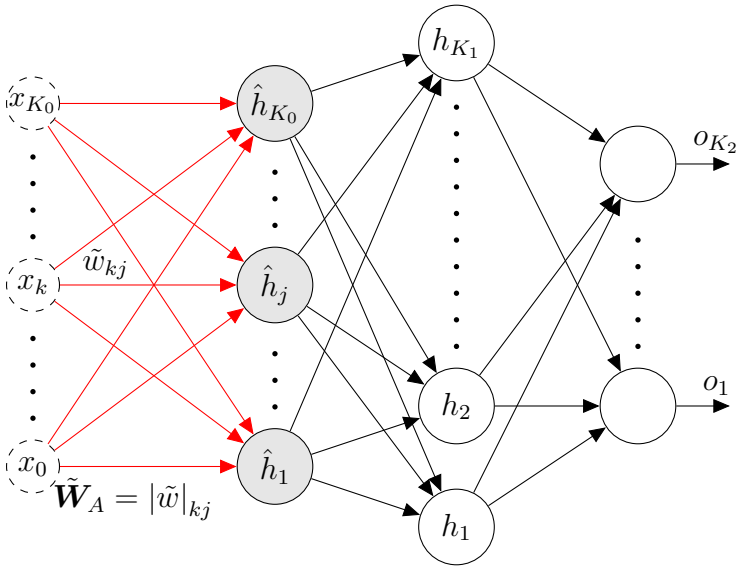


Figure 6.2: Linear Layer Network Adaptation

The number of coefficients to be estimated can be reduced further by assuming that the intra-frames dependencies do not vary through the time and the context. In that case, the matrices $\tilde{\boldsymbol{W}}_1, \ldots, \tilde{\boldsymbol{W}}_k$ are the same and equal to an $d \times d$ matrix $\tilde{\boldsymbol{W}}_a$. The adaptation matrix $\tilde{\boldsymbol{W}}_A$ can be expressed as

$$\tilde{\boldsymbol{W}}_A = \begin{bmatrix} \tilde{\boldsymbol{W}}_a & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \tilde{\boldsymbol{W}}_a \end{bmatrix}, \tag{6.12}$$

which means that only d^2 coefficients has to be estimated. In the last case, the short-term features can be transformed instead of modifying the weights of the SI network, which is a situation similar to the CMLLR/FMLLR dualism[36].

The CMLLR parameters estimated for adaptation of an GMM/HMM or a hybrid HMM system could be directly applied to the observed short-term feature vectors and used in the same context as the matrix $\tilde{\boldsymbol{W}}_a$. Experimental results have shown[70] that even if the CMLLR transforms are not estimated directly for the neural network, they actually have the speaker-normalizing effect.

6.3.3 Weights Interpolation

In [97], an alternative technique inspired by the MAP criterion used commonly for adaptation of GMM/HMM systems was introduced. The Weights Interpolation adaptation approach tries to mitigate the problems of the insufficient amount of data by exploiting the informative prior, for example the SI network.

In some situations, there is enough data to estimate a new channel/environment normalization matrix robustly, but the amount of speaker specific data is not sufficient

to estimate the speaker specific normalization matrix. Such a situation can occur, for example, in cases where there is a rather large set of speakers sharing the same environment and the same audio channel; however the amount of the per-speaker recordings is small.

Let's denote the weights of an SI network as $\boldsymbol{W}^{SI}$, the weights of the environment adapted network as $\boldsymbol{W}^{EA}$ and the weights of the speaker adapted network as $\boldsymbol{W}^{SA}$. The $\boldsymbol{W}^{EA}$ and $\boldsymbol{W}^{SA}$ weights were obtained as

$$\boldsymbol{W}^{EA} = \boldsymbol{W}^{SI} + \Delta\boldsymbol{W}^{EA} \tag{6.13}$$

and

$$\boldsymbol{W}^{SA} = \boldsymbol{W}^{SI} + \Delta\boldsymbol{W}^{SA} \tag{6.14}$$

respectively, where the matrices $\Delta\boldsymbol{W}^{EA}$ and $\Delta\boldsymbol{W}^{SA}$ are the channel/environment characterization and the speaker characterization coefficients respectively. Because the coefficients of the $\boldsymbol{W}^{SA}$ matrix can be estimated unreliably and the network with weights $\boldsymbol{W}^{EA}$ performance is (supposedly) suboptimal, a new weight-interpolated weights $\boldsymbol{W}^{AD}$ are obtained as

$$\boldsymbol{W}^{AD} = \boldsymbol{W}^{SI} + (1-\kappa)\Delta\boldsymbol{W}^{EA} + \kappa\Delta\boldsymbol{W}^{SA}, \tag{6.15}$$

which can be rewritten using the Eq. (6.13) and the Eq. (6.14) as

$$\boldsymbol{W}^{AD} = (1-\kappa)\boldsymbol{W}^{EA} + \kappa\boldsymbol{W}^{SA}. \tag{6.16}$$

The mixing parameter κ must be determined beforehand and depends on the amount of the speaker-specific adaptation data (i.e. on the reliability of the matrices $\boldsymbol{W}^{SA}$). In general, the cross-validation can be used to choose from a set of several predefined values of κ[97, 70].

6.4 Eigenvoices Adaptation

The eigenvoice paradigm was inspired by the progress in the field of digital image processing, especially in the field of face recognition. The eigenfaces paradigm is based on the observation that the dimensionality of the problem of face recognition is much smaller than the dimensionality of the 2D images containing the image[60, 114]. As a useful approximation, it can be assumed that every individual face is a linear combination of a certain small amount of face components called *eigenfaces*. During the training phase, the individual primitive components are determined. During the recognition phase, the weighting factors associated with the individual components are determined. Because the number of the eigenfaces is small, the number of the corresponding weighting factors is low. Therefore, even a very low amount of adaptation data is sufficient.

The process of isolation of the eigenvoices (similarly to isolation of the eigenfaces) is started with a set of SD models, let the number of SD models be T. The coefficients of

each SD model are written as a single long vector[1]. These long vectors are often referred to as *supervectors*.

A set of T supervectors is then processed by a dimensionality reduction technique, PCA or ICA or SVD is among the most used. Because of large dimensionality of the supervectors, it is usually used a method that allows to isolate only the first K most important components. Usually, first 10–50 components is sufficient[64, 63].

Each new SD model $\tilde{\boldsymbol{W}}$ can be estimated as[27]

$$\tilde{\boldsymbol{W}} = \bar{\boldsymbol{W}} + \boldsymbol{\Phi} \times \boldsymbol{\Gamma}, \tag{6.17}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K]$ is the eigenvectors matrix consisting of K retained components (eigenvoices), $\boldsymbol{\Gamma} = [\gamma_1, \ldots, \gamma_K]^{\mathrm{T}}$ is the vector of mixing coefficient describing the individual speaker and $\bar{\boldsymbol{W}}$ is the per-speaker mean of the adaptation parameters. The previous formula can be written alternatively to emphasize the relations for a single element of $\tilde{\boldsymbol{W}}$ as

$$\tilde{w}_k = \bar{w}_k + \sum_s \phi_{ks}\gamma_s, \tag{6.18}$$

where $\tilde{w}_k$ and $\bar{w}_k$ are the k-th elements of supervectors $\tilde{\boldsymbol{W}}$ and $\bar{\boldsymbol{W}}$ respectively and ϕ_{ks} is the k-th element of the vector $\boldsymbol{\phi}_s$.

The task of adaptation process is then to determine the K-dimensional vector $\boldsymbol{\Gamma}$. In the case of ANN parameters, gradient descent can be used. The gradients can be calculated using an approach similar to backpropagation. Using the augmented form from the Eq. (4.2), the activation potential of an MLP-ANN can be written as

$$a = \hat{\boldsymbol{w}}^{\mathrm{T}} \cdot \hat{\boldsymbol{x}} = \sum_r \hat{w}_r \hat{x}_r, \tag{6.19}$$

where $\hat{x}_r$ and $\hat{w}_r$ are the r-th elements of $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{w}}$ respectively. Substituting the Eq. (6.18) into the Eq. (4.2) yields

$$a = \sum_r \left(\bar{w}_r + \sum_s \phi_{rs}\gamma_s \right) \hat{x}_r \tag{6.20}$$

$$= \bar{\boldsymbol{w}}^{\mathrm{T}} \cdot \hat{\boldsymbol{x}} + \sum_r \sum_s \phi_{rs}\gamma_s \hat{x}_r \tag{6.21}$$

$$= \bar{\boldsymbol{w}}^{\mathrm{T}} \cdot \hat{\boldsymbol{x}} + \sum_s \gamma_s \left(\sum_r \phi_{rs}\hat{x}_r \right). \tag{6.22}$$

The gradient $\frac{\partial E}{\partial \gamma_k}$ can be evaluated using

$$\frac{\partial E}{\partial \gamma_k} = \sum_j \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial \gamma_k}, \tag{6.23}$$

[1]The ordering of the coefficients is arbitrary, however the order must be kept the same for all the SD models

where $\frac{\partial E}{\partial a_j}$ can be determined by the usual error backpropagation and $\frac{\partial a_j}{\partial \gamma_k}$ can be evaluated using the Eq. (6.22) as

$$\frac{\partial a_j}{\partial \gamma_k} = \sum_r \phi_{rk} \hat{x}_r \tag{6.24}$$

Because the weights of the new speakers SD network are obtained as a linear combination of the eigenvoices vectors, the vectors can be considered as a basis vector for the eigenvoice space. Knowing that, it is easy to realize that the starting set of T SD vectors should be big enough to generate sufficient covering of all speaker variations.

6.5 Special and Hybrid Paradigms

6.5.1 Speaker Morphing

In [55], an alternative to the common model transform approach was explored. Let $\mathcal{X}^{(a)} = \boldsymbol{x}_0^{(a)}, \boldsymbol{x}_1^{(a)}, \ldots, \boldsymbol{x}_{t-1}^{(a)}$ be a sequence of acoustic observations of a speaker a. The goal of speaker morphing is to find a transformation function $\mathcal{F}_a(\cdot)$ such that $\mathcal{F}_a\left(\mathcal{X}^{(a)}\right)$ will approximate the observation sequence $\mathcal{X}^{(r)}$ produced by an reference speaker. In mathematical terms,

$$\mathcal{F}_a^\star(\cdot) = \arg\min_{\mathcal{F}(\cdot)} \sum_i E\left(\mathcal{F}\left(\boldsymbol{x}_i^{(a)}\right), \boldsymbol{x}_i^{(r)}\right), \tag{6.25}$$

where $E(\cdot,\cdot)$ is an pair-wise error function. In [55], the MSE function was used and the approximating function $\mathcal{F}_a^\star(\cdot)$ was a feed-forward neural network of a fixed topology. The search for the optimal mapping function was done via the optimization of the weights of the neural network through backpropagation.

The drawback of this approach is its assumption on existence of paired training data. This means not only a parallel corpus must be developed, but also an unique mapping between the individual feature vectors must be found, which is task of the same complexity as the task of searching for the function $\mathcal{F}_a^\star(\cdot)$.

6.5.2 Special Architectures

Another approach to dealing with speaker normalization or speaker adaptation task is to extend the topology of the network in such way that will either speaker-normalize the data implicitly or will allow for more sophisticated (and possibly more powerful) speaker normalization.

Superstructures and Mixtures of Experts

The Meta-Pi Network[45] is a modular classifier that allows for a fusion of heterogeneous sources of information via a "combinational superstructure". As authors proved experimentally, the network can be trained in such way to perform speaker normalization implicitly.

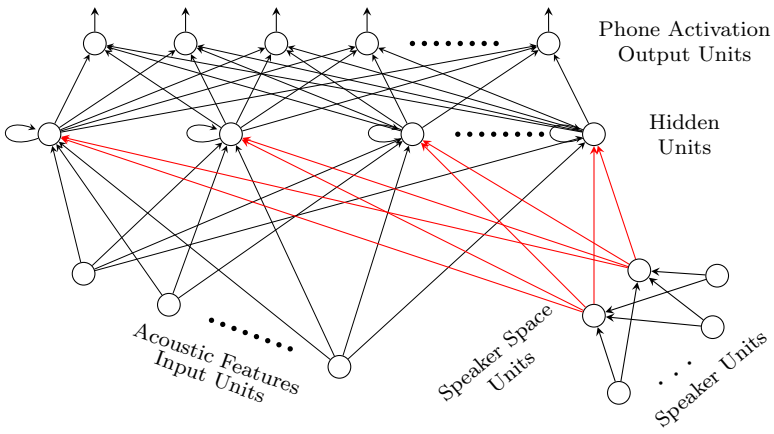


Figure 6.3: A Speaker Sensitive Network topology

Second Order Units

In [121], second order units were used in the network. Second order units activation potential is computed as

$$a_k^{(n)} = \sum_{i,j} \hat{w}_{ijk} y_i^{(n-1)} y_j^{(n-1)} + \sum_m \hat{w}_{mk} y_m^{(n-1)}, \tag{6.26}$$

where $y_i^{(n-1)}$, $y_j^{(n-1)}$ and $y_m^{(n-1)}$ are outputs of i-th, j-th and m-th neuron from the previous layer of the network and $\hat{w}_{ijk}$ and $\hat{w}_{mk}$ are the associated augmented mixing weights.

A layer with second order units can be conceptualized as a layer consisting from first-order units whose weights can be changed dynamically even during the recognition phase. The authors argue that this allows for estimation of a canonical form of the input vectors. A properly trained network is therefore able to perform the speaker normalization implicitly, without the necessity to supply the speaker information.

Speaker Sensitive Network

An approach targeted directly at hybrid speech recognition systems was introduced in [111]. The basic idea is to train a *speaker sensitive* network, which is an SI recurrent network, in which the speaker variations are decoupled by design from the rest of the network (see the Fig. 6.3).

The speaker variation modeling is achieved by adding speaker-space units and speaker units. One speaker unit per speaker is added during the training phase. The activity of the speaker unit is +1, when the current training pair belongs to the speaker and 0 otherwise. Because of this, the activation values of the speaker-space units are functions of the weights associated to the connection between the given speaker-space unit and the active speaker unit. Therefore, every (even unknown) speaker can be described using a vector of speaker sensitive factors $\boldsymbol{\lambda}$ containing the activation values of the Speaker Space Units.

During the recognition phase, the speaker sensitive factors $\boldsymbol{\lambda}$ can be determined dynamically by a joint maximization of likelihood of the current recognition lattice,

$$W^{\star} = \arg\max_{W} \left\{ P(W) \int_{\Lambda} p(\boldsymbol{o}|W, \boldsymbol{\lambda}) p(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \right\}, \tag{6.27}$$

where Λ is the space of possible speaker parameters $\boldsymbol{\lambda}$, $\boldsymbol{o}$ is the sequence of observations and W and $W^{\star}$ is the possible word sequence and the recognized word sequence respectively. This search can be optimized in such way that it's speed is comparable to a normal decoding technique.

6.5.3 Compensation of Trends during Training

During the training of an MLP-ANN, an implicit assumption of stationarity of the training data is made. However, in many cases, the training data can be product of an underlying slowly varying non-stationary process. Of course, one cause of this fact in the speech recognition field of study is that the process actually is inherently non-stationary. Nonetheless, the non-stationarity can be brought in by the environment the speaker is in or by the recording channel. During the training, this problem can be coped with to some extent by virtue of using sufficient amount of data. With a sufficient amount of data and enough free variables, the network will be able to discover the inner variability and to learn to account for it to some extent. The impacts therefore may be limited.

In the adaptation phase, however, this can pose a problem, because the sole necessity for the compensation of non-stationarities increases the need for data. Several algorithms were proposed to cope with the non-stationarity of the data [80, 82] that can reduce (to some extent) the inner variability of the data, which, in turn, leads to higher homogeneity of the training and adaptation set. The higher homogeneity in turn results in a lower amount of data being needed for robust estimation of parameters (either weights or adaptation parameters) of the network.

6.6 Conclusion

In this chapter, several techniques for adaptation of a neural network were presented and discussed. Some of them are applicable for various types of neural networks, such as multilayer perceptron network or recurrent networks, the applicability of others is limited to some particular topology or an additional assumption (for example on data stationarity) must be made.

In the field of hybrid speech recognition, the main issue that should be addressed by a successful speaker adaptation technique is the ability to control the number of free variables on basis of amount of the adaptation data available. Moreover, a successful adaptation technique should allow for some kind of incremental adaptation.

By the term incremental adaptation it is meant a situation, where a speaker adaptation was already performed, however more additional data was obtained thereafter. In that case, it should be possible to use the newly obtained data to "tune" or tweak the

adaptation parameters only on the basis of the new adaptation data without the access to the previous adaptation data.

In the next chapter, a novel adaptation method inspired by several of the techniques introduced and published in the peer-reviewed papers will be introduced.

7 Proposed Approach to Adaptation of a Neural Network

In this chapter, the proposed adaptation method will be introduced and analyzed. Because the adaptation method was developed to reflect the real experimental and production systems' needs, the main design features of the used Czech and English speech recognition systems will be presented in this chapter as well.

During the development of the method as well as during the work on this thesis, recognition systems for two languages were developed. The system for the Czech speech recognition was trained on a telephone-quality speech corpora SpeechDat(E)[18] (8 kHz, 16 bits per sample), the system for the English speech recognition was trained on desk-microphone quality corpora TIMIT [38] and WSJCAM0[96]. Both these corpora were recorded at 16 kHz, 16 bits per sample.

7.1 Description of the Experimental Systems

7.1.1 Feature Extraction

The acoustical features extraction system was adopted from [99]. The LTSP features (described in this thesis on the page 9) are the final product of this stage. The LTSP features extraction starts with the mel-filterbank analysis.

Let $\boldsymbol{x}(\tau)$, $\boldsymbol{x}(\tau) = [x_1(\tau), \ldots, x_Q(\tau),]^{\mathrm{T}}$ be a vector of mel-filterbank band energies extracted at the time instance τ. Usually, $Q = 15$ for an 8kHz signal or $Q = 23$ for a 16kHz signal. These coefficients are computed from 25 ms signal window with shift of 10 ms, which means that every 10 ms a new mel-filterbank log-energies vector is produced.

To capture the long-term temporal trajectories, a sliding window of $S = 2N + 1$ mel-filterbank outputs is processed to form a single LTSP feature vector each time. A common choice of S is $S = 31$, therefore $N = 15$.

A window of the S consecutive mel-filterbank outputs is split into two parts, the first one, referred to as the "left" half, includes the mel-filterbank outputs with indices $1, \ldots, N+1$ and the second part, referred to as "right" includes the mel-filterbank outputs with indices $N + 1, \ldots, S$. The mel-filterbank output with the index $N + 1$ in the middle of the window is used in both parts and represents the time reference, i.e. the actual position in the signal. That said, the vectors from the left part and vectors from the right part can be conceptualized as vectors from the past and from the future respectively. To emphasize this temporality, the sequence of the vectors in the window will be written as $\boldsymbol{x}(t-N), \ldots, \boldsymbol{x}(t), \ldots, \boldsymbol{x}(t+N)$. Please note that the timing information here is relative

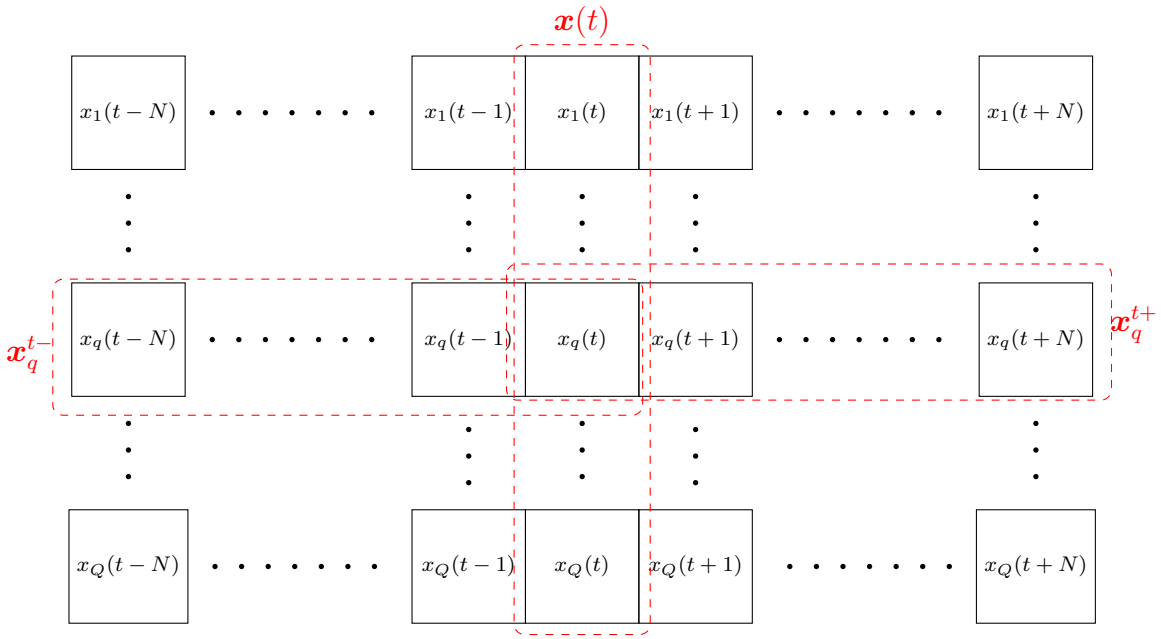


Figure 7.1: Scheme of a LTSP vector construction

to the given window.

The individual trajectories of each frequency band are then processed separately. The left and the right trajectory of the q-th frequency band $\boldsymbol{x}_q^{t-}$ and $\boldsymbol{x}_q^{t+}$ respectively can be formalized as

$$\boldsymbol{x}_q^{t-} = [x_q(t-N), \ldots, x_q(t)]^{\mathrm{T}} \tag{7.1}$$

$$\boldsymbol{x}_q^{t+} = [x_q(t), \ldots, x_q(t+N)]^{\mathrm{T}} \tag{7.2}$$

where $x_q(t-k)$ denotes the q-th element of the feature vector $\boldsymbol{x}(t-k)$, i.e. the log-energy output of q-th mel filter. Both the vectors $\boldsymbol{x}_q^{t-}$ and $\boldsymbol{x}_q^{t+}$ are then weighted by the respective half of the hamming window $w(n)$ defined as

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{2N}\right), \qquad n = 0, 1, \ldots, 2N, \tag{7.3}$$

where for the left trajectory, the index n runs from 0 to N and for the right trajectory, the index n runs from N to $2N$. The hamming-weighted vectors are then processed using DCT[1] and the first D, $D = 11$, coefficients are retained. Because of the linearity of both the DCT and the windowing preprocessing, both these operation can be written as

$$\boldsymbol{d}_q^{t-} = \underset{w(n)}{\mathrm{DCT}}\left[\boldsymbol{x}_q^{t-}\right] \tag{7.4}$$

$$\boldsymbol{d}_q^{t+} = \underset{w(n)}{\mathrm{DCT}}\left[\boldsymbol{x}_q^{t+}\right], \tag{7.5}$$

[1]To be precise, it is actually the DCT-II normalized(orthonormal) transform.

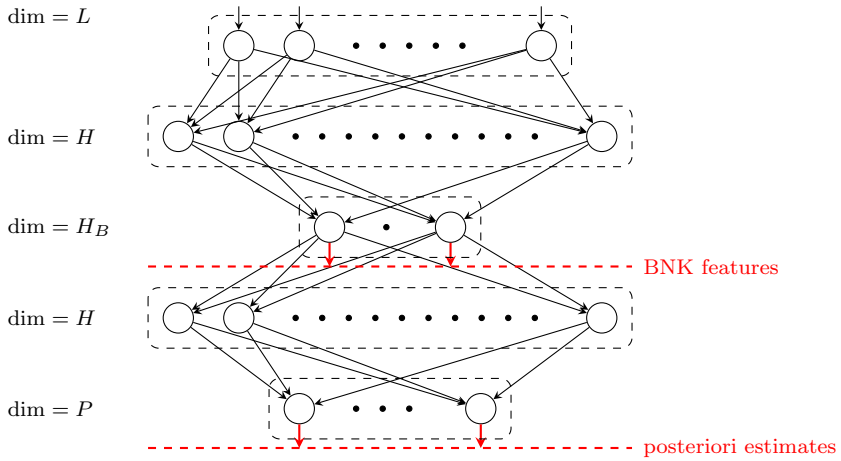


Figure 7.2: A scheme of two forward modes of a bottleneck neural network

where the operator $\mathrm{DCT}_{w(n)}$ denotes the DCT with windowing using the respective half of the window $w(n)$, $\boldsymbol{d}_q^{t-}$ and $\boldsymbol{d}_q^{t+}$ are vectors, each of length D, containing the DCT coefficients of the trajectory of the q-th frequency band.

The resulting coefficients vectors $\boldsymbol{d}_1^{t-}, \ldots, \boldsymbol{d}_Q^{t-}$ and $\boldsymbol{d}_1^{t+}, \ldots, \boldsymbol{d}_Q^{t+}$ are then concatenated together to form one LTSP vector $\boldsymbol{\chi}(t)$

$$\boldsymbol{\chi}(t) = \left[\boldsymbol{d}_1^{t-}, \ldots, \boldsymbol{d}_Q^{t-}, \boldsymbol{d}_1^{t+}, \ldots, \boldsymbol{d}_Q^{t+}\right]^{\mathrm{T}}. \tag{7.6}$$

The length M of the vector $\boldsymbol{\chi}(t)$ is $M = 2QD$. In the case of the 8kHz signal, in which case $K = 15$, this yields $M = 330$. In the case of the 16kHz signal with $K = 23$, the LTSP feature vectors are of length $M = 506$.

The LTSP features are then used for training a neural network. In the context of speech recognition, the neural network can be used either as a bottleneck features extractor or as a posteriori probabilities estimator.

7.1.2 Bottleneck Features Extractor

The neural network used as bottleneck features extractor has three hidden layers, each with the sigmoidal activation functions. The general topology is $M \times H \times H_B \times H \times P$, where M is the dimension of the LTSP feature vector, H is the dimension of hidden layer (in the experimental systems in this work, $H = 1500$), H_B is the dimension of the bottleneck feature vector (for historical reasons, $H_B = 36$ in the scope of this work) and P is the dimension of the output layer, which depends on the phonetic alphabet used during training. The network is trained to work as a posteriori probabilities estimator. For the production of the bottleneck features, the last two layers are removed, so that the H_B layer becomes the output layer.

7.1.3 Posteriori Probabilities Estimator

Two distinctive topologies of the networks used as posteriori probability estimators were tested. The first kind of network is of topology $M \times H \times P$, where M is the dimension of the LTSP feature vector, H is the dimension of the hidden layer (in the experimental systems in this work, $H = 1500$) and P is the dimension of output layer, which depends on the phonetic alphabet used during training.

However, training two separate networks — one for the bottleneck features extraction and the second one for the posteriori probabilities estimation – can quickly become a significant computational burden if not unfeasible completely. Because of that, the bottleneck topology was used even for posteriori probabilities estimation purposes. See the Fig. 7.2 for a demonstration of these two modes of operation.

7.2 Adaptation of Long Temporal Spectral Features

Given the feature extraction subsystem, the linear adaptation approach was adopted. Specifically, the method developed and presented in this work belongs to the group of methods adapting the first (input) layer of the network. The positive aspect of the linear adaptation methods is that they can be applied either on the weight matrix or, in cases when the weight matrix is fixed, on the input features. This duality resembles the duality of the FMLLR/CMLLR methods.

7.2.1 Linear Adaptation of the Weight Matrix

Suppose a neural network is defined as described in the section 4.3. The n-th layer of the network is a subject to a linear layer adaptation using a $K_n \times K_n$ matrix $\boldsymbol{\Gamma}$. A matrix notation can be used to write the n-th layer of the MLP-ANN as

$$\boldsymbol{a}_n = \boldsymbol{y}_{n-1}^{\mathrm{T}} \cdot \boldsymbol{\Gamma} \cdot \hat{\boldsymbol{W}}_n \tag{7.7}$$

$$\boldsymbol{y}_n = \boldsymbol{g}_n\left(\boldsymbol{a}_n\right), \tag{7.8}$$

where $\hat{\boldsymbol{W}}_n$ is the augmented form of the weight matrix containing the original weight matrix $\boldsymbol{W}_n$ as well as the bias vector $\boldsymbol{b}_n^{\mathrm{T}}$. As can be seen, only the Eq. (7.7) describing the activation potential differs, the Eq. (7.8) defining the unit activation is unaffected by the adaptation process.

Written in the scalar form, for an k-th element $a_k^{(n)}$ of the vector $\boldsymbol{a}_n$ it holds

$$a_k^{(n)} = \sum_p y_p^{(n-1)} \tilde{w}_{pk}^{(n)}, \tag{7.9}$$

where $\tilde{w}_{pk}^{(n)}$ is the (p,k)-th element of the matrix $\tilde{\boldsymbol{W}}_n = \boldsymbol{\Gamma} \cdot \hat{\boldsymbol{W}}_n$, $\tilde{\boldsymbol{W}}_n = \left[\tilde{w}^{(n)}\right]_{pk}$ and can be obtained as

$$\tilde{w}_{pk}^{(n)} = \sum_j \Gamma_{pj} \hat{w}_{jk}^{(n)}, \tag{7.10}$$

where Γ_{pj} is the (p,j)-th element of the matrix $\mathbf{\Gamma}$ and $\hat{w}_{jk}^{(n)}$ is the (j,k)-th element of the matrix $\hat{\boldsymbol{W}}$.

The matrix $\mathbf{\Gamma}$ can be determined using the approach that is used during training of the neural network. Suppose an error function is defined that is to be minimized by a proper choice of the adaptation parameters. On other words, the matrix $\mathbf{\Gamma}$ can be obtained by solving the following equation

$$\frac{\partial E}{\partial \mathbf{\Gamma}} = 0, \tag{7.11}$$

where E is the error function to be minimized, which can be equivalently written as a set of D_n^2 equations

$$\frac{\partial E}{\partial \Gamma_{ij}} = 0, \quad \text{for all admissible pairs } i,j, \tag{7.12}$$

where Γ_{ij} is the (i,j)-th element of the matrix $\mathbf{\Gamma}$.

Using the chain rule, the quantity $\frac{\partial E}{\partial \Gamma_{ij}}$ can be rewritten as

$$\frac{\partial E}{\partial \Gamma_{ij}} = \sum_k \frac{\partial E}{\partial a_k^{(n)}} \frac{\partial a_k^{(n)}}{\partial \Gamma_{ij}}, \tag{7.13}$$

where the quantities $\frac{\partial E}{\partial a_k^{(n)}} = \sigma_k^{(n)}$ can be easily obtained using the formulae described on the page 29.

Using the formulae Eq. (7.9) and Eq. (7.10), the activation potential $a_k^{(n)}$ can be written as

$$a_k^{(n)} = \sum_p y_p^{(n-1)} \sum_q \Gamma_{pq} \hat{w}_{qk}^{(n)}, \tag{7.14}$$

so the quantity $\frac{\partial a_k^{(n)}}{\partial \Gamma_{ij}}$ can be written as

$$\frac{\partial a_k^{(n)}}{\partial \Gamma_{ij}} = y_i^{(n-1)} \hat{w}_{jk}^{(n)}. \tag{7.15}$$

Substituting the Eq. (7.15) back into the Eq. (7.13) leads to

$$\frac{\partial E}{\partial \Gamma_{ij}} = y_i^{(n-1)} \sum_k \hat{w}_{jk}^{(n)} \sigma_k^{(n)}, \tag{7.16}$$

Using the equations Eq. (7.16) and Eq. (4.25) and Eq. (4.28), all the elements $\frac{\partial E}{\partial \Gamma_{ij}}$ can be evaluated. Any iterative gradient algorithm for minimizing the set of equations defined by the Eq. (7.12) can be used. Usually, for practical reasons, the same method used for neural network training is used for the adaptation as well.

7.2.2 Minimum Error Linear Transform

The matrix $\mathbf{\Gamma}$ could be used for adaptation of the neural networks used in the experimental systems, however given the length M of the LTSP vector, which is either 330 or 506 in the case of the 8 kHz signal or 16 kHz signal respectively, the number of free variables is $330 \times 330 \approx 2^{17}$ or $506 \times 506 \approx 2^{18}$ in case of the 8kHz signal or 16kHz signal respectively. Estimation of such number of variables robustly needs a significant amount of data, which is, as has been already said, not the case in most adaptation scenarios.

It is possible, however, to assume that the matrix $\mathbf{\Gamma}$ has an inner structure, in other words that $\mathbf{\Gamma}$ is a function of a G-dimensional vector variable $\boldsymbol{\gamma}'$, $\boldsymbol{\gamma}' = [\gamma'_1, \dots \gamma'_G]$ such that

$$\mathbf{\Gamma} = \mathbf{\Gamma}\left(\boldsymbol{\gamma}'\right), \tag{7.17}$$

where presumably $G \ll M$. Assuming this, instead of optimizing the Eq. (7.12), the set of equations

$$\frac{\partial E}{\partial \gamma'_i} = 0 \qquad \text{for } 1 \leq i \leq G \tag{7.18}$$

must be solved. Moreover, because the error function is of the form

$$E(\mathbf{\Psi}|\boldsymbol{\gamma}') = \sum_q E^q, \tag{7.19}$$

the formula Eq. (7.18) can be rewritten as

$$\sum_{q=0}^{T-1} \frac{\partial E^q}{\partial \gamma'_i} = 0 \qquad \text{for } 1 \leq i \leq G, \tag{7.20}$$

where T is the number of adaptation samples. Using the chain rule, the expression $\frac{\partial E^q}{\partial \gamma'_i}$ can be expanded to

$$\frac{\partial E^q}{\partial \gamma'_i} = \sum_k \frac{\partial E^q}{\partial a_k^{(n)}} \frac{\partial a_k^{(n)}}{\partial \gamma'_i} \tag{7.21}$$

$$= \sum_k \frac{\partial E^q}{\partial a_k^{(n)}} \sum_{lm} \frac{\partial a_k^{(n)}}{\partial \Gamma_{lm}} \frac{\partial \Gamma_{lm}}{\partial \gamma'_i}, \tag{7.22}$$

where the quantities $\frac{\partial E}{\partial a_k^{(n)}} = \sigma_k^{(n)}$ and $\frac{\partial a_k^{(n)}}{\partial \Gamma_{lm}}$ can be determined using the equations Eq. (7.15) and Eq. (4.25).

The quantities $\frac{\partial \Gamma_{lm}}{\partial \gamma'_i}$ can be evaluated using the knowledge about the function $\mathbf{\Gamma}(\boldsymbol{\gamma}')$, that is about the structure of the matrix $\mathbf{\Gamma}$. Unfortunately, there is no general way or method that could be used. The problem of choice of the matrix structure involves a significant insight into the problem.

In the case of the LTSP parametrization, one of the possibilities is to relate the linear transform of the LTSP vector with the linear transform of the mel-filterbank outputs.

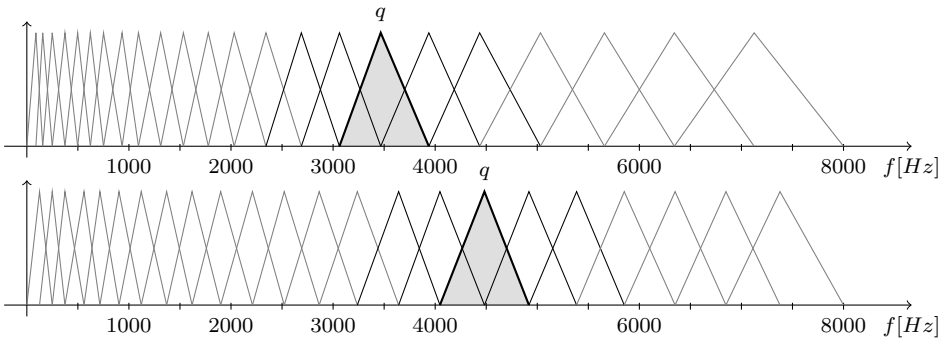


Figure 7.3: The shift of location of the q-th filter as a result of mel-filterbank outputs interpolation.

As has been already mentioned, the VTLN technique can be represented as a linear transform. Therefore, a general linear transform of mel-filterbank outputs can be used as a speaker normalization technique. Moreover, because of larger number of free variables, it is possibly more powerful then VTLN, where the shape of the matrix is controlled by only one free variable (the VTLN factor). See the Fig. 7.3 for a graphic illustration of this process.

Given the vector of the mel-filterbank output $\boldsymbol{x}(t) = [x_1(t), \ldots, x_Q(t),]^{\mathrm{T}}$, the transformed (speaker normalized) vector $\tilde{\boldsymbol{x}}(t)$ can be obtained as

$$\tilde{\boldsymbol{x}}(t) = \boldsymbol{\gamma}\boldsymbol{x}(t), \tag{7.23}$$

where $\boldsymbol{\gamma}$ is a normalization matrix. The duality the matrix $\boldsymbol{\gamma}$ in the Eq. (7.23) vs. the vector $\boldsymbol{\gamma}'$ in the Eq. (7.17) can be resolved without any loss of generality by assuming that the vector $\boldsymbol{\gamma}$ consists of elements of the matrix $\boldsymbol{\gamma}$, i.e. $\boldsymbol{\gamma} = [\gamma_{11}, \ldots, \gamma_{1Q}, \gamma_{21}, \ldots, \gamma_{kl}, \ldots, \gamma_{QQ}]^{\mathrm{T}}$, where the element γ_{kl} is the (k,l)-th element of the matrix $\boldsymbol{\gamma}$.

In the scalar form, the transform of a single element $\tilde{x}_i(t)$, which is the i-th element of the vector $\tilde{\boldsymbol{x}}(t)$ can be written as

$$\tilde{x}_i(t) = \sum_{j=1}^{Q} \gamma_{ij} x_j(t) = \boldsymbol{\gamma}_i \boldsymbol{x}(t), \tag{7.24}$$

where $x_j(t)$ is the j-th element of the original (unadapted) vector $\boldsymbol{x}(t)$ and $\boldsymbol{\gamma}_i$ is the i-th row of the matrix $\boldsymbol{\gamma}$.

The adapted left trajectories of the i-th band is then

$$\begin{aligned}\tilde{\boldsymbol{x}}_i^{t-} &= \left[\sum_{j=1}^{Q}\gamma_{ij}x_j(t-N)\ ,\quad \ldots\quad ,\ \sum_{j=1}^{Q}\gamma_{ij}x_j(t)\right]^{\mathrm{T}} \\ &= \sum_{j=1}^{Q}\gamma_{ij}\left[x_j(t-N)\ ,\quad \ldots\quad ,\ x_j(t)\right]^{\mathrm{T}} \\ &= \sum_{j=1}^{Q}\gamma_{ij}\boldsymbol{x}_j^{t-},\end{aligned} \tag{7.25}$$

that is, the normalized trajectory of the i-th filter output can be obtained as a weighted sum of the non-normalized trajectories. A similar result holds for the right trajectories. In that case,

$$\tilde{\boldsymbol{x}}_i^{t+} = \sum_{j=1}^{Q}\gamma_{ij}\boldsymbol{x}_j^{t+}. \tag{7.26}$$

The next step in the normalized LTSP feature vector construction is the DCT. Because of the linearity of the DCT, the DCT of a normalized vector is

$$\begin{aligned}\tilde{\boldsymbol{d}}_i^{t-} &= \underset{w(n)}{\mathrm{DCT}}\left[\tilde{\boldsymbol{x}}_i^{t-}\right] = \underset{w(n)}{\mathrm{DCT}}\left[\sum_{j=1}^{Q}\gamma_{ij}\boldsymbol{x}_j^{t-}\right] \\ &= \sum_{j=1}^{Q}\gamma_{ij}\left[\underset{w(n)}{\mathrm{DCT}}\,\boldsymbol{x}_j^{t-}\right] \\ &= \sum_{j=1}^{Q}\gamma_{ij}\boldsymbol{d}_j^{t-}\end{aligned} \tag{7.27}$$

that is, the normalized vector $\tilde{\boldsymbol{d}}_i^{t-}$ can be obtained as a weighted sum of the non-normalized vectors $\boldsymbol{d}_j^{t-}$. Similarly for the vector $\tilde{\boldsymbol{d}}_i^{t+}$.

The normalized LTSP vector $\tilde{\boldsymbol{\chi}}(t)$ consists of the normalized vectors $\tilde{\boldsymbol{d}}_{t-}(i)$ and $\tilde{\boldsymbol{d}}_{t+}(i)$ concatenated, i.e.

$$\begin{aligned}\tilde{\boldsymbol{\chi}}(t) &= \left[\tilde{\boldsymbol{d}}^{t-}(1),\ldots,\tilde{\boldsymbol{d}}^{t-}(Q),\tilde{\boldsymbol{d}}^{t+}(1),\ldots,\tilde{\boldsymbol{d}}^{t+}(Q)\right]^{\mathrm{T}} \\ &= \left[\sum_{j=1}^{Q}\gamma_{1j}\boldsymbol{d}_j^{t-},\ldots,\sum_{j=1}^{Q}\gamma_{Qj}\boldsymbol{d}_j^{t-},\sum_{j=1}^{Q}\gamma_{1j}\boldsymbol{d}_j^{t+},\ldots,\sum_{j=1}^{Q}\gamma_{Qj}\boldsymbol{d}_j^{t+}\right]^{\mathrm{T}} \\ &= \boldsymbol{\Gamma}\boldsymbol{\chi}(t),\end{aligned} \tag{7.28}$$

where the matrix $\mathbf{\Gamma}$ has the following shape

$$\mathbf{\Gamma} = \begin{bmatrix} \boldsymbol{\gamma}_{11} & \dots & \boldsymbol{\gamma}_{1Q} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\gamma}_{Q1} & \dots & \boldsymbol{\gamma}_{QQ} & 0 & \dots & 0 \\ 0 & \dots & 0 & \boldsymbol{\gamma}_{11} & \dots & \boldsymbol{\gamma}_{1Q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \boldsymbol{\gamma}_{Q1} & \dots & \boldsymbol{\gamma}_{QQ} \end{bmatrix}, \tag{7.29}$$

whose block elements $\boldsymbol{\gamma}_{ij}$ are diagonal $D \times D$ matrices

$$\boldsymbol{\gamma}_{ij} = \gamma_{ij} \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}. \tag{7.30}$$

The partial derivative $\frac{\partial \Gamma_{lm}}{\partial \gamma'_k} = \frac{\partial \Gamma_{lm}}{\partial \gamma_{ij}}$ is then

$$\frac{\partial \Gamma_{lm}}{\partial \gamma_{ij}} = \begin{cases} 1 & \text{if} \quad \lfloor l-1/D \rfloor = i-1 \\ & \text{and} \quad \lfloor m-1/D \rfloor = j-1 \\ & \text{and} \quad m \bmod D = l \bmod D \\ 1 & \text{if} \quad \lfloor l-1/D \rfloor = DQ+i-1 \\ & \text{and} \quad \lfloor m-1/D \rfloor = DQ+j-1 \\ & \text{and} \quad m \bmod D = l \bmod D \\ 0 & \text{else} \end{cases}, \tag{7.31}$$

which, in turn can be used[83] for simplification of the Eq. (7.22)

$$\frac{\partial E^q}{\partial \gamma_{ij}} = \sum_k \frac{\partial E^q}{\partial a_k^{(l)}} \sum_{lm} \frac{\partial a_k^{(l)}}{\partial \Gamma_{lm}} \frac{\partial \Gamma_{lm}}{\partial \gamma_{ij}} \tag{7.32}$$

$$= \sum_k \frac{\partial E^q}{\partial a_k^{(l)}} \operatorname{Tr}\left[\left[\frac{\partial a_k^{(l)}}{\partial \mathbf{\Gamma}} \right]^{\mathrm{T}} \frac{\partial \mathbf{\Gamma}}{\partial \gamma_{ij}} \right], \tag{7.33}$$

where

$$\frac{\partial a_k^{(l)}}{\partial \mathbf{\Gamma}} = \begin{pmatrix} \frac{\partial a_k^{(l)}}{\partial \Gamma_{11}} & \dots & \frac{\partial a_k^{(l)}}{\partial \Gamma_{1M}} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_k^{(l)}}{\partial \Gamma_{M1}} & \dots & \frac{\partial a_k^{(l)}}{\partial \Gamma_{MM}} \end{pmatrix} \tag{7.34}$$

and

$$\frac{\partial \mathbf{\Gamma}}{\partial \gamma_{ij}} = \begin{pmatrix} \frac{\partial \Gamma_{11}}{\partial \gamma_{ij}} & \dots & \frac{\partial \Gamma_{1M}}{\partial \gamma_{ij}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Gamma_{M1}}{\partial \gamma_{ij}} & \dots & \frac{\partial \Gamma_{MM}}{\partial \gamma_{ij}} \end{pmatrix} \tag{7.35}$$

7.2.3 Choice of the Error Function

The choice of the error function $E(\cdot)$ (eventually $E^q(\cdot)$) affects the statistical properties of the $\boldsymbol{\gamma}$ estimate. Usually, the same error function that was used for the SI network training is used during the adaptation phase. In the case, when the cross-entropy error function from the Eq. (4.19) is used, possibly in combination with the softmax activation function from the Eq. (4.14) in the output layer, the resulting estimator of $\boldsymbol{\gamma}$ has some attractive properties.

Suppose, a set of adaptation data Ψ is given and the cross-entropy error function is used for the speaker adaptation.

Let the training data be created by virtue of force-aligning a single utterance $\boldsymbol{\xi}$ represented by a sequence of the LTSP vectors $\boldsymbol{\chi}(0), \ldots, \boldsymbol{\chi}(T-1)$ and a reference transcript W. The force-alignment procedure determines a sequence $\boldsymbol{s}$, $\boldsymbol{s} = s_0, \ldots, s_{T-1}$, which is the most probable sequence of states the vocal tract went through while producing the utterance $\boldsymbol{\xi}$ given the reference transcript W.

The sequence $\boldsymbol{s}$ is then used to create the teacher information (or target) sequence $\boldsymbol{t}_0, \ldots, \boldsymbol{t}_{T-1}$ where each vector was constructed as follows. Given the "correct" state s_q in the time instance q, a hard-label[2] vector $\boldsymbol{t}_q$, $\boldsymbol{t}_q = [t_{q1}, \ldots, t_{qP}]^{\mathrm{T}}$ is created. For every element t_{qi}, $1 \leq i \leq P$, of the vector $\boldsymbol{t}_q$ holds

$$t_{qi} = \begin{cases} 1 & \text{if } i = s_q \\ 0 & \text{else.} \end{cases} \tag{7.36}$$

Assume now that the neural network is trained sufficiently, so that it can be assumed that the elements o_{qi} of the output vector $\boldsymbol{o}_q = [o_{q1}, \ldots, o_{qi}, \ldots, o_{qP}]^{\mathrm{T}}$ approximate the posteriori probability $p(i|\boldsymbol{\chi}(q))$, i.e. $o_{qi} = \tilde{p}(i|\boldsymbol{\chi}(q))$.

The error contribution E^q of the q-th observation vector $\boldsymbol{\chi}(q)$ will be

$$E^q = -\sum_r \lim_{t \to t_{qr}} t \ln \frac{o_{qr}}{t} = -\ln \tilde{p}(s_q|\boldsymbol{\chi}(q)), \tag{7.37}$$

where $\tilde{p}\,(s_q|\boldsymbol{\chi}(q))$ is the estimate of probability of the state s_q (i.e. the "correct" state) given the observation $\boldsymbol{\chi}(q)$ and the fact that the identity

$$\lim_{t \to 0+} t \ln \frac{o_{qr}}{t} = 0 \tag{7.38}$$

was used. The complete error E is then

$$E = \sum_q E^q = -\sum_q \ln \tilde{p}\,(s_q|\boldsymbol{\chi}(q))\,, \tag{7.39}$$

which is the log-likelihood function. Minimizing E with respect to $\boldsymbol{\gamma}$, i.e.

$$\boldsymbol{\gamma}^\star = \arg\min_{\boldsymbol{\gamma}} E(\Psi, \boldsymbol{\gamma}) = \arg\max_{\boldsymbol{\gamma}} \sum_q \ln \tilde{p}(s_q|\boldsymbol{\chi}(q), \boldsymbol{\gamma}), \tag{7.40}$$

[2] A hard-label is a technical term used to emphasize the fact that the vectors are created by 1-of-n coding and the position of the 1 value represents the most probable state.

means that in this case, the normalization parameters $\boldsymbol{\gamma}$ are estimated in such way that maximizes the likelihood of the seen utterance.

Maximum likelihoods estimator have several positive asymptotic properties, including consistency, asymptotic normality and efficiency[84].

7.3 Using the MELT Normalization

7.3.1 Selection of the Number of Free Variables

The structure of the mel-filterbank normalization matrix can be tweaked to choose lower or higher number of free variables. This is a very attractive property, when dealing with limited amount of data. Moreover, the number of free variables can be chosen on per-speaker basis.

As has been said, the multiplication of the mel-filterbank by a matrix $\boldsymbol{\gamma}$ can be interpreted as interpolation of the mel-filterbank.

Conceptually, the interpolation can be seen as shifting the triangle-shaped filter either to the left or to the right (which is exactly what is happening during the VTLN) and changing the gain (the "amplification") of the filters.

The control of the number of free variables is done through the number of diagonals of the matrix $\boldsymbol{\gamma}$. In the context of this work, the following convention will be used.

The number of diagonals is controlled via the MELT factor. Suppose the MELT factor is k, $0 \leq k \leq Q$, which will be written more conveniently as MELT $= k$, then the number of diagonals is equal to $1 + 2(k-1)$ if $k \geq 1$. The case when $k = 0$ will be discussed separately.

Now let's consider the case, when MELT $= 1$, i.e. when the $Q \times Q$ matrix $\boldsymbol{\gamma}$ is only a single diagonal matrix $\boldsymbol{\gamma}$,

$$\boldsymbol{\gamma} = \begin{bmatrix} \gamma_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \gamma_{QQ} \end{bmatrix}. \tag{7.41}$$

In that case, only the gain of the individual filters is controlled, no shift of the filter in the frequency range is done.

With increasing number of diagonals the number of free variables increases as well as the interpolation capability. For example, when MELT $= 2$, a tridiagonal matrix $\boldsymbol{\gamma}$,

$$\boldsymbol{\gamma} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & & & & \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \gamma_{Q-1\,Q-2} & \gamma_{Q-1\,Q-1} & \gamma_{Q-1\,Q} & \\ & & & \gamma_{Q\,Q-1} & \gamma_{QQ} & \end{bmatrix}, \tag{7.42}$$

is used for interpolation, the left and the right neighbors are used to determine the shift.

Now, let's consider the aforementioned boundary situation, when MELT $= 0$. In the context of the MELT framework, this situation may be, albeit somewhat forcefully,

defined as a transformation using a transformation matrix $\boldsymbol{\gamma}$ of the shape

$$\boldsymbol{\gamma} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}. \tag{7.43}$$

This is a very useful convention, albeit mainly from formal reasons. This allows us to include the situation, when no adaptation is actually performed, i.e. $\boldsymbol{x}'(k) = \boldsymbol{x}(k)$, in the MELT framework.

An incremental adaptation can be performed as well – additional diagonals can be added to an already existing matrix, when new speaker-specific data is acquired. The new data can be used to determine the coefficients of the newly added diagonals. Of course, this approach can be inferior to the approach when all the speaker-specific data are used to determine all needed coefficients, but sometimes, the previous data are not available.

7.3.2 Selection of the Normalization Locus

The MELT normalization technique establishes a link between the mel-filterbank transform and the input layer adaptation.

The speaker adapted system can be obtained in three fashions.

1. An speaker-adapted neural network can be obtained by replacing the original augmented input layer $\hat{\boldsymbol{W}}_1$ by the adapted layer $\tilde{\boldsymbol{W}}_1$ obtained as $\tilde{\boldsymbol{W}}_1 = \hat{\boldsymbol{W}}_1\boldsymbol{\Gamma}$. This results in virtually zero increase of the computational demands (only one matrix multiplication is needed during the SA recognition system startup)

2. Instead of the original mel-filterbank output $\boldsymbol{x}(t)$, the normalized filterbank output $\tilde{\boldsymbol{x}}(t)$ obtained as $\tilde{\boldsymbol{x}}(t) = \boldsymbol{\gamma}\boldsymbol{x}(t)$ can be used for LTSP feature vectors construction. This approach has moderate computational demands increase.

3. Instead of the original LTSP vector $\boldsymbol{\chi}(t)$, the normalized LTSP vector $\tilde{\boldsymbol{\chi}}(t)$ obtained as $\tilde{\boldsymbol{\chi}}(t) = \boldsymbol{\Gamma}\boldsymbol{\chi}(t)$ can be fed into the SI neural network. This approach has significant computational overhead and the previous two methods should be favored. In some real world applications, however, this can be the only way how to employ the speaker normalization.

7.3.3 The Adaptation Algorithm

The adaptation process is quite straightforward and the algorithm is close to the algorithm of the backpropagation used during the training of the ANN weights.

1. First, the MELT factor is chosen and the matrix $\boldsymbol{\gamma}$ is initialized. A suitable initialization is $\boldsymbol{\gamma} = \boldsymbol{I}$, $\boldsymbol{I}$ being the unity matrix.

2. The error function E is chosen.

3. The set of equations Eq. (7.18) or Eq. (7.20) is solved. For solving it, an iterative method is to be used.
   a) The speaker-specific feature vectors are propagated through the adapted ANN – see the previous subsection for info on possible ways how to perform the adaptation.
   b) For computation of quantities $\frac{\partial E^q}{\partial \gamma_{ij}} = \frac{\partial E^q}{\partial \gamma'_k}$, the Eq. (7.33) is used.
      i. In the Eq. (7.33), the elements $\frac{\partial a_k^{(n)}}{\partial \Gamma_{ij}}$ of the matrix $\frac{\partial a_k^{(l)}}{\partial \mathbf{\Gamma}}$ can be evaluated using the Eq. (7.15)
      ii. In the Eq. (7.33), the elements $\frac{\partial \Gamma_{lm}}{\partial \gamma_{ij}}$ of the matrix $\frac{\partial \mathbf{\Gamma}}{\partial \gamma_{ij}}$ can be evaluated using the Eq. (7.31).
      iii. In the Eq. (7.33), the quantities $\frac{\partial E^q}{\partial a_k^{(l)}}$, can be evaluated using the backpropagation algorithm (see the Eq. (4.28)).
   c) Use the gradients accumulated in the previous step to perform an update of the matrix $\boldsymbol{\gamma}$. Evaluate the stopping criterion. If the criterion is not met, go back to a).

7.4 Conclusion

In this chapter, a novel adaptation technique called MELT (Minimum Error Linear Transform) was introduced. The MELT adaptation technique was developed for an hybrid speech recognition system using the LTSP features during acoustic modeling.

The positive aspects of the MELT method are as follows

1. Variable number of free variables.
2. The method is computationally efficient during adaptation phase.
3. Ties together mel-filterbank interpolation normalization and LTSP normalization.
4. Flexible, it can be applied either on the mel-filterbank outputs or on the LTSP vector or on the weight matrix of the input layer.
5. Based on application, it is possible to achieve zero computational overhead during the recognition phase.
6. Under circumstances relatively common in the field of the hybrid speech recognition, it can be proven that the normalization parameters are estimated in a maximum-likelihood fashion.

8 Speech Corpora Used In This Work

8.1 The Czech SpeechDat(E)

The Czech SpeechDat(E) is a Czech portion of the SpeechDat(E) speech corpus. The SpeechDat(E) is a telephone speech corpus targeted primarily at a construction of telephone IVR systems. The design of the corpora is detailed in [18]. The content of the database consists of several types of utterances: isolated digits, number strings, natural numbers, money amounts, yes/no questions, dates, times, application keywords/command phrases and phonetically rich words and sentences.

The Czech portion contains recordings from 1052 speakers (526 males and 526 females). Due to the date of creation of the database, the recordings cover only the land-line telephone network.

For purposes of this work, only the set of 12 phonetically rich sentences was used for training, since the incorporation of other types of utterances could result in phonetically unbalanced (thus unrepresentative) training data. Using only the mentioned corpora portion, about 50 seconds of speech was available for each of the speakers.

For evaluation purposes, the sentences defined in testing portion of database[1] (200 speakers) were used. The training portion of database[2] (852 speakers) was split to the training (700 speakers) and the development set (152 speakers).

The audio is sampled at 8 kHz and stored as 8-bit A-law.

To speedup the training, instead of flat-alignment, a phone-level alignment was used that was obtained by means of force-aligning the reference transcript by an existing proprietary HMM/GMM speech recognition system. The SAMPA phonetic alphabet (44 phonemes) was reduced. Specifically, the long (/a:/, /e:/, /i:/, /o:/, /u:/) and the corresponding short vowels (/a/, /e/, /i/, /o/, /u/) were merged into one phoneme. The reasoning was as follows. Since the short and the corresponding long vowel variants differ only in duration, their spectral envelope stays the same. The duration should be modeled in the later stages of recognition process (i.e. in HMM) and not in the neural network.

The LTSP coefficients were obtained using these parameters: analysis window 25 ms, 10 ms shift, 15 log-mel coefficients, DCT-len 11. The resulting LTSP vector was 330 elements long.

[1] `A3TSTCS.SES`
[2] `A3TRNCS.SES`

8.1.1 Phoneme Level Language Model

For the phoneme language model, the training data of the SpeechDat(E) database were used. For the estimation of probabilities of unseen n-grams the *SRI Language Modeling Toolkit*(SRILM) implementation of Ristad's natural discounting law ([92]) was used. During the n-gram modeling, two additional n-grams were added to allow for modeling the start and the stop of an utterance. The n-gram coverage can be found in the Table 8.1.

n-gram order	1	2	3	4	5	6	7	8
counts	40	1315	18742	136084	459970	824924	1004876	1002513

Table 8.1: SpeechDat(E) corpus phoneme coverage

8.2 The DARPA TIMIT Acoustic-phonetic Continuous Speech Corpus

The TIMIT Corpus[34] is a corpus of English read speech designed to provide data for automatic speech recognition system. Because of its age, it is relatively small in size (compared to the speech corpora sizes nowadays).

The corpus contains recordings of 630 speakers, grouped into 8 groups according to the speaker's dialect. The speaker set was selected to provide a sufficient covering of 7 distinctive dialect regions. Additionally, an eighth group called "Army Brat" is introduced that covers the speakers who frequently moved during their childhood.

Every speaker recorded 10 sentences, thus the corpora contains 6300 sentences in total (5.4 hours). From the total of 10 sentences, 2 sentences are referred to as dialectal type sentences, 5 sentences as phonetically compact and 3 sentences as phonetically diverse. The dialectal sentences were designed with the intention of exposing the dialectal varieties and thus, all speakers uttered the same two text prompts. The phonetically compact sentences were designed to provide as diverse coverage of phone pairs as possible. Special care was paid to introduce the occurrences thought to be either difficult or of particular interest. The phonetically diverse sentences were selected from existing text corpora. The selection process was based on maximization of the variety of allophonic contexts found in the texts.

Sentence Type	#Sentences	#Speakers	Total	#Sentences/Speaker
Dialect (SA)	2	630	1260	2
Compact (SX)	450	7	3150	5
Diverse (SI)	1890	1	1890	3
Total	2342		6300	10

Table 8.2: TIMIT speech material, from [38]

The recordings are sampled at 16 kHz and stored as 16-bit linear PCM (compressed using the lossless *shorten* compression algorithm[95]). The TIMIT corpus includes

time-aligned orthographic, phonetic and word transcriptions. Moreover, the testing and training sets are defined thoroughly. The training set contains 4620 utterances, however usually the dialect sentences are omitted, which results in 3696 sentences ($\approx$ 3.5 hours of speech) and the test set (again, with dialect sentences omitted) consists of 1344 sentences ($\approx$ 1 hour of speech).

The fact that detailed orthographic and phonetic alignments are provided makes this corpus a suitable corpus for basic experiments. Because its size is generally considered large enough to provide an unbiased view on the methods' performance while small enough to provide a fast turnaround time for experiments, it is generally considered as a standard database for speech recognition[71]. Due to its limited size, the applicability of the obtained acoustic models in the real-world is somewhat limited.

The phonetic alphabet used in the corpora was inspired by ARPABET. For automatic speech recognition purposes, the original alphabet (61 phones) is usually compacted into a 39 phones alphabet[68].

The LTSP coefficients were obtained using these parameters: analysis window 25 ms, 10 ms shift, 23 log-mel coefficients, DCT-len 11. The resulting LTSP vector is 506 elements long.

8.2.1 Phoneme Level Language Model

For the phoneme language model, the training data of the TIMIT database were used. For the estimation of probabilities of unseen n-grams, the SRILM implementation of Ristad's natural discounting law ([92]) was used. During the n-gram modeling, two additional n-grams were added to allow for modeling the start and the stop of an utterance. The Table 8.3 shows the n-gram coverage.

n-gram order	1	2	3	4	5	6	7	8
counts	41	1294	9775	17086	15543	13438	121146	11127

Table 8.3: TIMIT corpus phoneme coverage

8.3 WSJCAM0 Cambridge Read News

The Cambridge Read News corpus is a corpus complementing the original DARPA Continuous Speech Recognition Corpus (CSR-I) WSJ0. However, the original CSR-I (or WSJ0) corpus contains recordings of American English speakers, the WSJCAM0 contains recordings of native British English speakers.

The corpus is comprised of recordings of 140 speakers. The training part of the corpus consists of 92 speakers (39 females and 53 males). The development test set consists of 20 speakers and each of the two evaluation test sets consists of 14 speakers. Approximately 90 sentences and an additional common set of 18 adaptation sentences has been recorded for each speaker. The adaptation sentences were selected from the set of 40 adaptation sentences in the WSJ0 corpus[37].

The sentence sets prepared for the speakers participating in the recording of the evaluation test set and the development test set were selected specifically in such a manner that the first 40 sentences contained only words from a fixed 5000 word vocabulary (denoted as 5k-closed) and the other 40 sentences contained words from a fixed 64000 word vocabulary (denoted as 20k-open).

Recordings were made from two microphones: a far-field desk microphone and a head-mounted close-talking microphone. For the experiments, only the far-field desk microphone recordings were used. The recordings are sampled at 16 kHz and stored as 16-bit linear PCM (compressed using the lossless *shorten* compression algorithm[95]). The total amount of audio data available for training is approximately 18 hours, accompanied by 2 hours of development audio and 3 hours of evaluation (test) audio.

The WSJCAM0 alphabet was modified in a way similar to [68]. Moreover, for historical reasons[3], the phonemes /aa/ and /ao/, /ah/ and /ax/, /sh/ and /zh/ were merged. The resulting alphabet consists of 44 phonemes. Besides the three new phonemes covering British English (/oh/ for the vowel in "pot", /ia/ in "peer", /ea/ in "pair" and /ua/ in "poor"), the alphabet is identical to the alphabet used during the TIMIT experiments.

The LTSP coefficients were obtained using these parameters: analysis window 25 ms, 10 ms shift, 23 log-mel coefficients, DCT-len 11. The resulting LTSP vector is 506 elements long.

8.3.1 Phoneme Level Language Model

For the phoneme language model, only the training data of the WSJCAM0 corpus was used. For the estimation of probabilities of unseen n-grams, the SRILM implementation of Ristad's natural discounting law ([92]) was used. During the n-gram modeling, two additional n-grams were added to allow for modeling of start and the stop of an utterance. The Table 8.4 shows the n-gram coverage.

n-gram order	1	2	3	4	5	6	7	8
counts	44	1391	14707	50449	83152	96420	98617	97351

Table 8.4: WSJCAM0 Phone-level order n-gram counts for LM-SMALL language model

8.3.2 Word Level Language Model

For experiments with a word-level LVCSR, three different language models were used. For simplicity, they will be referred to as LM-ZERO, LM-SMALL and LM-BIG.

- The language model LM-ZERO is a zerogram language model. There are 1576 words in the recognition network.
- The language model LM-SMALL is a bigram language model that was trained on the training and the development portions of the transcripts from the WSJCAM0

[3] to retain a compatibility with the American English alphabet used in our department

corpus. The modified Knesser-Ney smoothing for interpolation with no bigram cutoff was used. Owing to the fact that both the training and the development data were used, there are no OOV words during the training and the tuning phase. The Table 8.5 shows the n-gram coverage.

n-gram order	1	2
counts	12075	74388

Table 8.5: WSJCAM0 Word-level order n-gram counts for LM-SMALL language model

- The language model LM-BIG is a trigram language model that was trained from the English Gigaword Corpus (1200M of tokens) using the *English Gigaword language model training recipe*[4]. For the interpolation, the modified Knesser-Ney smoothing method was used (as implemented in the SRILM toolkit). For cutoffs, bigram cutoff 3 and trigram cutoff 5 were used. The dictionary size was limited to the most frequent 64000 words. Because other than WSJCAM0 corpus data were used for the language modeling, there are OOV words even during the training phase. The Table 8.6 shows the n-gram coverage.

n-gram order	1	2	3
counts	64000	6501697	12151781

Table 8.6: WSJCAM0 Word-level order n-gram counts for LM-BIG language model

8.4 Conclusion

In this chapter, the speech corpora used in this thesis were introduced and described. The speech corpora were chosen specifically to cover a wide range of possible variations. The size of speech corpora varies from small (TIMIT) to medium (SpeechDat(E), WSJCAM0). Moreover, the corpora cover two languages (or three, if the American English and the British English are seen as different languages) and different recording conditions – either telephone bandwidth or desc microphone bandwidth.

[4] `http://www.keithv.com/software/giga/`

9 Experiments and Results

9.1 Preliminary Experiments

The first battery of experiments was run on the two smaller databases – the TIMIT and the SpeechDat(E). Its purpose was, for one thing, to verify the implementation of neural network training software and, for another, to verify the functionality of the proposed adaptation method and the method's implementation and to evaluate the possible improvements of the recognition accuracy. Moreover, the experiments were used to formulate the adaptation strategy.

9.1.1 Objectives of the Preliminary Experiments

The first objective of the preliminary experiments was to validate the software developed in a partial fulfillment of the scope of this work. To accomplish this, the SpeechDat(E) and the TIMIT corpora were chosen. The reason, why the SpeechDat(E) was chosen is that the Czech speech recognition is of major interest in the author's supervisory department. The reason, why the TIMIT was chosen is that using this de-facto standard English speech corpus will enable evaluation of the performance of the training system and the resulting phone recognition system in the scope of world-wide research.

Different neural network topologies, mainly with the 1CTX (single context network) topology and the LCRC (left context, right context network) topology were experimented with. Moreover, the influence of a different number of neurons in the hidden layer on the recognition accuracy was evaluated.

The second objective was to evaluate a suitable adaptation strategy. Before thorough testing of any proposed adaptation method, it is beneficial to answer basic questions about adaptation procedures. Some of these elementary questions include whether to include the frames containing silence into the speaker-specific data and how much of the adaptation data is actually needed to see any recognition accuracy improvements, when using the speaker-adapted model. It is always beneficial to cross-check the findings on multiple speech corpora.

Several experiments were performed to evaluate the sensitivity of the recognition accuracy on choices of the adaptation matrix structure and/or different choices of phoneme classes that the adaptation should be performed on.

The shape of the matrix $\boldsymbol{\gamma}$ was either "`diag`" representing a diagonal matrix or "`band`" representing a tridiagonal matrix. The class of phonemes the adaptation was performed on is denoted by either "`all`", representing adaptation on all phoneme classes or "`nosil`" representing adaptation on all phonemes except silence and non-speech event classes and,

finally "`vowel`" representing adaptation only on vowel phonemes.

Regularization Influence

In some cases, when not enough data is available, regularization approach can be used as a remedy of this problem. Intuitively, the regularization increases "stiffness" of the adaptation task, i.e. prevents the values of $\boldsymbol{\gamma}$ from diverting too much from some other, a-priori chosen, valid solution.

To compute the regularization cost, the Frobenius norm in the form of

$$E_r = \|\boldsymbol{\gamma} - \boldsymbol{I}\|_{\mathrm{F}} \tag{9.1}$$

was used, where $\boldsymbol{I}$ denotes the unity matrix. In this case, the solutions that are too far (in the sense of the Frobenius norm) from the unity matrix, are penalized. Since the optimal ratio between the training error and the regularization error is unknown, the Eq. (7.19) is modified to include the term E_r in the following way

$$E(\Psi|\boldsymbol{\gamma}) = \sum_q E^q + \kappa E_r(\boldsymbol{\gamma}), \tag{9.2}$$

where the constant κ is called the *regularization coefficient* and is usually determined by choosing its value from a predefined set of weights. The error criterion Eq. (9.2) is then optimized instead of the original criterion given by the Eq. (7.19). The gradient of the modified (regularized) error function is then

$$\frac{\partial E(\Psi|\boldsymbol{\gamma}')}{\partial \gamma_i} = \sum_q \frac{\partial E^q}{\partial \gamma_i'} + 2\kappa\gamma_i' \qquad \text{for } 1 \leq i \leq G, \tag{9.3}$$

where T is the number of adaptation samples. It is worth noticing that if $\kappa = 0$, the criterion Eq. (9.2) turns back into the original criterion Eq. (7.19).

9.1.2 Experiments Performed on the SpeechDat(E) Corpus

Baseline System

The baseline SI network was trained on the complete training part of the corpus. Two test sets were defined. The first test set was identical to the official SpeechDat(E) test set. It was used to compare the performance of the SI system to other research labs' results. Unfortunately, the SpeechDat(E) does not specify an adaptation test set. Therefore, the other test set, intended for the adaptation evaluation, had to be prepared using the procedure described below.

From the official testing set, the 97 speakers that were represented by all 12 sentences were selected. For each speaker, the appropriate set of 12 utterances was split randomly into a set of 10 utterances and a set of the 2 remaining utterances. The sets of 10 sentences were used as the adaptation data and the sets of 2 sentences as the test data.

The audio recordings are about 6–8 seconds long; however the utterances themselves are just about 2–5 seconds long. Taking only the actual speech into account, there was approximately 40 seconds of speech data available to adapt the neural network on.

The baseline scores are as follows.

- Baseline: UWB, Acc = 77.35 %
  - UWB phon. alphabet
  - 1CTX-1500, ≈ 660k parms
- Comparison: BUT[1], Acc = 75.76 %
  - Sampa phon. alphabet
  - LCRC-1500, ≈ 1.6M parameters
- Adaptation test set, before adaptation Acc = 75.63 %
  - Adaptation on approx. 40 seconds of speech

Please note that the phonetic alphabets used in the UWB recognizer and in the BUT recognizer differ from each other. To allow a fair comparison, a third alphabet, called "merged" alphabet, was designed and the texts subjected to comparison were translated into it. The alphabet was created by means of merging the ambiguous phones into a single phonetic table entry.

The proposed merging approach dealt primarily with ambiguities originating mainly from voiced and devoiced variations of the same phoneme and from different pronunciation variants. When both the recognized phonemic sequences were translated into the merged alphabet, accuracy of them both was evaluated individually. According to the scores, no system performed statistically significantly better than the other.

Comparison Against the VTLN

First, preliminary experiments to verify the performance gain (or loss) against the bilinear VTLN were performed. Using the SI ANN, the testing data set was recognized. The recognized output was then used as a reference phone alignment and the computation of normalization factors was performed on the recognition result instead of on the reference phonetic transcript. Please note that these results originate from a different development stage of the MELT system, thus even the baseline results differ. The important message here is, however, that MELT performs better than VTLN. The overall results can be found in the Table 9.1. The symbols ANN-1500 and ANN-2500 denote that neural networks with 1500 and 2500 neurons in the hidden layer were used.

[1] Brno University of Technology: Phoneme Recognizer based on Long Temporal Context, `http://speech.fit.vutbr.cz/en/software/phoneme-recognizer-based-long-temporal-context`

	base	VTLN	MELT=2
ANN-1500	76.64	80.16	80.20
ANN-2500	79.25	81.20	81.51

Table 9.1: Comparison of the MELT and the VTLN performance (Acc) when the reference transcript is unknown

Adaptation Performance

The experimental results are shown in the Table 9.2. Using the MELT method with a tri-diagonal matrix (MELT = 2), in the table denoted as `band_nosil`, the adaptation process improved the recognition score of about 1 %.

	$\kappa = 0$	$\kappa = 10$	$\kappa = 100$	$\kappa = 500$	$\kappa = 1000$
`band_all`	75.30	75.30	75.54	75.92	75.87
`band_nosil`	$\mathbf{76.54}_{p=0.003}$	76.45	76.30	76.11	75.63
`diag_all`	74.92	75.06	74.92	76.02	75.73
`diag_nosil`	75.87	76.07	75.92	75.49	75.49
`diag_vocal`	75.87	75.87	75.83	76.26	76.07

Table 9.2: SpeechDat(E): Recognizer accuracy (Acc) given the adaptation scheme and the regularization coefficient κ

For the best combination of the regularization constant and the adaptation matrix $\boldsymbol{\gamma}$ shape, the Wilcoxon signed rank test under the null hypothesis that median of the difference between the unadapted and adapted networks is zero was performed. The p-value was $p = 0.003$, which is sufficient to reject the null hypothesis at the level $\alpha = 0.003$.

As already has been said, the experiments on the SpeechDat(E) were supposed to provide an insight into the correctness of the neural network training software. As can be seen from the comparison of the baseline to the BUT system, the baseline performs on par with the BUT system, yet with only 40 % of parameters (660k vs. 1.6M).

9.1.3 Experiments Performed on the TIMIT Corpus

Baseline System

The basically same set of experiments was performed on theTIMIT corpus as well. The official TIMIT test set was used for the evaluation of the SI system performance. For the evaluation of the adaptation performance, a modified test set was created as follows. The speaker dialect region sentences (2 sentences per speaker) were used as the adaptation data and the rest of sentences (8 sentences per speaker) were used for the evaluation of the adaptation performance.

- Baseline : UWB, Acc = 76.65 %

  - CMU phon. alphabet
  - LCRC-300, ≈ 330k parameters
- Comparison: BUT[2], Acc = 75.76 %
  - CMU phon. alphabet
  - LCRC-500, ≈ 550k parameters
- Unadapted subset Acc = 78.04 %
  - Adaptation on approx. 12 seconds of speech

Both the baseline and the reference system (BUT[100]) were using the same phonetic alphabet, so a direct comparison was possible. From the comparison it is clear that the system trained using the software developed in partial fulfillment of this thesis goals performs better, while it uses only 60 % of the BUT system parameters.

Adaptation Performance

	$\kappa = 0$	$\kappa = 10$	$\kappa = 100$	$\kappa = 500$	$\kappa = 1000$
band_all	75.30	75.30	75.54	75.92	75.87
band_nosil	78.12	78.30	78.34	78.58	$\mathbf{78.75}_{p=0.017}$
diag_all	74.92	75.06	74.92	76.02	75.73
diag_nosil	78.38	78.34	78.36	78.60	78.72
diag_vocal	75.87	75.87	75.83	76.26	76.07

Table 9.3: TIMIT: Recognizer accuracy (Acc) given the adaptation scheme and the regularization coefficient κ

Because the number of adaptation data was quite small (12 seconds per speaker), the regularization turned up as beneficial. The improvement of about 0.7 % might seem small, but it is statistically significant nonetheless. The statistical significance test was performed using the Wilcoxon signed rank test.

9.1.4 Conclusion and Findings of the Preliminary Experiments

The preliminary experiments suggest that the software and the procedure used for SI neural network training is fully functional. Moreover, the resulting phoneme recognition system performs at least the same as the reference system, while keeping the number of parameters significantly lower (40 % – 60 % of the number of parameters of the reference system). The most probable cause is the training algorithm. Instead of a gradient descent algorithm, the L-BFGS and the iRPROP (for reference, please refer to the subsection 4.6.3

[2]Brno University of Technology: Phoneme Recognizer based on Long Temporal Context, `http://speech.fit.vutbr.cz/en/software/phoneme-recognizer-based-long-temporal-context`

of this work) algorithms were implemented and used for the neural network training. When compared to a steepest gradient descent optimization, these two algorithms provide faster convergence and are possibly less prone to stopping in local minima.

As for the adaptation algorithm, the preliminary experiments suggest that the algorithm as well as the implementation are functional. Moreover, the experiments suggest that possibly more than 20–30 seconds of speech is necessary. The adaptation should be performed only on clean speech, the non-speech events and silent parts of adaptation utterances shall be removed.

Finally, it can be concluded that the method is sensitive to the number of free variables (cf. comparison between `diag` and `band`) and a more thorough evaluation of this sensitivity should be provided.

9.2 WSJCAM0 – The Main Experimental Corpus

9.2.1 Experiment Flowchart

Three sets of experiments were designed. The first two sets are based on the recognition of phonemes and words using the posterior probabilities estimates. The posterior probabilities estimates are computed using a fully trained neural network, converted to likelihoods (using the a-priori probabilities of individual states) and recognized using an HMM decoder.

The third set of experiments is somewhat different. Instead of using the outputs of the neural network, the bottleneck features can be extracted (see the subsection 7.1.2 for reference). These features are then used instead of the commonly used MFCC features to train a GMM/HMM system.

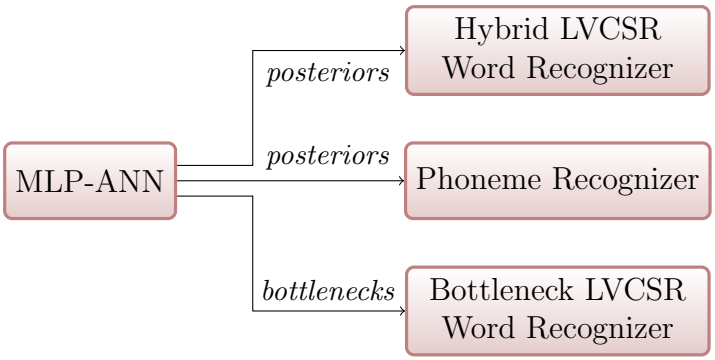


Figure 9.1: WSJCAM0 Experiments flowchart

For all these experiments, the same neural network was used. First, a SI network suitable both for bottleneck features production and posterior probabilities estimates was trained. Using this network, the bottleneck features on the training and the development sets were extracted and the Bottleneck LVCSR system was trained.

During the SAT, the reference phonetic transcript was used and the teacher information vectors were generated by means of force aligning the phonetic transcript using the likelihoods obtained using the posterior probabilities estimates. After the SAT was finished, the adapted bottleneck features were extracted and the SAT Bottleneck LVCSR system was trained.

9.3 WSJCAM0 – The Reference System

The reference recognition system is a common MFCC based GMM/HMM system. The parametrization process was performed as follows. The mel-filter bank coefficients were extracted from signal window 25 ms long at the frame rate of 10 ms. The coefficient were transformed to 13 cepstral coefficients using the DCT and the mean normalization was performed. The Δ and $\Delta\Delta$ (or acceleration) coefficients were computed from the raw MFCC coefficients and the feature vector was augmented using these. Therefore, the total number of features is 39.

The Table 9.4 contains performance metrics of the reference system. The notation is as follows. The overall recognizer accuracy per sentence is denoted as Acc, the 2.5% ($CI_{Acc-}(\%)$) and the 97.5% ($CI_{Acc+}(\%)$) confidence percentiles (representing the 95% confidence interval) of the accuracy per speaker were determined using the percentile corrected bootstrap method and the value denoted as σ is the standard deviation of the per-speaker recognition accuracy.

	Acc(%)	$CI_{Acc-}(\%)$	$CI_{Acc+}(\%)$	σ
`si_tr`	92.28	91.84	92.64	1.9
`si_dt`	85.26	82.34	87.17	5.5
`si_et`	83.64	81.13	85.41	5.8

Table 9.4: WSJCAM0: The recognition score of the reference (a GMM/HMM) system

9.3.1 Comparison of the Reference System

To verify the performance of the reference system and, consequently, to establish the credibility of the baseline system performance, the reference system was compared against other systems that use the WSJCAM0 test corpus. Because the papers usually report scores on different subsets, the previously mentioned scores are not directly usable. Therefore, the scores were computed using the procedures and subsets mentioned in the compared papers. Moreover, if the WER measure was used instead of the Acc measure, it was transformed to the Acc measure.

In the very new paper [120], the used GMM/HMM system is reported to achieve 88.12 % and 86.93 % on the subsets `dt5a` and `dt5b` respectively. This compares favorably to our numbers on the same datasets: 88.62 % and 88.08 %.

The authors of the corpus published a technical report [96] comparing their recognition accuracy (88.6 % on `si_dt5` and 82.1 % on `si_dt20`) with the previously published

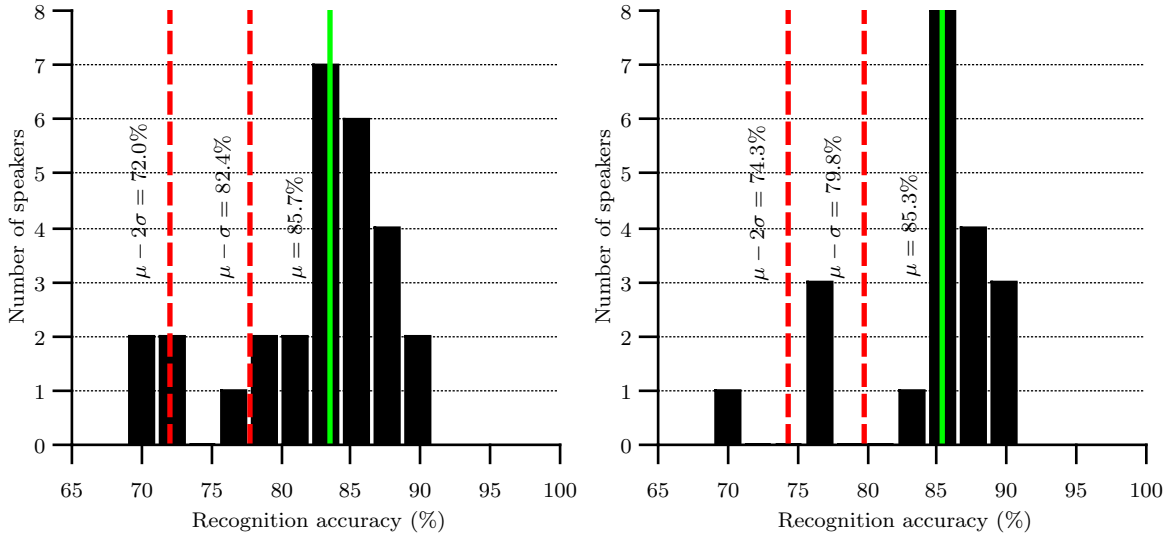


Figure 9.2: Reference system – Recognition accuracy histogram on `si_dt` (left) and `si_et` (right) sets

WSJ0 corpus to prove the correctness and consistency of the WSJCAM0 corpus. The ABBOT recognition engine was used to obtain these numbers. At the time of writing the paper, the ABBOT was regarded as a state-of-the-art recognition system and used up-to-date techniques including the ANNs and a fusion of four frontends. The standard November 1993 WSJ0 language model[37] was used. Performance numbers on these subsets of the reference system are: 88.3 % and 82.4 %. The authors of paper [123] report 87.2 % Acc on `si_dt5b` (the reference system has 88.3%).

Two other authors used the HTK and report very similar scores. In paper [5], the bigram language model from the original WSJ0 corpus[37] was used. Using this setup, the authors report 68.6 % Acc. Likewise the authors [57] achieved a similar Acc of 69.1 %.

Moreover, a Ph.D. thesis [22] discussing the possibility of inclusion of the eye movement modality into a LVCSR system reports performance on the WSJCAM0 corpus. Specifically, the work publishes performance scores on three different sets: 21.0% on the `si_dta` set and 20.81% on the `si_dtb` set and 20.91% on the combined `si_dta`+`si_dtb` set. These scores are obtained using a bigram backoff language model trained on the WSJCAM0 training and evaluation data.

9.4 WSJCAM0 – Results for the Baseline (Unadapted) System

9.4.1 Phoneme Recognition

The feature vectors have a quite long temporal span. It can be argued that even though the neural network models use only monophones (or more precisely, it uses three-state

monophone units), the long temporal windows enable the neural network to "see" the last one or more states of the previous and the first one or more states of the following monophone unit and to take advantage of it. In other words this means that the neural network itself would construct a language model. Assuming this hypothesis to be valid, it would not be possible to achieve a significant performance gain employing language models of lower orders during recognition. What means "lower order"? Because the architecture, that was employed, has no internal state nor memory and the input feature vectors span approximately three consecutive phonemes[3], the highest order language model approximated internally would probably be close to 3-gram LM.

LM order	1	2	3	4	5	6	7	8
`si_tr` Acc(%)	85.07	85.15	87.14	90.34	91.89	92.99	93.01	93.43
`si_dt` Acc(%)	75.26	76.28	79.14	80.98	81.93	82.19	82.27	82.16
`si_et` Acc(%)	77.09	77.09	80.04	81.82	82.66	83.00	82.85	82.92

Table 9.5: WSJCAM0:Phoneme Recognition Acc of the baseline (unadapted) system

As can be seen from the table, there is basically no difference in the recognition accuracy between unigram and bigram language models, there is however a significant increase of accuracy when language models of higher orders are employed. This leads us to conclusion that the language model constituted by the neural network is an approximation of a bigram LM.

Experiments with multi-pass phoneme recognition procedures were performed as well. In the first pass, the phoneme lattice was generated using a pseudo n_1-gram language model and pruned to contain only the Q best hypotheses in each node. This language model will be referred to as a pseudo n_1-gram, because the n_1-gram language probabilities were used, however the recognition lattice was only a bigram lattice. In the second pass, the pruned lattice was re-scored using a full n_2-gram language model. In the Table 9.6, the influence of n_1 and n_2 constants on recognition accuracy is presented.

n_2 \ n_1	4	5	6	7	8
0	81.82	82.66	83.00	82.85	82.92
5	83.89	84.26	84.40	84.51	84.48
6	84.11	84.47	84.63	84.63	84.59
7	83.86	84.51	84.51	84.64	84.59
8	83.93	84.45	84.64	84.62	84.58

Table 9.6: WSJCAM0: Influence of the rescoring setting on the phoneme recognition accuracy

As can be seen, the re-scoring is a beneficial approach. The pruning constant Q has an influence on the recognition accuracy as well, however this is quite a marginal effect.

[3]The rate of speech in English is usually reported to vary between 12-16 phonemes per second[116, 76].

The limiting factor is most probably the quality of the phoneme language model. It has a significant influence on the recognition speed, though. In the Table 9.7, the influence of the pruning coefficient Q on the recognition accuracy is presented.

Q \ n_1/n_2	5/5	5/6	5/7	5/8
2	84.26	84.47	84.51	84.45
3	84.80	85.10	85.12	85.04
4	85.01	85.33	85.39	85.32

Table 9.7: WSJCAM0: Influence of pruning setting on phoneme recognition accuracy

For the all next experiments with phoneme recognition system, the values $n_1 = 5$, $n_2 = 6$ and $Q = 2$ were chosen. The performance difference between this setup and the best performing setup is about 1% absolutely, however this is on the account of significantly increased recognition times (the RT factor for the chosen configuration is 1/467, the RT factor for the best config is 2).

9.4.2 Hybrid LVCSR Word Recognition

The estimates of the posterior probabilities of individual states were converted to likelihoods using the a-priori probabilities obtained during the training phase. These likelihoods were then used in the LVCSR system in the same way as the usual likelihoods obtained from the gaussian mixture models. The Table 9.8 shows the performance of the system for all the three data sets.

	Acc(%)	CI_{Acc-}(%)	CI_{Acc+}(%)	σ
`si_tr`	89.85	89.24	90.38	2.8
`si_dt`	84.35	80.67	86.43	6.5
`si_et`	85.06	82.59	86.83	5.8

Table 9.8: WSJCAM0: Word recognition Acc of a hybrid, posteriori estimates based speech recognition system

For experiments, the LM-SMALL was used to determine the optimal values of the language model weight (LWM) and word insertion penalty (WIP). The optimal values were determined experimentally, using a gradient descent algorithm with gradient approximation. The recognition itself was run using these constants and the LM-BIG language model. This allowed us to perform the tuning phase much faster, after verification that the values determined using the LM-SMALL are (almost) optimal for LM-BIG as well.

From the Fig. 9.3, it can be seen that the average recognition score for the `si_dt` dataset is highly biased because of one speaker performing significantly worse than others. The recognition accuracy is even lower than $\mu - 3 \times \sigma$, the rule-of-thumb for detection of outliers. The ID of the speaker is `c42`. According to the information from the speech

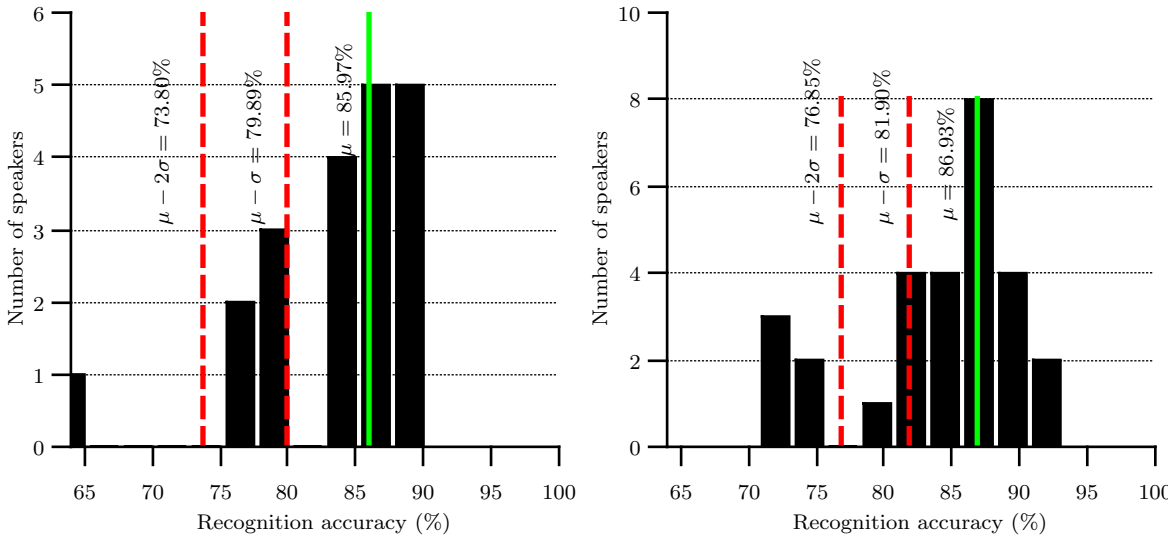


Figure 9.3: Monopohone posteriors – Recognition accuracy histogram on `si_dt` (left) and `si_et` (right) sets

corpus, the speaker is female, 18 years old. From the subjective point of view, the speaker does not feature any particular speech impediment or dialect, only the speech tempo is somewhat increased.

To compare this recognizer to the reference recognizer, the p-value of the Wilcoxon signed rank test was computed. For the `si_dt` set $p = 0.08$ and for the `si_et` set, the p-value is $p = 0.01$. This indicates that the recognizer performs better than the reference one.

9.4.3 Bottleneck LVCSR Word Recognition

In these experiments, the bottleneck features were extracted instead of the posterior estimates. The bottleneck features were then used to train a common GMM/HMM recognition system. The Table 9.9 shows the performance for all three datasets. The confidence intervals were determined using the bias corrected and percentile accelerated bootstrap method.

	Acc(%)	CI_{Acc-}(%)	CI_{Acc+}(%)	σ
`si_tr`	90.77	90.17	91.22	2.5
`si_dt`	85.42	82.14	87.42	6.0
`si_et`	84.85	82.53	86.49	5.4

Table 9.9: WSJCAM0: Performance of a hybrid, bottleneck features based speech recognition system

As can be seen from the results, the performance is on the par with the results obtained using the monophone posteriors method. The p-values of the Wilcoxon rank sum test are

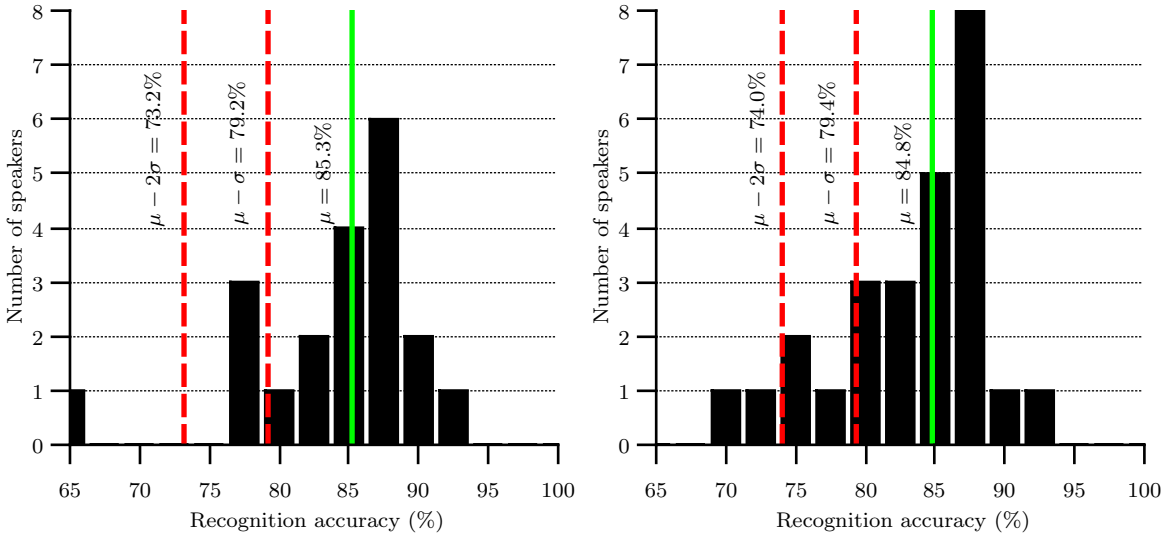


Figure 9.4: Hybrid LVCSR – recognition accuracy histogram on `si_dt` (left) and `si_et` (right) sets

$p = 0.64$ and $p = 0.99$ for the `si_dt` set and the `si_et` set, respectively. Possibly more interesting comparison is with the reference MFCC-based GMM/HMM recognizer, the p-values of the Wilcoxon rank sum test are $p = 0.77$ and $p = 0.01$ for the `si_dt` set and the `si_et` set, respectively.

9.5 WSJCAM0 – Speaker Adaptive Training

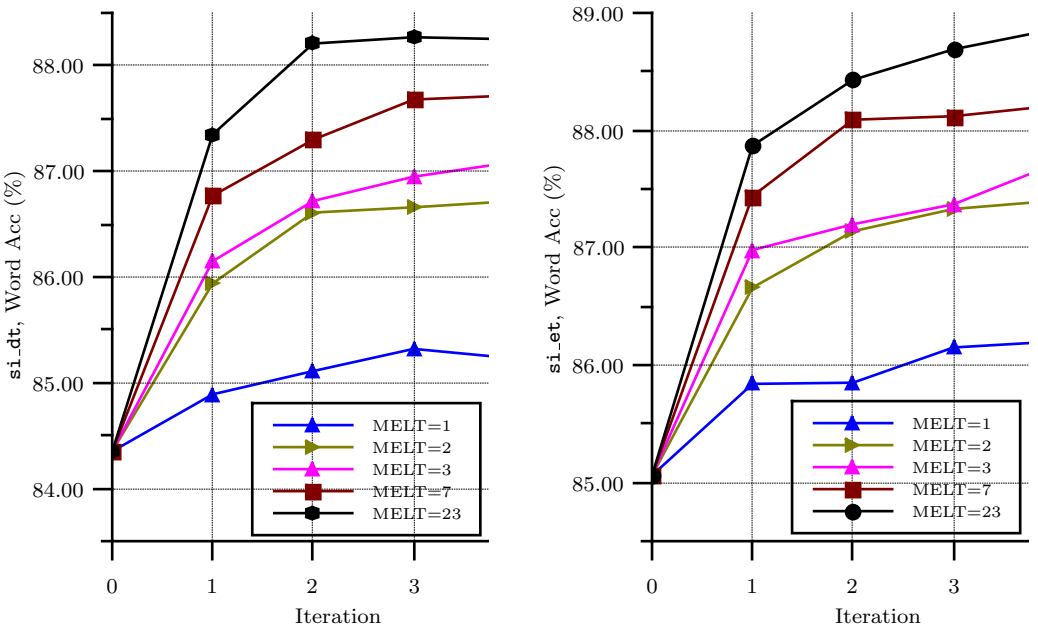
9.5.1 Training Process Description

The SAT training process was initialized using the data from the SI network training process. As the $cANN(0)$, the final SI neural network was used and the MELT matrices ($SPK(0)$) were initialized to the unity matrix. The target vectors (representing the teacher information) in the k-th step were obtained by means of force aligning the reference phonetic transcript using the likelihoods produced using the $cANN(k)$.

The number of SAT cycles was limited to 4. As can be seen from the Fig. 9.5, the scores did not increase significantly after the third SAT cycle anyway. Please note that the correct transcript was used even for the development and evaluation sets. The evaluation process was the same as for the SI network. The WIP and the LMW constants were determined using the `si_dt` set and then used during the recognition of the `si_et` set.

While this setup can be seen unpractical or even misleading, it allows monitoring the influence of the MELT level on the capacity or possibility of improvement with other factors omitted. Later, when discussing the unsupervised SAT, the values obtained here will be used as "oracle" values, i.e. the best values possible to obtain.

The following experiments were performed.



(a) Performance on the `si_dt` set

(b) Performance on the `si_et` set

Figure 9.5: The word recognition accuracy as a function of number of the SAT cycles.

1. The phoneme recognition, the 5/6 language model was used

2. The LVCSR using posteriors estimates. The LM-BIG language model was used.

9.5.2 Results

As can be seen on the Fig. 9.6, increasing the number of free variables improves the overall performance of the classifier. Moreover, it suggests that the performance is largely a linear function of the percentage of the total number of adapted coefficients. To confirm the assumption of linear dependence of score on the total number of free variables, linear fitting experiments were performed and the goodness of fit was evaluated. The first result (obtained for the strictly diagonal matrix) and the last result (obtained for full matrix adaptation) were not used for fitting.

On the Fig. 9.7(a), the recognition score relatively to the percentage of all adaptation matrix coefficients is depicted. The coefficient of determination R^2[110], $R^2 = 0.95$ suggests that the linear approximation fits the data well. On the Fig. 9.7(b), the recognition score vs. the absolute number of mel-filterbank outputs used for interpolation is depicted. The linear fit is, however, worse (the coefficient of determination is $R^2 = 0.89$) than in the previous case.

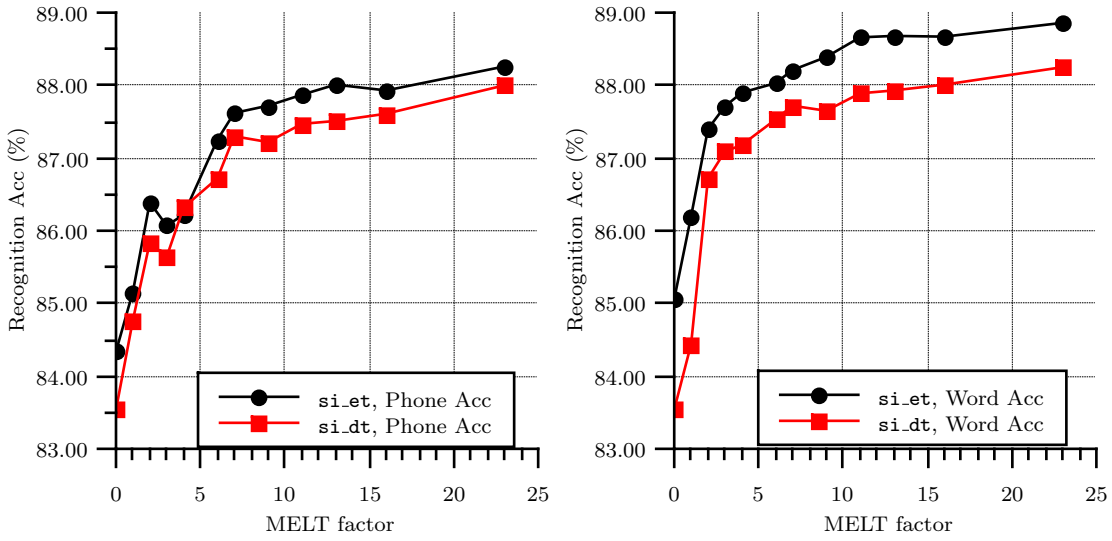


Figure 9.6: The phoneme and the word recognition score as a functions of the MELT factor

9.5.3 Conclusion

The performed experiments suggest that the MELT approach is beneficial in the speaker adaptive training. The same transformation/normalization matrix can be used for the phoneme recognizer as well as for the word recognizer adaptation.

	Acc(%)	$CI_{Acc-}(\%)$	$CI_{Acc+}(\%)$	σ
si_dt	87.96	84.57	89.93	5.8
si_et	88.63	86.99	89.81	4.0

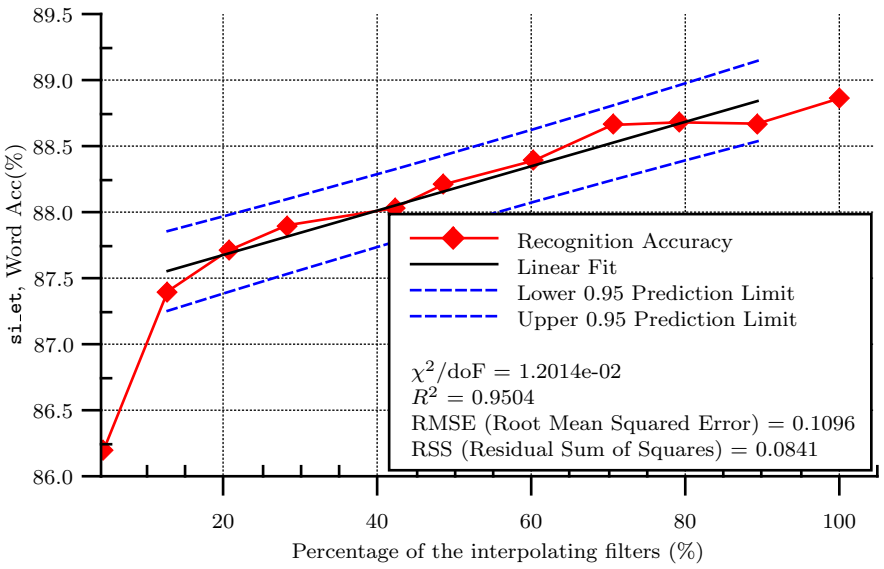
Table 9.10: WSJCAM0: Performance of a hybrid, bottleneck features based speech recognition system, after SAT with MELT=23

By the comparison between the Table 9.10 and the Table 9.9, it can be concluded that the application of the SAT paradigm leads to an improvement of about 3.5 % in absolute numbers, which can be interpreted as approximately $\Lambda_{WER} = 23\,\%$ reduction of WER.

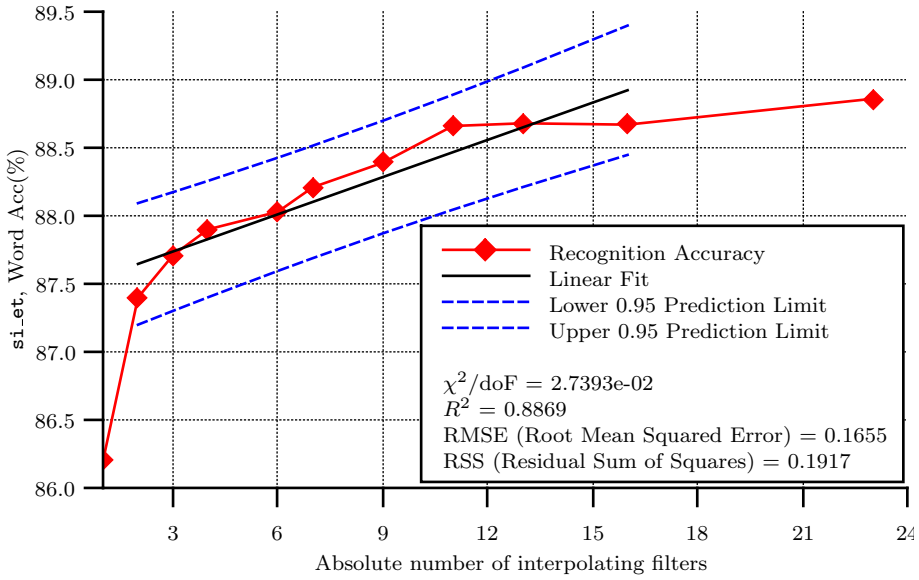
9.6 WSJCAM0 – Semi-supported Speaker Adaptive Training for the Hybrid Paradigm

9.6.1 Description of the Semi-Supported SAT

As has been already mentioned, the experiments performed in the previous section are useful from a theoretical point of view. The applicability of the method is, however,



(a) Recognition accuracy vs. percentage of interpolating filters



(b) Recognition accuracy vs. abs. number of interpolating filters

Figure 9.7: The word recognition accuracy on the evaluation set (`si_et`) as a function of the percentual number and the absolute number of filters used for interpolation, respectively.

severely limited in real life applications. The semi-supported SAT is a viable option in many real-life tasks.

The semi-supported SAT assumes that there exist a (quite often a relatively small) set of complete adaptation data, i.e. utterances together with the associated correct transcripts. This assumption can be fulfilled relatively easily. Usually, during the adaptation the user is presented with a known text (either specifically prepared for the adaptation or the speaker is given the choice to supply the text on his/her own) and the user reads the presented text. Then, the forced alignment procedure is executed to determine the correct pronunciation variants and word boundaries. Optionally, the likelihood of the forced-aligned utterance can be used to determine the sentences unsuitable for the adaptation and the user is asked to re-read them.

In the context of the work on the WSJCAM0 corpus, the 17 adaptation sentences supplied within the WSJCAM0 speech corpus were used. These 17 sentences stand for approximately 40 seconds of adaptation data.

The SAT process was performed as follows. First, the SI network was used in the combination with the adaptation data to determine the speaker normalization matrices. Using the speaker normalization matrices, the WIP and LMW constant for the speech recognition task and the phone insertion penalty and the weight of the phoneme language model were determined on the `si_dt` set. Word and phoneme recognition task were then performed on the `si_et` set using the aforementioned matrices and the experimentally determined penalties and constants (i.e. the WIP and the LMW for the word recognition task and the phone insertion penalty and the phoneme language model weight for the phoneme recognition task).

The following experiments were performed.

1. The phoneme recognition using the posterior estimates converted to likelihoods, the 5/6 language model was used

2. The hybrid LVCSR using posteriors estimates converted to likelihoods. The LM-BIG language model was used.

9.6.2 Results

The experiments were performed with a wide variety of MELT factors. The factors were chosen specifically so that each consecutive MELT factor represents approximately a 10 % increase of non-zero elements of the normalization matrix.

From the Fig. 9.8 it can be concluded that the aforementioned 40 seconds of acoustic data is sufficient for the MELT normalization of degree of about 7. In the Table 9.11, the word recognition Acc performance of the system normalized using the MELT factor 7 is shown.

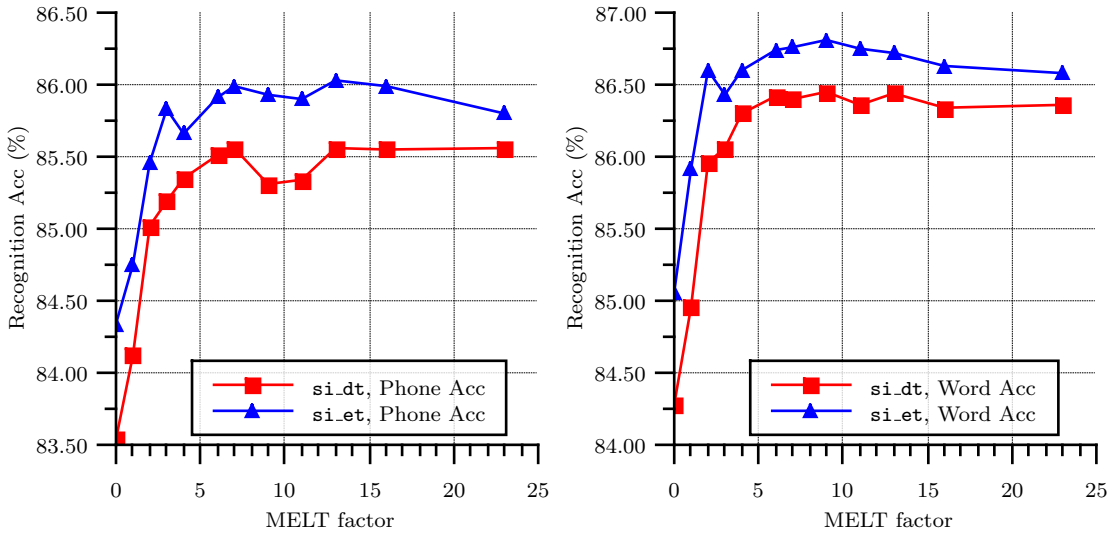


Figure 9.8: The phone recognition Acc (left) and the word recognition Acc (right) as a function of the MELT factor

	Acc(%)	$CI_{Acc-}(\%)$	$CI_{Acc+}(\%)$	σ
si_dt	86.40	82.78	88.45	6.1
si_et	86.76	84.99	88.23	4.5

Table 9.11: The word recognition accuracy on the semi-supported SAT task

9.6.3 Conclusion

As has been demonstrated, the MELT SAT can be used in a semi-supported fashion. For this application, the typical amount of data is quite small. The experiments suggest that given the amount of adaptation data, the MELT factor and the recognition constants (WIP, LMW) can be determined beforehand, during the development of the speech recognition system.

For the WSJCAM0 corpora setup, the optimal MELT factor was experimentally determined to be approximately 7. Using this factor, the overall improvement over the baseline (Table 9.8) was approx. 1.8 % in absolute numbers, which gives approximately 12 % reduction of the WER.

Compared to the oracle values (previous chapter), the semi-supported SAT on the 40 seconds of speech reached approx. 57 % of the possible improvement (oracle value for MELT=7 is 88.21 %).

9.7 WSJCAM0 – Unsupervised Speaker Adaptive Training for the Hybrid Paradigm

9.7.1 Description of the Two-pass Unsupervised SAT

In many practical situations, even the semi-supported SAT approach is not feasible or practical. In these situations, the completely unsupervised approach is usually used.

In this approach, the speaker utterances are recognized twice. In the first run (or first pass), the utterances are recognized using the SI recognizer. The recognized text is then taken as the referential transcript and is used to determine the speaker normalization matrices. Using the speaker normalization matrices, the utterances are recognized again (second pass).

Therefore, this approach relaxes the assumption of the availability of the reference transcript – the referential transcript is generated automatically. However, this method has some potential drawbacks. First, because of its two-pass nature, it is significantly more computationally demanding. Second, since the SI recognizer has a non-zero error rate (which is quite obviously the reason why an adaptation is performed), the adaptation is performed on a partially incorrect transcript.

When using a system providing the confidence factors for each of the recognized words, an additional threshold CF must be determined. The threshold CF specifies the minimal confidence the word has to have to be used for the adaptation. All the recognized words having the associated confidence lower than the predefined threshold CF are omitted during the adaptation.

The determination of the optimal factor CF is not always a straightforward task, because with an increasing value of CF, the amount of available adaptation data decreases. Therefore, a trade-off between the quality of the adaptation data and its amount must be determined. This is usually done experimentally, by virtue of testing the performance for several possible values of the CF threshold and choosing the one that performs best.

The SAT process was as follows. First, the SI network was used to recognize the `si_dt` set. The recognized text (and the associated confidence factor values) was used to determine the speaker normalization matrices. Using the speaker normalization matrices, the WIP and the LMW constants for the speech recognition task were determined on the `si_dt` set. The word recognition task was then executed on the `si_et` set using the aforementioned matrices and the experimentally determined penalties and weights. The following experiments were performed.

1. The hybrid LVCSR using posteriors estimates converted to likelihoods. The LM-BIG language model was used.

9.7.2 Results

As has been said, the solution of the problem of choosing the right threshold CF is not always the straightforward one. This is due to the tight coupling between the amount of data and the credibility of the data. The experiments were targeted at the exploration

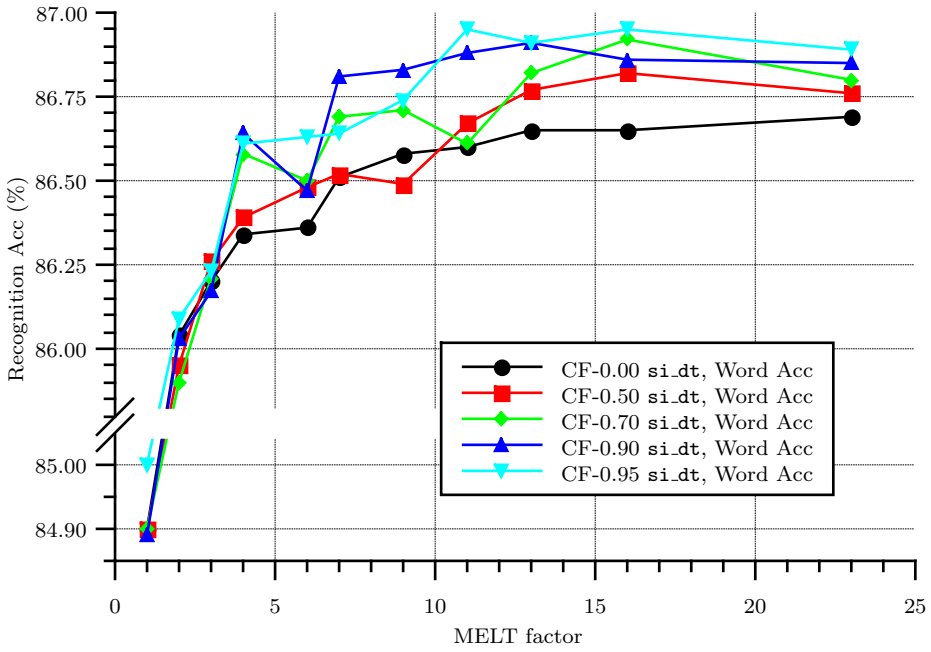


Figure 9.9: The word recognition accuracy on `si_dt` after unsupervised adaptation

of the influence of the choice of the threshold CF on the performance of the adapted system.

The experiments suggest (see the Fig. 9.9 for reference) that the confidence factor pruning is a beneficial technique in most cases. The system adapted using all the data (CF = 0.0) performs worst in most cases (all cases except one).

The optimal combination of MELT factor and the CF threshold is, for the WSJCAM0 setup, MELT = 11 and CF = 0.95. This combination yields the system performing best on the development data. Usage of the combination of these values during the adaptation leads to a speech recognition system, whose performance is depicted in more detail in the Table 9.12.

	Acc(%)	$CI_{Acc-}(\%)$	$CI_{Acc+}(\%)$	σ
`si_dt`	86.95	83.25	88.86	6.0
`si_et`	87.38	85.40	88.81	4.6

Table 9.12: The word recognition Acc on the unsupervised SAT task

The values for CF = 0.0 can be interpreted as the case where no system providing the confidence factors is available. In this case the optimal choice of the MELT factor is MELT = 23. The performance of the system constructed using MELT = 23 and CF = 0.00 is depicted in the Table 9.13.

	Acc(%)	$CI_{Acc-}(\%)$	$CI_{Acc+}(\%)$	σ
`si_dt`	86.69	82.75	88.60	6.3
`si_et`	87.23	85.19	88.73	4.8

Table 9.13: Word recognition Acc on the unsupervised SAT task, when no confidence factors are considered

9.7.3 Conclusion

As has been experimentally demonstrated, the MELT approach can be used relatively easily in the combination with the two-pass unsupervised speaker normalization scheme.

The drawbacks of this approach include an increased computational complexity during the development phase (as an additional threshold must be determined experimentally) and an increased computational complexity during the speaker adaptation, as the speaker utterances must be recognized twice.

A speech recognition engine providing the confidence factors is not a must. However, when available, it is beneficial to use it. The system obtained using the confidence factors uses a lower MELT factor (therefore it is supposed to be more robust) and performs better.

The overall improvement over the baseline (see the Table 9.8) is approx. 2.3 % in absolute numbers, which gives approx. 15.4 % reduction of WER. Compared to the oracle values, the unsupported SAT reached approx. 70 % of the possible improvement (oracle value for MELT = 11 is 88.39 %).

It should be noted that the optimal values determined experimentally are optimal only when the real-world conditions are sufficiently close to the conditions that were active during the determination of these values. It is especially true for the amount of adaptation data (approx. 3 – 4 minutes of speech) and the unadapted system performance. Should one or both these conditions change, there might exist a setup yielding a better performing system.

9.8 WSJCAM0 – The Unsupervised Speaker Adaptive Training for the Bottleneck Paradigm

9.8.1 Description of the Task

In the field of Bottleneck LVCSR systems, a bottleneck neural network is used as a feature extractor. In these systems, the ML or MAP adaptation of the GMM mixtures coefficients is usually used.

In the scope of this thesis, a different approach was explored. First, the SAT neural network was trained as described in the subsection 9.5.1. The trained SAT neural network was used to produce adapted bottleneck features on the `si_tr` and the `si_dt` datasets. The bottleneck system was trained using these adapted bottleneck features.

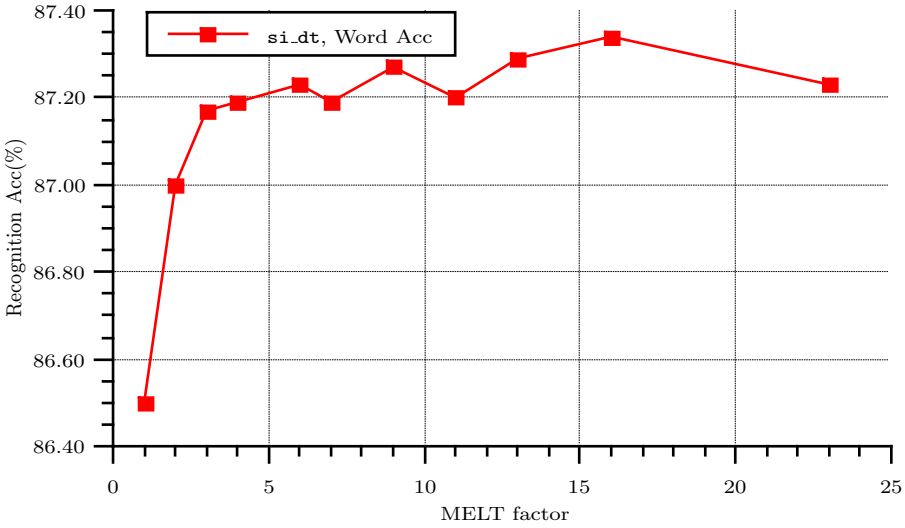


Figure 9.10: The word recognition accuracy on the `si_dt` set after the unsupervised adaptation of the Bottleneck LVCSR system

During the recognition phase, both the `si_et` and the `si_dt` data sets were recognized. The recognized text, together with the associated confidence factors was used to determine the optimal MELT matrices. Using the MELT matrices, the `si_dt` was used to determine the WIP and the LMW values. These values, together with the associated MELT matrices were used to recognize the `si_et` data set.

A single experiment on unsupervised adaptation using the MELT = 23 and the cutoff threshold CF = 0.95 was performed to verify that the MELT approach is viable even in such systems.

9.8.2 Results

First, the performance on the `si_dt` set was evaluated for several values of the MELT factor. See the Fig. 9.10 for the performance graph. Then, the best performing MELT factor (that is MELT = 16) was used and the `si_et` set was adapted using the mentioned MELT factor and recognized using the WIP and LMW values determined beforehand on the `si_dt` set. See the Table 9.14 for the complete performance metrics.

	Acc(%)	$CI_{Acc-}(\%)$	$CI_{Acc+}(\%)$	σ
`si_dt`	87.34	83.88	89.06	5.5
`si_et`	87.43	85.49	88.83	4.5

Table 9.14: The word recognition Acc on the unsupervised SAT task for a Bottleneck LVCSRs system

9.8.3 Conclusion

The experiments performed in this section suggest that MELT normalization can be used even for tasks when the neural network is used as a feature extractor. The speaker adapted system performs about 2.5 % better than the original Bottleneck LVCSR system, which represents approx 16 % reduction of WER. Moreover, the ML or the MAP adaptation of the GMM mixtures could be performed to increase the recognition accuracy furthermore.

9.9 Conclusion

In this chapter, the proposed method for the SAT and the speaker normalization was studied experimentally. The experiments were designed specifically to support the claims of the universality of the proposed method.

The MELT method can be used in the phoneme recognition task as well as in the LVCSR task. The method can be applied in hybrid speech recognition systems, where the neural network is used for posteriori probabilities estimation or in Bottleneck GMM/HMM LVCSR systems where the neural network is used as a features extractor. It is possible to use it in cases where only a limited amount of the speaker-specific data is available (as little as 15 seconds) or in cases, where only speaker utterances without the reference transcript are available.

A wide variety of speech corpora was used. The corpora differ in language, in the amount of data per speaker, number of speakers and in complete overall size of the corpus. The speech recognition systems used as baselines in the scope of this work perform the same or even better than the common (unadapted) systems reported in scientific papers.

This is especially true in the case of phoneme recognizers – the neural networks used for the task of phoneme recognition have a smaller number of parameters and achieve recognition accuracy that is higher than the accuracies reported so far in scientific papers. The reason behind this is presumably the modified algorithm of the neural networks training.

10 Conclusion and Future Work

In this work, a novel adaptation method called MELT (Minimum Error Linear Transform) was introduced. This method can be used as a speaker-normalization method in the field of speaker adaptation and in the field of the speaker adaptive training. The method is based on exploiting the frequency and time dependencies in the LTSP feature vectors fed into the neural network being adapted.

The presented approach establishes a link between the linear transform of the LTSP feature network and the linear transform of the source mel-filter bank log-energies output; however this is not the only possible or the only sensible approach.

To allow for robust functioning, the presented approach provides for a relatively simple way of tweaking the number of free variables to be estimated during the speaker normalization process. The number of free variables can be different for different speakers when the amount of data for each speaker differs significantly.

Additionally, all the original thesis goals were fulfilled.

1. A high-performance software for neural network training in the environment of grid-computing was developed. During the course of the work, systems based on similar principles were developed and introduced by other teams[117, 106, 98], however the training engine developed to fulfill the goals of the work has been tuned specifically for the grid-environment and HW equipment used in the author's department. When the GPU is used, the software is capable of achieving approximately 40× speedup compared to the single threaded CPU functioning of the commonly used QuickNet software[33]. The unique feature of the software is the ability to use all GPU devices found in the given computer. Thus, speedups over factor of 100 are achievable relatively easily.

   Moreover, the system uses a different training approach. Instead of stochastic descent, the software package allows the user to choose between the L-BFGS optimization or the iRPROP optimization. Both these methods converge very quickly. The iRPROP method is especially suitable for the training of neural networks with the bottleneck topology.

2. Hybrid speech recognition systems for Czech and English were built. These systems, used as baselines in this work, achieve the same or better recognition accuracy than the systems presented in peer-reviewed research papers. Moreover, both phoneme recognition systems and word recognition systems were developed.

3. Both semi-supported and completely unsupervised speaker adaptation paradigms were evaluated and tested. The proposed method is applicable in both cases.

4. The proposed method was evaluated on speech corpora of different languages, different sizes, different recording channels and different corpora designs. The method achieves approximately a 12% reduction of the word error rate even for 40 seconds of annotated speech and approximately a 15.4% reduction of the WER when used in a completely unsupervised manner on circa 4 minutes of speech. For higher amounts of speaker-specific data, the improvements could be even larger. The important fact is that the improvements are reported on a real-world system with a strong language model.

5. The L-BFGS method is used for an estimation of the speaker-normalization parameters. These methods converge very fast and thus, the method can be used even for a very large amount of speaker specific data or a very large amount of speakers.

10.1 Future Work

The method and the experiments presented in this work pose a good starting point for further development of the methods, the training system and the hybrid speech recognition systems. The possible research can be done in three interesting directions.

1. Enhance the support for very fast training or training on very large speech corpora (thousands of hours). To achieve this goal, a grid-computing environment is necessary. To achieve acceptable training times, the training software must be distributed – support for training on multiple computers is a must. When using multiple computers, sometimes the scalability of the approach plays a significant role and industrial standards with documented behavior should be prioritized over ad-hoc solutions[62]. The industry standard for inter-computer communication is OpenMPI[35]. The OpenMPI standard and the available software implementations allow for effective scaling of the task. Moreover, the optimization algorithm should be revisited and optimized for this task.

2. Allowing for an automatic discovery of the frequency-time dependencies. The described method is an effective one and has the interesting property of establishing a link between the mel-filterbank outputs and the LTSP feature vectors, however in many cases it would be favorable to allow for an automatic deduction of these dependencies. A method called *Soft weight-sharing*[79, 61] could be used as a suitable starting point for a further study in this area.

3. The actual creation of a system trained on thousands of hours of speech data. From a theoretical point of view, there is no significant difference between training a system on tens or hundreds of hours of speech data and training a system on thousands of hours of speech data (except for increasing the time needed for training the system). From the practical point of view, however, training a system on such an amount of data posses a significant challenge even when the standard GMM/HMM approach is used.

Bibliography

[1] Victor Abrash et al. "Connectionist Speaker Normalization And Adaptation". In: *Proceedings of the Fourth European Conference on Speech Communication and Technology, Eurospeech-1995*. International Speech Communication Association. Sept. 1995, pp. 2183–2186 (cit. on p. 54).

[2] Alejandro Acero. "Acoustical And Environmental Robustness In Automatic Speech Recognition". PhD Thesis. Pittsburgh, Pennsylvania 15213: Carnegie Mellon University Department of Electrical and Computer Engineering, Sept. 1990 (cit. on p. 45).

[3] Tasos Anastasakos and Sreeram V. Balakrishnan. "The Use of Confidence Measures in Unsupervised Adaptation of Speech Recognizers". In: *Proc. Int. Conf. on Spoken Language Processing ICSLP98*. Sydney, Australia, Dec. 1998, pp. 2303 –2306 (cit. on p. 49).

[4] Tesam Anastasakos et al. "A compact model for speaker-adaptive training". In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. Vol. 2. Oct. 1996, 1137 –1140 vol.2. ISBN: 0-7803-3555-4. DOI: 10.1109/ICSLP.1996.607807 (cit. on p. 48).

[5] Christophe Van Bael and Simon King. "The Keyword Lexicon - An accent-independent lexicon for automatic speech recognition". In: *Proc. Int. Congress of Phonetic Sciences*. 2003, pp. 1165–1168 (cit. on p. 88).

[6] Raimo Bakis. "Continuous speech recognition via centisecond acoustic states". In: *The Journal of the Acoustical Society of America* 59.S1 (1976), S97–S97. ISSN: 00014966. DOI: 10.1121/1.2003011 (cit. on p. 11).

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *Neural Networks, IEEE Transactions on* 5.2 (Mar. 1994), pp. 157 –166. ISSN: 1045-9227. DOI: 10.1109/72.279181 (cit. on p. 33).

[8] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3 (Feb. 2003), pp. 1137–1155 (cit. on p. 16).

[9] Maximilian Bisani and Hermann Ney. "Bootstrap estimates for confidence intervals in ASR performance evaluation". In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*. Vol. 1. May 2004, DOI: 10.1109/ICASSP.2004.1326009 (cit. on p. 20).

[10] Christopher M. Bishop. "A fast procedure for retraining the multilayer perceptron". In: *International Journal of Neural Systems* 2.3 (1991), pp. 229–236 (cit. on p. 53).

[11] Christopher M. Bishop. "Exact calculation of the Hessian matrix for the multilayer perceptron". In: *Neural Computa* 4 (4 July 1992), pp. 494–501. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.4.494. URL: http://dl.acm.org/citation.cfm?id=148167.148171 (cit. on p. 53).

[12] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. 1st ed. Oxford University Press, USA, Jan. 18, 1996. ISBN: 9780198538646. URL: http://www.worldcat.org/isbn/0198538642 (cit. on pp. 36, 40).

[13] Christopher M. Bishop. "Training with noise is equivalent to Tikhonov regularization". In: *Neural Comput.* 7 (1 Jan. 1995), pp. 108–116. ISSN: 0899-7667. DOI: 10.1162/neco.1995.7.1.108. URL: http://dl.acm.org/citation.cfm?id=211171.211185 (cit. on p. 40).

[14] Herve Bourlard and Nelson Morgan. "Continuous speech recognition by connectionist statistical methods". In: *Neural Networks, IEEE Transactions on* 4.6 (Nov. 1993), pp. 893 –909. ISSN: 1045-9227. DOI: 10.1109/72.286885 (cit. on p. 13).

[15] John S. Bridle. "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters". In: *Advances in neural information processing systems 2*. Ed. by David S. Touretzky. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990. Chap. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters, pp. 211–217. ISBN: 1-55860-100-7. URL: http://dl.acm.org/citation.cfm?id=109230.109256 (cit. on pp. 35, 40).

[16] Daniel Clark Burnett. "Rapid Speaker Adaptation for Neural Network Speech Recognizers". PhD Thesis. 20000 NW Walker Road Beaverton, OR 97006: Oregon Graduate Institute of Science and Technology, Apr. 1997 (cit. on p. 50).

[17] James Carpenter and John Bithell. "Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians". In: *Statistics in Medicine* 19.9 (2000), pp. 1141–1164. ISSN: 1097-0258. DOI: 10.1002/(SICI)1097-0258(20000515)19:9<1141::AID-SIM479>3.0.CO;2-F. URL: http://dx.doi.org/10.1002/(SICI)1097-0258(20000515)19:9<1141::AID-SIM479>3.0.CO;2-F (cit. on p. 21).

[18] Jan Černocký et al. "Recording of Czech and Slovak Telephone Databases within SpeechDat-E." In: *TSD*. Ed. by Václav Matoušek et al. Vol. 1692. Lecture Notes in Computer Science. Springer, 1999, pp. 388–391. ISBN: 3-540-66494-7. URL: http://dblp.uni-trier.de/db/conf/tsd/tsd1999.html#CernockyPRHT99 (cit. on pp. 63, 76).

[19] Barry Chen, Qifeng Zhu, and Nelson Morgan. "Learning long-term temporal features in LVCSR using neural networks." In: *Proceedings of ICSLP 2004*. Jeju Island, Korea, Oct. 2004 (cit. on pp. 9, 10).

[20] Christopher Cieri et al. *Fisher English Training Speech, Parts 1 and 2*. Linguistic Data Consortium. Philadelphia, 2004 and 2005. ISBN: 1-58563-313-5 (cit. on p. 42).

[21] Ronan Collobert and Samy Bengio. "A gentle Hessian for efficient gradient descent". In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*. Vol. 5. May 2004, DOI: 10.1109/ICASSP.2004.1327161 (cit. on pp. 34, 54).

[22] Neil James Cooke. "Gaze-Contingent Automatic Speech Recognition". PhD Thesis. Department of Electronic, Electrical and Computer Engineering School of Engineering University of Birmingham Birmingham B15 2TT United Kingdom: University of Birmingham Department of Electronic, Electrical and Computer Engineering, Dec. 2006 (cit. on p. 88).

[23] Mario Costa. "Probabilistic interpretation of feedforward network outputs, with relationships to statistical prediction of ordinal quantities". In: *International Journal of Neural Systems (IJNS)* 7.5 (Dec. 1996), pp. 627–638. DOI: 10.1142/S0129065796000610 (cit. on p. 40).

[24] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems (MCSS)* 2 (4 1989), pp. 303–314. ISSN: 0932-4194. DOI: 10.1007/BF02551274 (cit. on p. 13).

[25] Vassilis Digalakis, Dimitry Rtischev, and Leo G. Neumeyer. "Speaker adaptation using constrained estimation of Gaussian mixtures". In: *Speech and Audio Processing, IEEE Transactions on* 3.5 (Sept. 1995), pp. 357 –366. ISSN: 1063-6676. DOI: 10.1109/89.466659 (cit. on p. 47).

[26] Rob A. Dunne and Norm A. Campbell. "On the pairing of the Softmax activation and cross-entropy penalty functions and the derivation of the Softmax activation function". In: *Proc. 8th Aust. Conf. on Neural Networks*. Melbourne, 1997, pp. 181–185 (cit. on p. 35).

[27] Stéphane Dupont and Leila Cheboub. "Fast speaker adaptation of artificial neural networks for automatic speech recognition". In: *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*. Vol. 3. 2000, 1795 –1798 vol.3. DOI: 10.1109/ICASSP.2000.862102 (cit. on pp. 54, 58).

[28] Bradley Efron. "Better Bootstrap Confidence Intervals". English. In: *Journal of the American Statistical Association* 82.397 (1987), pp. 171–185. ISSN: 01621459. URL: http://www.jstor.org/stable/2289144 (cit. on p. 21).

[29] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179 –211. ISSN: 0364-0213. DOI: 10.1016/0364-0213(90)90002-E. URL: http://www.sciencedirect.com/science/article/pii/036402139090002E (cit. on p. 25).

[30] Scott E. Fahlman. "Faster-Learning Variations on Back-Propagation: An Empirical Study". In: *Proceedings, of the 1988 Connectionist Models Summer School*. Ed. by David S. Touretzky, Geoffrey E. Hinton, and Terrence J. Sejnowski. Los Altos CA: San Francisco, CA: Morgan Kaufmann, 1989, pp. 38–51 (cit. on p. 35).

[31] Scott E. Fahlman and Christian Lebiere. "The Cascade-Correlation Learning Architecture". In: *Advances in Neural Information Processing Systems*. Vol. 2. Morgan Kaufmann, 1990, pp. 524–532. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.4325 (cit. on p. 54).

[32] Craig L. Fancourt and José C. Principe. "Optimization in companion search spaces: the case of cross-entropy and the Levenberg-Marquardt algorithm". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on* 2 (2001), pp. 1317–1320. DOI: 10.1109/ICASSP.2001.941168 (cit. on p. 38).

[33] Philipp Färber. *Quicknet on MultiSpert: Fast Parallel Neural Network Training*. Tech. rep. TR-97-047. 1947 Center Street, Berkeley, CA, 94704: Internetional Computer Science Institute, Dec. 1997 (cit. on p. 103).

[34] W. Fisher, G. Doddington, and Goudie K. Marshall. "The DARPA speech recognition research database: Specification and status". In: *Proceedings of the DARPA Speech Recognition Workshop*. 1986, pp. 93–100 (cit. on p. 77).

[35] Edgar Gabriel et al. "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation". In: *Proceedings, 11th European PVM/MPI Users' Group Meeting*. Budapest, Hungary, Sept. 2004, pp. 97–104 (cit. on p. 104).

[36] Mark J.F. Gales. "Maximum likelihood linear transformations for HMM-based speech recognition". In: *Computer Speech & Language* 12.2 (1998), pp. 75–98. URL: http://www.ingentaconnect.com/content/ap/la/1998/00000012/00000002/art00043 (cit. on p. 56).

[37] John Garofalo et al. *Continous Speech Recognition (CSR-I) Wall Street Journal (WSJ0) news*. Complete corpus. Linguistic Data Consortium, 1993 (cit. on pp. 78, 88).

[38] John S. Garofolo et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. CD-ROM. Philadelphia: Linguistic Data Consortium, 1993 (cit. on pp. 63, 77).

[39] Roberto Gemello et al. "Adaptation of Hybrid ANN/HMM Models Using Linear Hidden Transformations and Conservative Training". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 1. May 2006, p. I. DOI: 10.1109/ICASSP.2006.1660239 (cit. on pp. 52, 54).

[40] Roberto Gemello et al. "Linear hidden transformations for adaptation of hybrid ANN/HMM models". In: *Speech Commun.* 49 (10-11 Oct. 2007), pp. 827–835. ISSN: 0167-6393. DOI: 10.1016/j.specom.2006.11.005. URL: http://dl.acm.org/citation.cfm?id=1284914.1285132 (cit. on pp. 52, 54).

[41] Christian Gollan and Michiel Bacchiani. "Confidence scores for acoustic model adaptation". In: *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. Apr. 2008, pp. 4289 –4292. DOI: 10.1109/ICASSP.2008.4518603 (cit. on p. 49).

[42] František Grézl. "TRAP-based Probabilistic Features For Automatic Speech Recognition". PhD Thesis. Brno University of Technology Faculty of Information Technology, 2007 (cit. on p. 10).

[43] František Grézl and Petr Fousek. "Optimizing bottle-neck features for lvcsr". In: *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. Apr. 2008, pp. 4729 –4732. DOI: 10.1109/ICASSP.2008.4518713 (cit. on p. 10).

[44] František Grézl et al. "Probabilistic and Bottle-Neck Features for LVCSR of Meetings". In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 4. Apr. 2007, DOI: 10.1109/ICASSP.2007.367023 (cit. on p. 10).

[45] John B. Hampshire II and Alex Waibel. "The Meta-Pi network: building distributed knowledge representations for robust multisource pattern recognition". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 14.7 (July 1992), pp. 751 –769. ISSN: 0162-8828. DOI: 10.1109/34.142911 (cit. on p. 59).

[46] J. B. Hampshire and B. Pearlmutter. "Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function". In: *Proceedings of the 1990 Connectionist Models Summer School*. Ed. by David S. Touretzky et al. San Mateo, CA: Morgan Kaufmann, 1990, pp. 159–172 (cit. on p. 40).

[47] Babak Hassibi, David G. Stork, and Gregory J. Wolff. "Optimal Brain Surgeon and general network pruning". In: *Neural Networks, 1993., IEEE International Conference on*. 1993, 293 –299 vol.1. DOI: 10.1109/ICNN.1993.298572 (cit. on p. 52).

[48] Hynek Hermansky and Nelson Morgan. "RASTA processing of speech". In: *Speech and Audio Processing, IEEE Transactions on* 2.4 (Oct. 1994), pp. 578 –589. ISSN: 1063-6676. DOI: 10.1109/89.326616 (cit. on p. 8).

[49] Josef Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen". Diploma thesis. Institut für Informatik, Technische Universität München, 1991 (cit. on p. 26).

[50] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.2 (1998), pp. 107–116. DOI: http://dx.doi.org/10.1142/S0218488598000094 (cit. on p. 26).

[51] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computations* 9 (8 Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dl.acm.org/citation.cfm?id=1246443.1246450 (cit. on p. 26).

[52] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359 –366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8. URL: http://www.sciencedirect.com/science/article/pii/0893608089900208 (cit. on p. 13).

[53] Chang-Wen Hsu and Lin-Shan Lee. "Higher Order Cepstral Moment Normalization for Improved Robust Speech Recognition". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 17.2 (Feb. 2009), pp. 205 –220. ISSN: 1558-7916. DOI: 10.1109/TASL.2008.2006575 (cit. on p. 43).

[54] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, May 5, 2001. ISBN: 978-0130226167. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0130226165 (cit. on p. 46).

[55] Xuedong D. Huang, K. F. Lee, and A. Waibel. "Connectionist speaker normalization and its applications to speech recognition". In: *Neural Networks for Signal Processing [1991]., Proceedings of the 1991 IEEE Workshop*. Oct. 1991, pp. 357 –366. DOI: 10.1109/NNSP.1991.239506 (cit. on p. 59).

[56] Raymond Hubbard and M. J. Bayarri. *P-Values are not Error Probabilities*. published electronicaly. Nov. 2003. URL: http://www.uv.es/sestio/TechRep/tr14-03.pdf (cit. on p. 20).

[57] J. J. Humphries and P. C. Woodland. "The use of accent-specific pronunciation dictionaries in acoustic model training". In: *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. Vol. 1. May 1998, pp. 317 –320. DOI: 10.1109/ICASSP.1998.674431 (cit. on p. 88).

[58] Christian Igel and Michael Hüsken. "Empirical evaluation of the improved Rprop learning algorithms". In: *Neurocomputing* 50 (2003), pp. 105–123. ISSN: 0925-2312. DOI: 10.1016/S0925-2312(01)00700-7. URL: http://linkinghub.elsevier.com/retrieve/pii/S0925231201007007 (cit. on p. 35).

[59] F. Jelinek. "Continuous speech recognition by statistical methods". In: *Proceedings of the IEEE* 64.4 (Apr. 1976), pp. 532 –556. ISSN: 0018-9219. DOI: 10.1109/PROC.1976.10159 (cit. on p. 1).

[60] Michael Kirby and Lawrence Sirovich. "Application of the Karhunen-Loeve procedure for the characterization of human faces". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 12.1 (Jan. 1990), pp. 103 –108. ISSN: 0162-8828. DOI: 10.1109/34.41390 (cit. on p. 57).

[61] Fatih Köksal, Ethem Alpaydyn, and Günhan Dündar. "Weight Quantization for Multi-layer Perceptrons Using Soft Weight Sharing". In: *Artificial Neural Networks — ICANN 2001*. Ed. by Georg Dorffner, Horst Bischof, and Kurt Hornik. Vol. 2130. Lecture Notes in Computer Science. Springer Berlin / Heidelberg,

2001, pp. 211–216. ISBN: 978-3-540-42486-4. DOI: 10.1007/3-540-44668-0_30 (cit. on p. 104).

[62] Stanislav Kontár. "Parallel training of neural networks for speech recognition". In: *Proc. 12th International Conference on Soft Computing MENDEL'06*. Brno, CZ: Brno University of Technology, 2006, p. 6. ISBN: 80-214-3195-4. URL: http://www.fit.vutbr.cz/research/view_pub.php?id=8180 (cit. on p. 104).

[63] Roland Kuhn et al. "Rapid speaker adaptation in eigenvoice space". In: *Speech and Audio Processing, IEEE Transactions on* 8.6 (Nov. 2000), pp. 695 –707. ISSN: 1063-6676. DOI: 10.1109/89.876308 (cit. on p. 58).

[64] R. Kuhn et al. "Fast speaker adaptation using a priori knowledge". In: *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*. Vol. 2. Mar. 1999, 749 –752 vol.2. DOI: 10.1109/ICASSP.1999.759776 (cit. on p. 58).

[65] Hugo Larochelle et al. "Exploring Strategies for Training Deep Neural Networks". In: *J. Mach. Learn. Res.* 10 (June 2009), pp. 1–40. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1577069.1577070 (cit. on p. 34).

[66] Yann LeCun, John S. Denker, and Sara A. Solla. "Optimal brain damage". In: *Advances in neural information processing systems 2*. Ed. by David S. Touretzky. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990. Chap. Optimal brain damage, pp. 598–605. ISBN: 1-55860-100-7. URL: http://dl.acm.org/citation.cfm?id=109230.109298 (cit. on p. 52).

[67] Y. LeCun et al. "Efficient BackProp". In: *Neural Networks: Tricks of the trade*. Ed. by G. Orr and Muller K. Vol. 1524. Lecture Notes in Computer Science. Springer, 1998. ISBN: 3-540-65311-2 (cit. on p. 34).

[68] Kai-Fu Lee and Hsiao-Wuen Hon. "Speaker-independent phone recognition using hidden Markov models". In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 37.11 (Nov. 1989), pp. 1641 –1648. ISSN: 0096-3518. DOI: 10.1109/29.46546 (cit. on pp. 78, 79).

[69] Vladimir I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet Physics Doklady* 10.8 (1966), pp. 707–710 (cit. on p. 18).

[70] Bo Li and Khe Chai Sim. "Comparison of Discriminative Input and Output Transformations for Speaker Adaptation in the Hybrid NN/HMM Systems". In: *Eleventh Annual Conference of the International Speech Communication Association*. Makuhari, Chiba, Japan, Sept. 2010, pp. 526–529. URL: http://www.isca-speech.org/archive/interspeech_2010/i10_0526.html (cit. on pp. 55–57).

[71] Carla Lopes and Fernando Perdigao. *Phoneme Recognition on the TIMIT Database*. Ed. by Ivo Ipsic. InTech, 2011. ISBN: 978-953-307-996-7. URL: http://www.intechopen.com/articles/show/title/phoneme-recognition-on-the-timit-database (cit. on p. 78).

[72] V. Mazya and G. Schmidt. "On approximate approximations using Gaussian kernels". In: *IMA Journal of Numerical Analysis* 16.1 (1996), pp. 13–29. DOI: 10.1093/imanum/16.1.13. URL: http://imajna.oxfordjournals.org/content/16/1/13.abstract (cit. on p. 13).

[73] J. McDonough et al. "Speaker-adapted training on the Switchboard Corpus". In: *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*. Vol. 2. Apr. 1997, 1059 –1062 vol.2. DOI: 10.1109/ICASSP.1997.596123 (cit. on p. 48).

[74] T. Mikolov et al. "Neural network based language models for highly inflective languages". In: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. Apr. 2009, pp. 4725 –4728. DOI: 10.1109/ICASSP.2009.4960686 (cit. on p. 17).

[75] Marvin L. Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry, Expanded Edition*. The MIT Press, Dec. 1987. ISBN: 978-0-262-63111-2 (cit. on p. 27).

[76] Nikki Mirghafori, Eric Fosler, and Nelson Morgan. "Towards robustness to fast speech in ASR". In: *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*. Vol. 1. May 1996, 335 –338 vol. 1. DOI: 10.1109/ICASSP.1996.541100 (cit. on p. 89).

[77] Joao Neto et al. "Speaker-Adaptation for Hybrid HMM-ANN Continuous Speech Recognition System". In: *Proceedings of the Fourth European Conference on Speech Communication and Technology, Eurospeech-1995*. International Speech Communication Association, 1995, pp. 2171–2174. URL: http://hdl.handle.net/1842/1274 (cit. on pp. 50, 52, 54).

[78] Hermann Ney. "On the probabilistic interpretation of neural network classifiers and discriminative training criteria". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17.2 (Feb. 1995), pp. 107 –119. ISSN: 0162-8828. DOI: 10.1109/34.368176 (cit. on p. 40).

[79] Steven J. Nowlan and Geoffrey E. Hinton. "Simplifying neural networks by soft weight-sharing". In: *Neural Comput.* 4 (4 July 1992), pp. 473–493. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.4.473. URL: http://dl.acm.org/citation.cfm?id=148167.148169 (cit. on p. 104).

[80] Dong C. Park, Mohamed A. El-Sharkawi, and Robert J. Marks II. "An adaptively trained neural network". In: *Neural Networks, IEEE Transactions on* 2.3 (May 1991), pp. 334 –345. ISSN: 1045-9227. DOI: 10.1109/72.97910 (cit. on p. 61).

[81] Junho Park et al. "Training and adapting MLP features for Arabic speech recognition". In: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. Apr. 2009, pp. 4461 –4464. DOI: 10.1109/ICASSP.2009.4960620 (cit. on p. 55).

[82] C. E. Pedreira and N. M. Roehl. "On adaptively trained neural networks". In: *Neural Networks, 1993. IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference on*. Vol. 1. Oct. 1993, 565 –568 vol.1. DOI: 10.1109/IJCNN.1993.713978 (cit. on p. 61).

[83] Kaare Brandt Petersen and Michael Syskind Pedersen. *The Matrix Cookbook*. Nov. 14, 2008. URL: http://orion.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf (cit. on p. 71).

[84] Johann Pfanzagl. *Parametric Statistical Theory*. Ed. by R. Hamböker. De Gruyter Textbook. Berlin, Germany: Walter de Gruyter & Co, Aug. 1994, p. 387 (cit. on p. 73).

[85] Michael Pitz and Hermann Ney. "Vocal Tract Normalization Equals Linear Transformation in Cepstral Space". In: *Speech and Audio Processing, IEEE Transactions on* 13.5 (Sept. 2005), pp. 930 –944. ISSN: 1063-6676. DOI: 10.1109/TSA.2005.848881 (cit. on p. 47).

[86] Aleš Pražák. "Confidence Measures in Real-Time Large Vocabulary Continuous Speech Recognition Systems". PhD Thesis. Universit of West Bohemia Faculty of Applied Sciences, Oct. 2008 (cit. on p. 15).

[87] Josef Psutka et al. *Mluvíme s počítačem česky*. Academia, 2006, p. 746. ISBN: 80-200-1309-1 (cit. on p. 11).

[88] Lawrence R. Rabiner, Steve E. Levinson, and Mohan M. Sondhi. "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition". In: *Bell System Technical Journal* 62.4 (1983), pp. 1075–1105 (cit. on p. 11).

[89] Roger Ratcliff. "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." In: *Psychological Review* 97.2 (1990), pp. 285–308. DOI: 10.1037/0033-295X.97.2.285. URL: http://www.ncbi.nlm.nih.gov/pubmed/2186426 (cit. on p. 51).

[90] Michael D. Richard and Richard P. Lippmann. "Neural Network Classifiers Estimate Bayesian a posteriori Probabilities". In: *Neural Computations* 3.4 (Dec. 1991), pp. 461–483. ISSN: 0899-7667. DOI: 10.1162/neco.1991.3.4.461 (cit. on pp. 40, 41).

[91] Martin Riedmiller. *Rprop - Description and Implementation Details*. Technical Report. W-76128 Karlsruhe: Institut für Logik, Komplexität und Deduktionssysteme, University of Karlsruhe, Jan. 1994 (cit. on p. 35).

[92] Eric Sven Ristad. "A natural law of succession". In: *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*. Aug. 1998, p. 445. DOI: 10.1109/ISIT.1998.709050 (cit. on pp. 77–79).

[93] Anthony Robins. "Catastrophic forgetting in neural networks: the role of rehearsal mechanisms". In: *Artificial Neural Networks and Expert Systems, 1993. Proceedings., First New Zealand International Two-Stream Conference on*. Nov. 1993, pp. 65 –68. DOI: 10.1109/ANNES.1993.323080 (cit. on p. 51).

[94] Anthony Robins. "Catastrophic Forgetting, Rehearsal and Pseudorehearsal". In: *Connection Science* 7.2 (1995), pp. 123–146 (cit. on p. 51).

[95] Tony Robinson. *SHORTEN: Simple Lossless and near-lossless waveform compression*. Technical report CUED/F-INFENG/TR.156. Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, UK: Cambridge University Engineering Department, Dec. 1994 (cit. on pp. 77, 79).

[96] Tony Robinson et al. "WSJCAM0: A Brittish English Speech Corpus for Large Vocabulary Continuous Speech Recognition". In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Vol. 1. IEEE Computer Society, May 1995, pp. 81–84. DOI: 10.1109/ICASSP.1995.479278 (cit. on pp. 63, 87).

[97] S. Scanzio et al. "Adaptation of Hybrid ANN/HMM Using Weights Interpolation". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 5. May 2006, p. V. DOI: 10.1109/ICASSP.2006.1661455 (cit. on pp. 54, 56, 57).

[98] S. Scanzio et al. "Parallel implementation of artificial neural network training". In: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. Mar. 2010, pp. 4902 –4905. DOI: 10.1109/ICASSP.2010.5495108 (cit. on p. 103).

[99] Petr Schwarz. "Phoneme recognition based on long temporal context". PhD thesis. Brno, CZ: Brno University of Technology, 2009, p. 95. URL: http://www.fit.vutbr.cz/research/view_pub.php?id=9132 (cit. on pp. 10, 63).

[100] Petr Schwarz, Pavel Matějka, and Jan Černocký. "Hierarchical Structures of Neural Networks for Phoneme Recognition". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 1. May 2006, p. I. DOI: 10.1109/ICASSP.2006.1660023 (cit. on p. 85).

[101] Holger Schwenk and Jean-Luc Gauvain. "Training neural network language models on very large corpora". In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, 2005, pp. 201–208. DOI: 10.3115/1220575.1220601 (cit. on p. 17).

[102] The Editors of Scientific American. *The Scientific American Book of the Brain*. The Lyons Press, 2001. ISBN: 978-1585742851 (cit. on p. 22).

[103] Yi Shang and B.W. Wah. "Global optimization for neural network training". In: *Computer* 29.3 (Mar. 1996), pp. 45 –54. ISSN: 0018-9162. DOI: 10.1109/2.485892 (cit. on p. 34).

[104] Sangita R. Sharma. "Multi-stream approach to robust stream recognition". PhD Thesis. Oregon Graduate Institute of Science and Technology, Oct. 1999 (cit. on pp. 8, 9).

[105] Sidney Siegel and N. John Castellan. *Nonparametric statistics for the behavioral sciences*. 2nd ed. McGraw-Hill Humanities/Social Sciences/Languages, Jan. 1988. ISBN: 978-0070573574 (cit. on p. 19).

[106] X. Sierra-Canto, F. Madera-Ramirez, and V. Uc-Cetina. "Parallel Training of a Back-Propagation Neural Network Using CUDA". In: *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*. Dec. 2010, pp. 307 –312. DOI: 10.1109/ICMLA.2010.52 (cit. on p. 103).

[107] Sara A. Solla, Esther Levin, and Michael Fleisher. "Accelerated learning in layered neural networks". In: *Complex Systems* 2 (6 Dec. 1988), pp. 625–639. ISSN: 0891-2513. URL: http://dl.acm.org/citation.cfm?id=65512.65513 (cit. on p. 29).

[108] J. Stadermann and G. Rigoll. "Two-Stage Speaker Adaptation of Hybrid Tied-Posterior Acoustic Models". In: *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*. Vol. 1. Mar. 2005, pp. 977 –980. DOI: 10.1109/ICASSP.2005.1415279 (cit. on p. 52).

[109] Fritz Stager and Mukul Agarwal. "Three Methods to Speed up the Training of Feedforward and Feedback Perceptrons". In: *Neural Networks* 10.8 (1997), pp. 1435 –1443. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(97)00053-1. URL: http://www.sciencedirect.com/science/article/pii/S0893608097000531 (cit. on p. 38).

[110] Robert G. D. Steele and James H. Torrie. *Principles and procedures of statistics*. McGraw-Hill, 1960 (cit. on p. 93).

[111] Nikko Ström. "Speaker adaptation by modeling the speaker variation in a continuous speech recognition system". In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. Vol. 2. Oct. 1996, 989 –992 vol.2. DOI: 10.1109/ICSLP.1996.607769 (cit. on p. 60).

[112] Roberto Togneri, Aik M. Toh, and Sven Nordholm. "Evaluation and modification of cepstral moment normalization for speech recognition in additibe Babble ensemble". In: *Proceedings SST*. 2006, pp. 94–99 (cit. on p. 43).

[113] E. Trentin and M. Gori. "Robust combination of neural networks and hidden Markov models for speech recognition". In: *Neural Networks, IEEE Transactions on* 14.6 (Nov. 2003), pp. 1519 –1531. ISSN: 1045-9227. DOI: 10.1109/TNN.2003.820838 (cit. on p. 13).

[114] Matthew Turk and Alex Pentland. "Eigenfaces for Recognition". In: *Journal of Cognitive Neuroscience* 3.1 (Dec. 1991), pp. 71–86. DOI: 10.1162/jocn.1991.3.1.71 (cit. on p. 57).

[115] Jan Vaněk. "Discriminative training of acoustic models". PhD Thesis. University of West Bohemia Department of Cybernetics, 2009 (cit. on p. 16).

[116] Jan P. Verhasselt and Jean-Pierre Martens. "A fast and reliable rate of speech detector". In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. Vol. 4. Oct. 1996, 2258 –2261 vol.4. DOI: 10.1109/ICSLP.1996.607256 (cit. on p. 89).

[117] Karel Veselý, Lukáš Burget, and František Grézl. "Parallel Training of Neural Networks for Speech Recognition". In: *Text, Speech and Dialogue*. Ed. by Petr Sojka et al. Vol. 6231. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pp. 439–446. ISBN: 978-3-642-15759-2. DOI: 10.1007/978-3-642-15760-8_56 (cit. on p. 103).

[118] Michael N. Vrahatis, George D. Magoulas, and Vassilis P. Plagianakos. "Globally Convergent Modification of the Quickprop Method". In: *Neural Processing Letters* 12 (2 Oct. 2000), pp. 159–170. ISSN: 1370-4621. DOI: 10.1023/A:1009661729970. URL: http://dl.acm.org/citation.cfm?id=361131.361157 (cit. on p. 35).

[119] H. Wakita. "Normalization of vowels by vocal-tract length and its application to vowel identification". In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 25.2 (Apr. 1977), pp. 183 –192. ISSN: 0096-3518. DOI: 10.1109/TASSP.1977.1162929 (cit. on p. 44).

[120] Guangsen Wang and Khe Chai Sim. "Sequential Classification Criteria for NNs in Automatic Speech Recognition". In: *Proceedings of the Interspeech 2011 Conference*. Ed. by Piero Cosi et al. Causal Production Pty Ltd, Sept. 2011, pp. 441–444 (cit. on p. 87).

[121] R. L. Watrous. "Speaker normalization and adaptation using second-order connectionist networks". In: *Neural Networks, IEEE Transactions on* 4.1 (Jan. 1993), pp. 21 –30. ISSN: 1045-9227. DOI: 10.1109/72.182692 (cit. on p. 60).

[122] Paul J. Werbos. "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences." PhD thesis. Harvard University, 1974 (cit. on p. 29).

[123] Qin Yan and Saeed Vaseghi. "A comparative analysis of UK and US English accents in recognition and synthesis". In: *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. Vol. 1. May 2002, pp. 413 –416. DOI: 10.1109/ICASSP.2002.5743742 (cit. on p. 88).

[124] Steve J. Young et al. *The HTK Book, version 3.4*. Cambridge, UK: Cambridge University Engineering Department, 2006 (cit. on p. 8).

Authored or Co-authored Works

[1] R. Hippman et al. "Voice-supported Electronic Health Record for Temporomandibular Joint Disorders". In: *Methods of Information in Medicine* 49 (2010), pp. 168–172. ISSN: 0026-1270. URL: http://www.kky.zcu.cz/en/publications/HippmanR_2010_Voice-supported.

[2] J. Matoušek et al. "Identification and Automatic Detection of Parasitic Speech Sounds". In: *INTERSPEECH 2009, proceedings of 10th Annual Conference of International Speech Communication Association*. Brighton, Great Britain: ISCA, 2009, pp. 876–879. URL: http://www.kky.zcu.cz/en/publications/MatousekJ_2009_Identificationand.

[3] Miroslav Nagy et al. "Voice-controlled Data Entry in Dental Electronic Health Record". In: *Studies in Health Technology and Informatics* 2008 (2008), pp. 529–534. ISSN: 0926-9630. URL: http://www.kky.zcu.cz/en/publications/NagyMiroslav_2008_Voice-controlledData.

[4] Luboš Šmídl and Jan Trmal. "Keyword Spotting Result Post-processing to Reduce False Alarms". In: *Recent Advances in Signals ans Systems*. Vol. 9. Budapest: WSEAS Press, 2009, pp. 49–52. ISBN: 978-960-474-114-4. URL: http://www.kky.zcu.cz/en/publications/SmidlLubos_2009_KeywordSpotting.

[5] Jan Trmal, Jan Zelinka, and Luděk Müller. "Adaptation of a Feedforward Artificial Neural Network Using a Linear Transform". In: *Text, Speech and Dialogue*. Lecture Notes in Computer Science 6231 (2010), pp. 423–430. ISSN: 0302-9743. URL: http://www.kky.zcu.cz/en/publications/TrmalJan_2010_Adaptationof.

[6] Jan Trmal, Jan Zelinka, and Luděk Müller. "On Speaker Adaptive Training of Artificial Neural Networks". In: *Proceedings of Int. Conf. Interspeech 2010*. 2010. URL: http://www.kky.zcu.cz/en/publications/TrmalJan_2010_OnSpeakerAdaptive.

[7] Jan Trmal et al. "Feature Space Transforms for Czech Sign-Language Recognition". In: *Proceedings of the 9th Annual Conference of the International Speech Communication Association (Interspeech 2008)*. Causal Production Pty ltd., 2008, pp. 2036–2039. URL: http://www.kky.zcu.cz/en/publications/TrmalJan_2008_FeatureSpace.

[8] Jan Trmal et al. "Online TV captioning of Czech Parliamentary Sessions". In: *Text, Speech and Dialogue*. Vol. 6231. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pp. 416–422. ISBN: 3-642-15759-9. URL: http://www.kky.zcu.cz/en/publications/TrmalJan_2010_OnlineTVcaptioning.

[9] J. Trmal et al. "Comparison between GMM and decision graphs based silence/speech detection method". In: *Proceedings of the 11th international conference "Speech and computer" SPECOM'2006*. St. Petersburg: Anatolya Publishers, 2006, pp. 376–379. ISBN: 5-7452-0074-X. URL: http://www.kky.zcu.cz/en/publications/TrmalJ_2006_Comparisonbetween.

[10] J. Trmal et al. "Independent components for acoustic modeling". In: *Proceedings of Int. Conf. Interspeech 2006* 1 (2006), pp. 2486–2489. ISSN: 1990-9772. URL: http://www.kky.zcu.cz/en/publications/TrmalJ_2006_Independent.

[11] J. Trmal et al. "Silence/speech detection method based on set of decision graphs". In: *Lecture Notes in Artificial Intelligence*. 4188th ser. (2006), pp. 539–546. URL: http://www.kky.zcu.cz/en/publications/TrmalJ_2006_Silencespeech.

[12] Jan Vaněk et al. "Optimization of the Gaussian Mixture Model Evaluation on GPU". In: *Proceedings of Int. Conf. Interspeech 2011*. Vol. 1. International Speech Communication Association. Sept. 2011, pp. 1737–1740.

[13] Jan Vaněk et al. "Training of Speaker-Clustered Discriminative Acoustic Models for Use in Real-Time Recognizers". In: *Speech Processing*. Prague: Institute of Photonics and Electronics AS CR, 2010, pp. 152–158. ISBN: 978-80-86269-21-4. URL: http://www.kky.zcu.cz/en/publications/VanekJan_2010_Trainingof.

[14] Jan Zelinka, Jan Trmal, and Müller Luděk. "Low-dimensional Space Transforms of Posteriors in Speech Recognition". In: *Proceedings of Int. Conf. Interspeech 2010*. Vol. 2010. Makuhari, Chiba, Japan: Curran Associates, 2010, pp. 1193–1196. ISBN: 978-1-61782-123-3. URL: http://www.kky.zcu.cz/en/publications/ZelinkaJan_2010_Low-dimensionalSpace.

[15] Jan Zelinka et al. "Posterior Estimates and Transforms for Speech Recognition". In: *Lecture Notes in Artificial Intelligence* 2010 (2010), pp. 480–487. ISSN: 0302-9743. URL: http://www.kky.zcu.cz/en/publications/ZelinkaJan_2010_PosteriorEstimates.

[16] Jana Zvárová et al. "Biderectional Voice Interaction with Dental Electronic Health Record". In: *Med-e-Tel 2008* 2008 (2008), pp. 289–293. ISSN: 1818-9334. URL: http://www.kky.zcu.cz/en/publications/ZvarovaJana_2008_BiderectionalVoice.