# Optimization of Discriminative Models for Speech and Handwriting Recognition

Von der Fakultät für
Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
genehmigte Dissertation

vorgelegt von

Dipl.-Math. Simon Bernhard Wiesler
aus Ostfildern-Ruit

Berichter: Univ.-Prof. Dr.-Ing. Hermann Ney
Univ.-Prof. Dr.-Ing. Gerhard Rigoll

Tag der mündlichen Prüfung: 16. Dezember 2016

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

# Erklärung

Hiermit versichere ich, dass ich die vorliegende Doktorarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Aachen, March 14, 2017
Dipl.-Math. Simon Wiesler

# Abstract

Conventional speech recognition systems are based on Gaussian hidden Markov models. These systems are typically first trained generatively, i.e. a model of the acoustic signal is learned. In a subsequent discriminative training step, the models are fine-tuned to directly optimize the classifier. More recently, it has been found that neural network-based speech recognition systems outperform Gaussian mixture systems. Neural networks as considered in this work are discriminative models, i.e. they do not require a generative training step. Learning their parameters from data is a high-dimensional optimization problem. This optimization problem is the central topic of this thesis. Further contributions cover different aspects of modeling and training, such as generalization ability, model structure, and training criteria. The generality of our methods is confirmed by transferring them from speech to handwriting recognition.

In the first part of this thesis, we study a sub-class of neural networks, known as log-linear models. Because of their shallow structure, their training is a convex optimization problem. Our experiments show that this conceptually simple approach already reaches performance comparable to that of a discriminatively trained Gaussian mixture system. Furthermore, a theoretical convergence analysis of log-linear training is presented.

The second part of the thesis deals with deep neural networks. First, the feasibility of a recently proposed second-order batch optimization algorithm for large-scale tasks is investigated. Motivated by these results, a novel stochastic second-order optimization algorithm for neural network training is developed. This algorithm is capable of optimizing bottleneck networks from scratch. This allows for reducing the size of the models considerably, thereby accelerating both the training and evaluation of the networks. Furthermore, the bottleneck structure acts as a regularization method, thus the accuracy of the models is improved. Another contribution of this thesis is an investigation of sequence-discriminative training of neural networks, which in particular confirms the benefit of the bottleneck structure in combination with this method. Finally, we describe the neural network training tool, which has been implemented within the scope of this work as part of the the publicly available RWTH Aachen speech recognition toolkit.

# Zusammenfassung

Konventionelle Spracherkennungssysteme basieren auf Gaußschen Hidden Markov Modellen. Diese Systeme werden typischerweise zuerst generativ trainiert, d.h. sie lernen ein Modell des akkustischen Signals. In einem nachfolgenden diskriminativen Trainingsschritt wird direkt der Klassifikator optimiert. Neuerdings ist bekannt, dass Spracherkennungssysteme basierend auf neuronalen Netzen konventionellen Systemen überlegen sind. Neuronale Netze, so wie sie in dieser Arbeit verwendet werden, sind diskriminative Modelle, d.h. sie benötigen keinen generativen Trainingsschritt. Das Lernen der Modellparameter aus Daten ist ein hochdimensionales Optimierungsproblem, welches das zentrale Thema dieser Arbeit ist. Weitere wissenschaftliche Beiträge beschäftigen sich mit verschiedenen Aspekten von Modellierung und Training, insbesondere Generalisierbarkeit, Modellstruktur und Trainingskriterium. Die Allgemeingültigkeit der Ergebnisse dieser Arbeit wird durch deren Übertragung von Sprach- auf Handschrifterkennungsprobleme bestätigt.

Im ersten Teil dieser Arbeit werden log-lineare Modelle, eine Unterklasse der neuronalen Netze, untersucht. Aufgrund ihrer flachen Struktur ist ihr Training ein konvexes Optimierungsproblem. Wie unsere Experimente zeigen, erreicht schon dieser einfache Ansatz die Resultate eines diskriminativ trainierten Gaußschen Mischverteilungssystems. Des Weiteren wird eine theoretische Konvergenzanalyse des log-linearen Trainingsproblems hergeleitet.

Der zweite Teil der Arbeit behandelt tiefe neuronale Netze. Zunächst wird die Machbarkeit eines kürzlich vorgeschlagenen Optimierungsalgorithmus zweiter Ordnung untersucht. Motiviert von diesen Ergebnissen wird ein neuer stochastischer Algorithmus entwickelt. Mit diesem Algorithmus können neuronale Netze mit einer Flaschenhalsstruktur (engl. bottleneck) direkt trainiert werden. Dieser Ansatz ermöglicht eine starke Reduktion der Modellgröße und somit eine Beschleunigung des Trainings und der Auswertung des Modells in der Erkennung. Darüberhinaus wirkt die Modellstruktur als Regularisierung, wodurch die Erkennungsfehlerrate des Modells verbessert wird. Ein weiterer Beitrag dieser Arbeit ist eine Untersuchung des diskriminativem Trainings auf Sequenzebene. Insbesondere bestätigt sich der Vorteil der Flaschenhalsstruktur auch in Kombination mit diesem Verfahren.

Abschließend beschreiben wir eine Software für neuronale Netze, die im Rahmen dieser Arbeit als Teil des frei verfügbaren Spracherkennungssystems der RWTH Aachen implementiert wurde.

# Acknowledgment

At this point, I would like to express my gratitude to all the people who supported and accompanied me during the progress of this work. In particular, I would like to thank the following people:
Prof. Dr.-Ing. Hermann Ney for giving me the opportunity for doing research in this interesting area as well as his advice and support.
Prof. Dr.-Ing. Gerhard Rigoll who kindly accepted to review this thesis.
Dr. Ralf Schlüter for his advice and many helpful discussions.
Jens Ackermann, Pavel Golik, and Albert Zeyer for proof-reading this thesis.
Georg Heigold, who mentored me in the beginning of my time at i6 and introduced me to the concepts of discriminative modeling.
Björn Hoffmeister and Christian Plahl for introducing me to the practical aspects of developing a speech recognition system.
Yifan Gong, Jinyu Li, Frank Seide, Jian Xue, Dong Yu and colleagues for the friendly atmosphere and the support during my internship at Microsoft.
Alexander Richard, with whom I collaborated for several topics of this thesis.
Philippe Dreuw and Patrick Dötsch for help with the handwriting recognition system.
My office mates Christian Plahl, Martin Sundermeyer, and Albert Zeyer for the good atmosphere and good discussions about our work.
All my colleagues for the good times we had during working time and in our free time, including Pavel Golik, Stefan Hahn, Matthias Huck, Saab Mansour, Markus Nußbaum-Thom, David Nolden, David Rybach, Daniel Stein, Zoltán Tüske, and David Vilar. Thanks for the good teamwork and the fun we had – it was a great time with you!
My parents for always supporting me and giving me all the chances I had.
The biggest thank-you goes to my little family. My wife Tine for her love, her encouragement, and her support during all these years. And my two wonderful daughters: our little one Lotta and Paula, now four years old, who missed me on the many weekends when I was finishing this thesis. Thank you!

# Outline

# Chapter 1

# Introduction

Speech is the most common and natural way of human communication. This makes automatic speech recognition the natural choice for a human-machine interface. Recently, automatic speech recognition has become an important component of many commercial products such as mobile phones, dictation systems, and many more. Nowadays, large amounts of audio and video data are recorded and stored. Automatic speech recognition makes these data accessible for further natural language processing such as indexing, translation, or information extraction.

The speech recognition problem is defined as the task of converting an acoustic signal, which contains speech, to written text. Today's large vocabulary continuous speech recognition systems handle complete utterances of unconstrained speech. The *word error rate*, i.e. the Levenshtein distance between the correct and the recognized word sequence, is used as the evaluation measure for speech recognition systems.

Modern speech recognition systems are based on the statistical approach. Given a sequence of acoustic vectors representing the speech signal, the most likely word sequence according to a statistical model is chosen. Conventionally, this statistical model is decomposed into a *language model* and an *acoustic model*. The parameters of the models are learned from data in the *training* phase of the system.

Traditionally, the acoustic model has been represented by generative models. Discriminative models have the advantage of directly optimizing the posterior probability required in recognition. These models typically have a huge number of parameters. Learning these parameters from data is a high-dimensional optimization problem. In this thesis, we explore discriminative models both with a shallow structure (log-linear models) and a hierarchical structure (neural networks). The central topic of this thesis is the optimization problem, but related topics such as generalization ability, model structure, and training criteria are studied as well. The same methods and software as in speech recognition can be employed to build state-of-the-art handwriting recognition systems, which differ only in the feature extraction. We evaluate our methods on speech as well as handwriting recognition tasks.

In the remainder of this chapter, the statistical approach to speech recognition is reviewed and the general notation used in this thesis is introduced. Related work is discussed in Section 1.10.

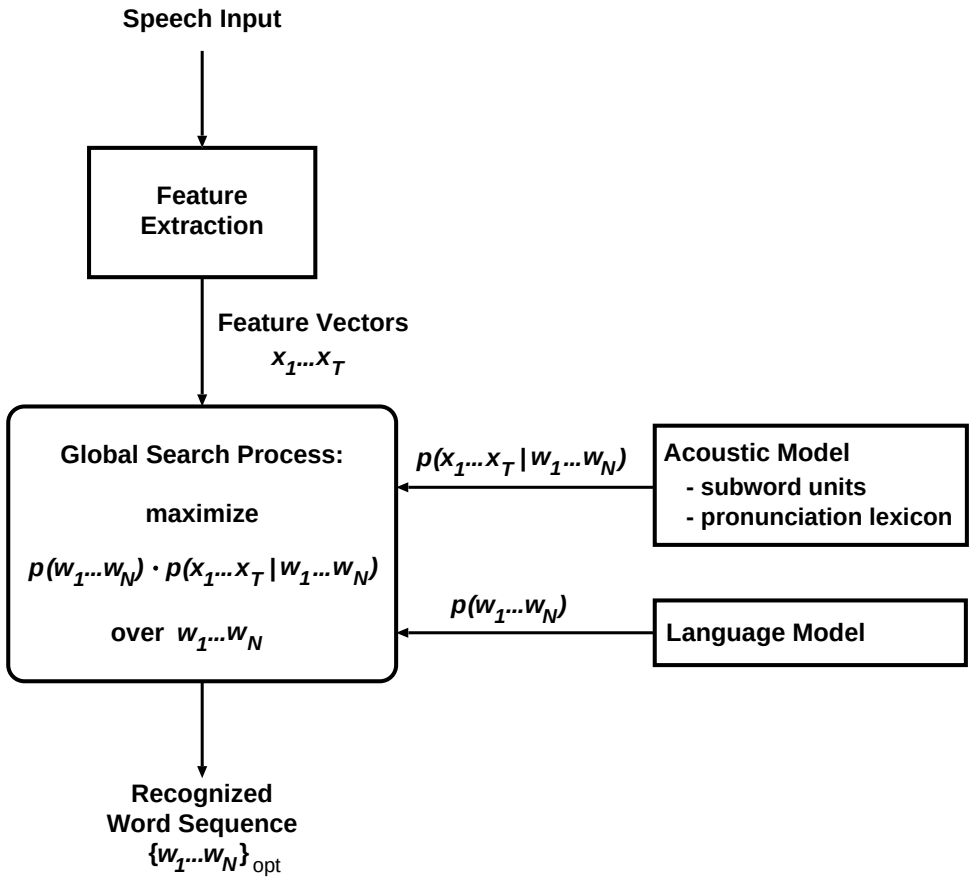


**Figure 1.1.** Basic architecture of a statistical speech recognition system [Ney 90]

## 1.1 Statistical Speech Recognition

Modern speech recognition systems are based on the statistical approach. Given a sequence of acoustic vectors $x_1^T = (x_1, \ldots, x_T)$, Bayes' decision rule [Bayes 63] states that the word sequence $\hat{w}_1^{\hat{N}} = (\hat{w}_1, \ldots, \hat{w}_{\hat{N}})$ which maximizes the *posterior probability* should be chosen:

$$\hat{w}_1^{\hat{N}} = \underset{w_1^N}{\operatorname{argmax}}\, p(w_1^N | x_1^T) \tag{1.1}$$

$$= \underset{w_1^N}{\operatorname{argmax}}\, p(w_1^N) p(x_1^T | w_1^N)\ . \tag{1.2}$$

In statistical speech recognition systems, statistical models are assumed for the two factors appearing in Eq. (1.2). The prior probability of a word sequence $p(w_1^N)$ is given by the language model (LM). The conditional probability of observing an acoustic vector

sequence for the given word sequence $p(x_1^T|w_1^N)$ is defined by the acoustic model (AM). The parameters of the models are learned from training data in the training phase of the system. Figure 1.1 depicts the general architecture of a statistical speech recognition system, which consists of four main components:

- The *feature extraction* module performs signal analysis to compute acoustic features $x_1^T$ from the input speech signal.
- The *language model* provides the prior probability of the hypothesized word sequence $w_1^N$ based on the syntax and semantics of the language.
- The *acoustic model* consists of statistical models for words or smaller sub-word units, and a *pronunciation lexicon* which defines the mapping from words to sub-word units, e.g. phonemes.
- The *search module* combines the knowledge sources to determine the best word sequence according to Eq. (1.2).

In the following sections, these components will be described in more detail.

## 1.2 Feature Extraction

The feature extraction module converts the digital speech signal into a sequence of acoustic vectors. This preprocessing stays apart from the statistical approach and is motivated by models of the human auditory system. Normalization techniques can be applied to remove information which is irrelevant for the speech recognition process. The complexity of the feature extraction depends on the structure of the acoustic model. Shallow models require a rather complex feature extraction. Hierarchical models learn a feature representation from data and require less preprocessing stages.

In most speech recognition systems, the feature extraction starts with a short-term spectral analysis [Rabiner & Schafer 79] based on a fast Fourier transform. The phase information of the Fourier transform is removed by applying the modulus and the dimension is reduced using triangular-shaped band filters. Further processing yields the widely used Mel-frequency cepstral coefficients (MFCC) [Davis & Mermelstein 80] or perceptual linear prediction (PLP) coefficients [Hermansky 90].

The short-term features lack dynamic information of the signal, which can be incorporated by concatenation of neighboring feature vectors. For generative models, a dimension reduction of the resulting feature vector is required. Typically, this is done using a linear discriminant analysis (LDA) [Häb-Umbach & Ney 92] which maximizes class separability in the lower dimensional feature space [Fisher 36]. For neural networks, less preprocessing steps are required and higher-dimensional features can be employed. In most systems, either MFCC features with a relatively large temporal context are used, or the filter bank outputs are used directly [Mohamed & Dahl$^+$ 12]. Using neural networks, it is even possible to work directly on the raw speech signal, see e.g. [Palaz & Collobert$^+$ 13, Tüske & Golik$^+$ 14].

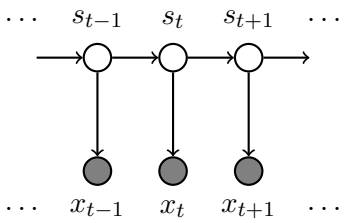


**Figure 1.2.** Illustration of an HMM as a graphical model. White vertices correspond to hidden states, gray vertices to observations. The arrows illustrate conditional dependence.

Speaker adaptation is an important component in state-of-the-art speech recognition systems and can be performed at different levels of the recognition system. At feature level, vocal tract length normalization is a popular technique to compensate for variability caused by different length of the vocal tract [Eide & Gish 96, Welling & Kanthak$^+$ 99]. In addition, constrained maximum likelihood linear regression (CMLLR) can be used for estimating speaker-dependent linear transformations of the features [Gales 98]. Invariance of the recognition system to speaker variations can also be improved by generating artificial training data using feature transformation methods. For example vocal tract length perturbation has been reported to improve speech recognition performance [Jaitly & Hinton 13].

## 1.3 Acoustic Model

The aim of acoustic modeling is to provide a statistical model for the conditional probability $p(x_1^T|w_1^N)$ of an acoustic vector sequence $x_1^T$ given a word sequence $w_1^N$. The de-facto standard in speech recognition is to assume a hidden Markov model (HMM) for the acoustic model [Baum & Eagon 67, Baker 75]. HMMs allow for modeling varying speaking rates and structuring the model into smaller sub-units.

An HMM is a statistical model which can be regarded as a stochastic finite state automaton. Each state has a probability distribution, which models the stochastic output of the state. The transitions between states are stochastic, i.e. the HMM has a probability distribution for the transition between states. In HMM terminology, the states of the automaton are called *hidden states* and the outputs are called *observations*. The hidden state sequence is assumed to be a first-order Markov process. The observations are conditionally independent of all other random variables given the state, see Fig. 1.2.

In speech recognition, the observations are the acoustic features and the state sequence corresponds to the positions in a word sequence. The HMMs have a left-to-right topology, which allows forward, skip, and loop transitions, and thus model varying speaking rates. A typical example of such a topology is shown in Fig. 1.3.

In the simplest case, words are modeled by HMMs with a pre-defined number of states and a left-to-right topology (*whole-word models*). Word sequence HMMs are

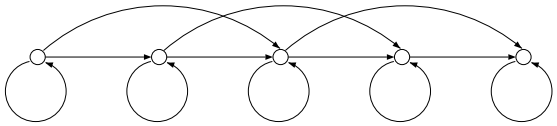

**Figure 1.3.** Depiction of the Bakis HMM topology.

obtained by concatenating word models. The conditional probability of an acoustic observation sequence $x_1^T$ given the word sequence is defined as the sum over all possible state sequences $s_1^T$ in this HMM:

$$p(x_1^T|w_1^N) = \sum_{s_1^T} p(x_1^T, s_1^T|w_1^N) \tag{1.3}$$

$$= \sum_{s_1^T} \prod_{t=1}^{T} p(x_t, s_t|x_1^{t-1}, s_1^{t-1}, w_1^N) \tag{1.4}$$

$$= \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|x_1^{t-1}, s_1^t, w_1^N)\ p(s_t|x_1^{t-1}, s_1^{t-1}, w_1^N)\ . \tag{1.5}$$

The first-order Markov assumption of the model implies that the probability of a state depends only on the predecessor state. Further, it is assumed that the observations are conditionally independent of all other random variables given the state. This implies:

$$p(x_1^T|w_1^N) = \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t, w_1^N)\ p(s_t|s_{t-1}, w_1^N)\ . \tag{1.6}$$

The quantities $p(x_t|s_t, w_1^N)$ and $p(s_t|s_{t-1}, w_1^N)$ are known as *emission probabilities* and *transition probabilities* respectively.

The parameters of the HMMs are tied. For example in the case of whole-word models, the parameters of the same word in different contexts are identical. As a result from the tying, the number of emission models is finite. Formally, the tying is a function that maps a word sequence and an HMM state to an *emission model label*

$$\mathcal{A}(w_1^N, s) = a \tag{1.7}$$

such that

$$p(x|s, w_1^N) = p(x|a)\ . \tag{1.8}$$

Tying is also used for the transition model. In the RWTH Aachen system, the probabilities of transitions of the same type (e.g. loop, forward, and skip) are tied.[1]

[1] See the system descriptions in Chapter A for details of the practical implementation.

The classical choice for the emission model is a Gaussian mixture model (GMM). In this case, the acoustic model is referred to as a Gaussian-mixture-HMM (GHMM). The GMM emission probabilities for a label $a$ and an observation $x$ are defined as

$$p(x|a) = \sum_{l=1}^{L_a} c_{a,l} \mathcal{N}(x|\mu_{a,l}, \Sigma_{a,l}) \ . \tag{1.9}$$

Here, $\mathcal{N}(x|\mu, \Sigma)$ denotes the normal distribution with mean $\mu$ and covariance matrix $\Sigma$. The non-negative mixture weights ($c_{a,l}$) are subject to the constraint

$$\sum_{l=1}^{L_a} c_{a,l} = 1 \tag{1.10}$$

for each label $a$. In the RWTH Aachen system, the covariances of all states and densities are tied and assumed to be diagonal. This choice ensures that the covariance matrix is non-singular. A more mundane advantage is that the calculation of likelihoods can be performed more efficiently [Kanthak & Schütz$^+$ 00].

In the training phase of the system, the parameters of the acoustic model are estimated from data. The GMM parameters can be estimated efficiently according to the maximum likelihood (ML) criterion using the expectation maximization (EM) algorithm [Dempster & Laird$^+$ 77]. The performance of a GHMM system can be improved by a subsequent *discriminative training* step. Starting from a ML initialization, an objective function based on the posterior probability of the word-sequence is optimized. The most widely-used objective functions are maximum mutual information (MMI) [Bahl & Brown$^+$ 86, Normandin 96, Valtchev & Odell$^+$ 97], and minimum phone error (MPE) [Povey & Woodland 02], which are discussed in detail in Chapter 3.

A number of techniques have been introduced for the extension of GHMM acoustic models to large vocabulary continuous speech recognition (LVCSR). Instead of whole-word HMMs, the word HMMs are built from small sub-word units. The *pronunciation lexicon* provides the mapping from words to sequences of sub-word units. Using sub-word instead of whole word models enables the speech recognition system to recognize words which have not been seen in the training data. Moreover, it has the advantage of a more reliable parameter estimation, because more training instances are available for these units.

The most widely-used sub-word units for LVCSR are *clustered triphones*. The pronunciation lexicon stores the mapping from words to sequences of *phonemes*, i.e. classes of speech sounds, which are perceived as equivalent to each other in a given language [Esling 10]. Each phoneme sequence uniquely defines a sequence of triphones, i.e. phonemes together with their predecessor and successor phonemes. Words are then modeled by concatenating several HMMs, each representing a triphone. The triphone HMMs typically have three or six states. A widely used variant is the Bakis topology [Bakis 76], which is depicted in Fig. 1.3. The motivation for using triphones is that the acoustic realization of a phoneme depends strongly on the surrounding phonemes. This effect,

which is known as *coarticulation*, also occurs across word boundaries. Therefore, it is important to consider the surrounding phonemes across word boundaries in triphone models [Hon & Lee 91, Odell & Valtchev$^+$ 94, Sixtus 03].

The number of all possible triphones is too large for estimating separate models for each of them reliably. Many triphones might even not be contained in the training data at all. Therefore, the parameters of acoustically similar HMM state models are tied [Young 92]. Commonly, a phonetic decision tree is used to determine a clustering of HMM states [Young & Odell$^+$ 94]. This top-down clustering approach has the advantage of assigning state models also to triphones unseen during training.

The pronunciations contained in the pronunciation lexicon are either generated manually or automatically. Typically, the pronunciations of a base vocabulary are generated manually. Automatic grapheme-to-phoneme conversion algorithms are used for generating missing pronunciations. In the RWTH Aachen system, a statistical approach for grapheme-to-phoneme conversion is employed [Bisani & Ney 03].

A word can have multiple pronunciations. For example the word "the" has a prevocalic and a preconsonantal pronunciation. The HMM for a word sequence is therefore in general non-linear and has different states for different pronunciations. The pronunciation probabilities are assumed to be included in the transition probabilities.

The GHMM approach for acoustic modeling has been dominant for about two decades. In recent years, discriminative approaches for acoustic modeling have received much attention. In particular, hybrid deep neural network (DNN)-HMMs have been shown to outperform GHMMs considerably. Neural networks and the hybrid approach are discussed in Section 1.9.

## 1.4 Language Modeling

The language model $p(w_1^N)$ provides a prior probability for a word sequence $w_1^N$. Ideally, the model implicitly covers syntax and semantics of the language. A major advantage of the factorization (1.2) into acoustic model and language model is that the language model can be trained on text data only, which can be obtained easily.

The classical choice for the language model is an $m$-gram count model [Bahl & Jelinek$^+$ 83], which makes the assumption that the probability of a word depends only on its $m-1$ predecessors (Markov assumption). According to the chain rule of probability, the probability of a word sequence $w_1^N$ can be factorized as

$$p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_1^{n-1}) \; . \tag{1.11}$$

Making use of the Markov assumption, Eq. (1.11) can be simplified to

$$p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}) \; . \tag{1.12}$$

Here, we use the convenient notation that $p(w_n|w_{n-m+1}^{n-1})$ is $p(w_n|w_1^{n-1})$ if $m$ is larger than $n$ and $p(w_1|w_1^0)$ is $p(w_1)$. Consecutive sequences of $m$ words are referred to as *m-grams*. In principle, the probabilities $p(w_n|w_{n-m+1}^{n-1})$ can be estimated as relative frequencies from $m$-gram counts. However, the number of possible $m$-grams grows exponentially in $m$. Even when using very large training sets, many $m$-grams have only very few observations or are not contained in the training data at all. Commonly, the $m$-gram probabilities are smoothed by discounting in combination with backing-off or interpolation [Katz 87, Generet & Ney$^+$ 95, Ney & Essen$^+$ 94, Kneser & Ney 95]. In discounting, probability mass is removed from the non-zero probabilities. A fall-back model of lower $m$-gram order is used for distributing the discounted probability mass over all unseen $m$-grams (backing-off) or over all $m$-grams (interpolation).

Recently, neural networks have also become popular for language modeling. Feed-forward neural networks (NNs) allow for estimating the $m$-gram probability directly without backing-off [Bengio & Ducharme$^+$ 01, Schwenk 07]. With recurrent neural networks (RNNs), even the Markov assumption is not required [Mikolov & Karafiát$^+$ 10, Sundermeyer & Schlüter$^+$ 12]. Despite of some recent progress in using neural network language models directly in search [Huang & Zweig$^+$ 14], they are mostly applied in a lattice or $n$-best list rescoring step.

## 1.5 Search

In the search process, the most likely word sequence according to Eq. (1.16) for the observed speech signal is computed. As shown in Fig. 1.1, the search module combines all knowledge sources: the acoustic model, the pronunciation model, and the language model. If the acoustic model is an HMM and the language model an $m$-gram model, the search module has to solve the maximization problem

$$\hat{w}_1^{\hat{N}} = \underset{w_1^N}{\operatorname{argmax}}\, p(w_1^N) p(x_1^T|w_1^N) \tag{1.13}$$

$$= \underset{w_1^N}{\operatorname{argmax}} \Big\{ \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}) \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t, w_1^N)\; p(s_t|s_{t-1}, w_1^N) \Big\} . \tag{1.14}$$

Usually, the summation over all state sequences is approximated by the maximum. This approximation is known as the maximum approximation or *Viterbi approximation* [Viterbi 67, Ney & Aubert 96]. Equation (1.14) then becomes

$$\hat{w}_1^{\hat{N}} = \underset{w_1^N}{\operatorname{argmax}} \Big\{ \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}) \max_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t, w_1^N) \quad p(s_t|s_{t-1}, w_1^N) \Big\} \quad . \tag{1.15}$$

In this formulation, the search process solves a single-source shortest path problem in a search network.

Both, Eq. (1.14) and Eq. (1.15) can be evaluated in linear time using dynamic programming [Bellman 57, Viterbi 67, Baum 72, Ney 84, Rabiner & Juang 86]. Nevertheless, the size of the search network is enormous. In LVCSR, it is clearly intractable to evaluate all hypotheses. Therefore, algorithms which discard unlikely hypotheses, i.e. which *prune* a large part of the search space, are required. As a consequence, *search errors* can occur, i.e. the best hypothesis may not be found because it is pruned. In principle, pruning allows to control computation time by trading speed for recognition quality. If the pruning parameters are adjusted properly, no significant search errors occur.

The approaches to the search problem can be broadly distinguished by two properties [Aubert 02]. First, the search network can be pre-compiled in advance and used as a static structure, or, it can be expanded dynamically during search. Second, one distinguishes between breadth-first and depth-first search algorithms.

A prominent example of the depth-first strategy is the A$^*$ algorithm [Hart & Nilsson$^+$ 68]. It organizes the search in a time-asynchronous manner depending on a heuristic estimate of the costs to complete the path. The A$^*$ algorithm has been applied to speech recognition in [Jelinek 69, Paul 91], but since its quality depends strongly on the heuristic, breadth-first search algorithms are commonly used.

Breadth-first search algorithms compute the likelihoods of all active hypotheses at each time frame and are commonly used in combination with beam-pruning. At each time step, beam-pruning compares the likelihoods and keeps only those hypotheses which have likelihoods sufficiently close to the current best one [Lowerre 76, Ney & Mergel$^+$ 87, Ortmanns & Ney 95]. The computational complexity of the beam-search can be reduced further with a prefix-tree representation of the pronunciation dictionary [Ney & Häb-Umbach$^+$ 92, Ortmanns & Eiden$^+$ 98]. Acoustic and language model look-aheads aim at pruning sub-optimal paths as early as possible [Ortmanns & Ney 00, Alleva & Huang$^+$ 96, Nolden & Ney$^+$ 11, Nolden & Schlüter$^+$ 11].

Weighted finite state transducer (WFST) provide a generic way to construct and optimize the search space [Allauzen & Mohri 03, Mohri & Riley 97]. A WFST is a weighted finite state automaton with input and output labels. In the WFST framework, generic algorithms are used to combine the knowledge sources and to minimize the search space. In particular, a static minimized WFST search network implicitly applies the lexical prefix tree and the language model look-ahead technique [Kanthak & Ney$^+$ 02]. WFST-based search networks can also be generated dynamically in order to reduce memory requirements. A comprehensive overview on WFST-based search algorithms is given in [Rybach 14].

## 1.6 Log-Linear Model Combination

For optimal performance, the impact of the individual statistical models on the decision rule can be weighted by introducing *scaling factors*. The scaling is performed in log-space such that the model in Eq. (1.2) is eventually a log-linear model. For instance,

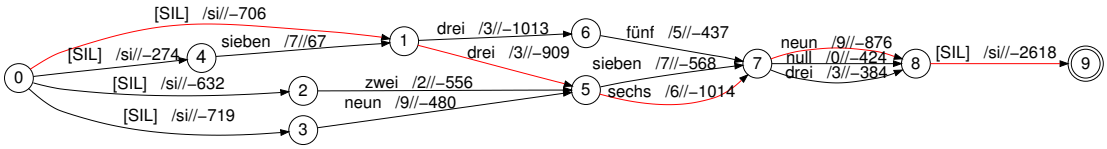


**Figure 1.4.** Example word lattice on a German digit strings recognition task (SieTill). The reference sequence is marked in red. [Heigold 10]

the decision rule with a language model scale $\gamma > 0$ becomes

$$\hat{w}_1^{\hat{N}} = \underset{w_1^N}{\operatorname{argmax}}\, p(w_1^N)^\gamma p(x_1^T | w_1^N) \; . \tag{1.16}$$

By convention, the emission model scale is fixed to 1.0 and only the scales for the language, transition, and pronunciation model are optimized. To keep the notational complexity at a minimum, we assume that all statistical models are already in their scaled form where appropriate.

## 1.7 Word Lattices

Many applications require a set of the most likely word sequences in addition to the first-best result. A common example is the application of models, which are too computationally expensive to be integrated during search, for example complex language models. The most direct form to represent a set of word sequences is an $N$-best list. A much more compact representation is achieved using *word lattices*.

In this work, a word lattice is defined as a weighted directed acyclic graph, where the arcs are labeled with pairs of a word and its pronunciation (word-pronunciations). The states are annotated with word boundary information including the time frame and the acoustic context in case of across word modeling. The arc weights are set to the language model, the acoustic model, or the combined log-probabilities. Usually, Viterbi approximation is applied on arc-level, i.e. the acoustic model score contains only the probability of the best HMM sequence. An example for a word lattice is shown in Fig. 1.4.

Lattices can be generated during a recognition pass, where the pruned search space is stored as a graph. The RWTH Aachen speech recognizer applies the word pair approximation [Ortmanns & Ney$^+$ 97], which simplifies their generation and makes them compact.

In this work, word lattices are important in the context of sequence-discriminative training, where they are used for representing the set of competing hypotheses [Valtchev & Odell$^+$ 97]. In sequence-discriminative training, it is usually assumed that the lattices also contain Viterbi arc-alignments. In our sequence-discriminative training experiments, the size of the lattices has been reduced by forward-backward pruning [Sixtus & Ortmanns 99], and the reference word sequence has been merged into the lattice.

## 1.8 Performance Measurement

The recognition quality of a speech recognizer is commonly measured in terms of its word error rate (WER) on transcribed test data. The word error rate is the normalized Levenshtein distance between the recognized word sequence and the correct word sequence. The normalization is performed w.r.t. the number of reference words. The Levenshtein distance is the minimal number of local edit operations (insertion, deletion, and substitution of words) required to transform one word sequence into the other [Levenshtein 66].

The quality of a language model can be measured in terms of its perplexity (PPL). The perplexity of a language model and a word sequence $w_1^N$ is defined as

$$\mathrm{PP}(w_1^N) = p(w_1^N)^{-\frac{1}{N}} = \left[\prod_{n=1}^{N} p(w_n|w_1^{n-1})\right]^{-\frac{1}{N}} . \tag{1.17}$$

The perplexity of a language model can be interpreted as the average number of choices to continue the word sequence $w_1^n$ at any position $n$.

## 1.9 Acoustic Modeling using Neural Networks

As outlined above, discriminatively trained GHMM speech recognition systems have been the state-of-the-art for decades. However, in recent years, *neural networks* have been shown to be superior to GMMs as acoustic models [Mohamed & Dahl$^+$ 09, Dahl & Ranzato$^+$ 10, Seide & Li$^+$ 11b, Sainath & Kingsbury$^+$ 11, Dahl & Yu$^+$ 12, Jaitly & Nguyen$^+$ 12]. Neural networks are discriminative models, i.e. they directly model the posterior probability required in classification. In contrast to discriminatively trained generative models, they are trained discriminatively from scratch. Another advantage of discriminative models is that their structure is not restricted to fit to a generative model. In particular, the structure of neural networks allows for the joint training of a discriminative feature extraction and the classifier.

Neural networks have already been used for speech recognition since the 1980's. At that time, neural networks were limited to isolated word recognition [Burr 86, Peeling & Moore$^+$ 86, Gold & Lippmann$^+$ 87] or had a very complex topology [Waibel & Hanazawa$^+$ 89]. These approaches could not compete with HMM based speech recognition systems. [Bourlard & Morgan 94] proposed the *hybrid approach*, which allows for using neural networks within the HMM framework. Initially, the success of the hybrid approach was rather limited, one reason being that the networks were undersized due to a lack of computing power. Neural networks were then mostly used as a feature extractor [Fontaine & Ris$^+$ 97, Hermansky & Ellis$^+$ 00, Grezl & Karafiát$^+$ 07].

In recent years, the interest in neural networks for speech recognition has again greatly increased. In [Mohamed & Dahl$^+$ 09] it was found that deep neural networks, i.e. neural networks with many hidden layers, outperform GMMs for phone recognition. [Seide &

Li$^+$ 11b] scaled DNNs to LVCSR. In earlier work on *deep learning* it was argued that deep neural networks require an unsupervised pre-training step [Hinton & Salakhutdinov 06]. Later it has been found that these techniques are not necessary for large-scale tasks like speech recognition [Seide & Li$^+$ 11a].

Since neural networks and the related log-linear models are the main topic of this thesis, they are described in more detail in the following.

### 1.9.1 Neural networks

A comprehensive overview of neural networks in general would go beyond of the scope of this work. This section introduces the basic concepts and the formalism for neural networks with a multi-layer perceptron structure, which can be found in text books such as [Bourlard & Morgan 94, Bishop 06].

In our context, neural networks are used as a model for class-posterior probabilities. For an observation space $\mathbf{R}^D$ and a finite number of $C$ classes, a neural network is a parameterized function

$$g_\theta : \mathbf{R}^D \to \mathbf{R}^C, \quad x \mapsto g_\theta(x) \ . \tag{1.18}$$

The parameters of the network $\theta$ are learned from data. Their exact form is specified below. Note that we only consider static inputs and the class of networks we consider is restricted to feed-forward neural networks.

The outputs of the network are used as estimates for the class-posterior probability, therefore, we write

$$\left(g_\theta(x)\right)_c = p_\theta(c|x) \ . \tag{1.19}$$

This suggests the use of the decision rule

$$r : \mathbf{R}^D \to \{1, \ldots, C\}, \quad x \mapsto \operatorname*{argmax}_c \ p_\theta(c|x) \ . \tag{1.20}$$

The basic idea of neural networks is a hierarchical processing of the data, i.e. $g_\theta$ is a composition of parameterized functions. Mostly, the network is assumed to be organized as a stack of *layers*. Such an $L$-layer network is a composition of $L$ functions

$$g_\theta = g^{(L)}_{W^{(L)},b^{(L)}} \circ g^{(L-1)}_{W^{(L-1)},b^{(L-1)}} \cdots \circ g^{(1)}_{W^{(1)},b^{(1)}} \ . \tag{1.21}$$

Each function is of the form

$$g^{(l)}_{W^{(l)},b^{(l)}} : \mathbf{R}^{D_{l-1}} \to \mathbf{R}^{D_l} \ , \quad l = 1, \ldots, L \tag{1.22}$$

with $D_l \in \mathbf{N}$, $D_0 = D$, $D_L = C$, and parameters $W^{(l)} \in \mathbf{R}^{D_{l-1} \times D_l}$ and $b^{(l)} \in \mathbf{R}^{D_l}$. These functions are referred to as layers with *weight matrix* $W^{(l)}$ and *bias* $b^{(l)}$. The function $g^{(L)}$ is the *output layer*, and the remaining $g^{(l)}$ are the *hidden layers* of the network. The parameters of the network are the weight matrices and biases of all layers:

$$\theta = \left((W^{(1)}; b^{(1)}), \ldots, (W^{(L)}; b^{(L)})\right) \ . \tag{1.23}$$

Each layer itself is a composition of the affine transformation defined by $(W^{(l)}, b^{(l)})$ and an *activation function* $\boldsymbol{\sigma}_l$:

$$g^{(l)}_{W^{(l)},b^{(l)}}(x) = \boldsymbol{\sigma}_l({W^{(l)}}^\top x + b^{(l)}) \; . \tag{1.24}$$

To enable non-linear classification, the activation function needs to be non-linear. Typically, it has the same functional form in all components:

$$\boldsymbol{\sigma}_l : \mathbf{R}^{D_l} \to \mathbf{R}^{D_l}, \quad y \mapsto (\sigma(y_1), \ldots, \sigma(y_{D_l})) \; , \tag{1.25}$$

with some $\sigma : \mathbf{R} \to \mathbf{R}$. The classical choice for the activation function is the *sigmoid* function:

$$\sigma : \mathbf{R} \to \mathbf{R}, \quad y \mapsto \frac{1}{1 + e^{-y}} \; . \tag{1.26}$$

In order to ensure that the output of the network is a proper probability distribution, the *softmax* non-linearity [Bridle 90] can be used at the output layer:

$$\boldsymbol{\sigma}_L : \mathbf{R}^C \to \mathbf{R}^C, \quad y \mapsto \frac{1}{Z(y)}(e^{y_1}, \ldots, e^{y_C}) \; . \tag{1.27}$$

Here,

$$Z(y) = \sum_{c=1}^{C} e^{y_c} \tag{1.28}$$

is the *normalization term*.

The intermediate values required for evaluating the network with a given input $x$ are called *activations*:

$$\begin{aligned} x^{(0)} &= x \\ x^{(1)} &= g^{(1)}_{W^{(1)},b^{(1)}}(x^{(0)}) \\ &\vdots \\ x^{(L)} &= g^{(L)}_{W^{(L)},b^{(L)}}(x^{(L-1)}) \end{aligned} \; . \tag{1.29}$$

A typical illustration of a neural network as a graph is shown in Fig. 1.5. Calculating the activations in successive order corresponds to a traversal of this graph in topological order and is known as a *forward pass*.

### 1.9.2 Neural network training

In the supervised training setting, a *training sample*, i.e. a sequence of labeled observations, is given:

$$(x_n, c_n)_{n=1,\ldots,N} \subset \mathbf{R}^D \times \{1, \ldots, C\} \; . \tag{1.30}$$

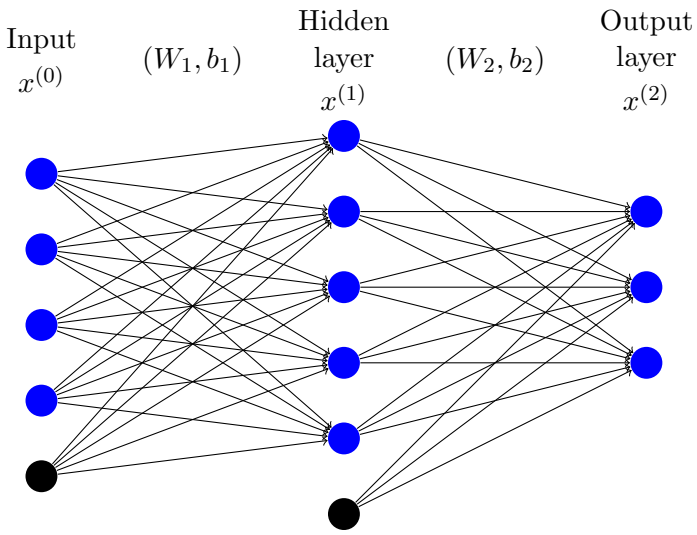


**Figure 1.5.** Illustration of a neural network with a four-dimensional input, a five-dimensional hidden layer, and a three-dimensional output layer. The biases are marked in black.

In training, the parameters of the network are optimized according to a training criterion, which depends on the training sample:

$$\mathcal{F}(\theta) = \sum_{n=1}^{N} \mathcal{F}_n(\theta) \ . \tag{1.31}$$

We use the convention that the objective function is minimized in training. Mostly, the cross-entropy (CE) criterion (with hard labels) is used [Bridle 90]:

$$\mathcal{F}_n(\theta) = -\log p_\theta(c_n|x_n) \ . \tag{1.32}$$

Note that the cross-entropy criterion requires that the output of the network is normalized, which is the case when a softmax output layer is used. The cross-entropy criterion is also known as maximum mutual information (MMI) and as conditional maximum likelihood criterion.

The minimum of the objective function can in general not be determined analytically. Therefore, iterative numerical optimization methods are employed, which make use of the gradient of the objective function. This gradient is computed using the *backpropagation algorithm*[2]. The algorithm performs a forward and a backward pass through the

[2]The history of the backpropagation algorithm is difficult to track down. Often [Rumelhart & Hinton$^+$ 86] is cited, although the same method has been used in many earlier works already in the 1960's, e.g [Bryson & Denham$^+$ 63]. The backpropagation algorithm for neural networks is also a special case of the reverse mode in automatic differentiation [Griewank 12], which has been developed in the 1970's. Confer [Schmidhuber 14] for a discussion.

network. In the forward pass, the activations of each layer are calculated in successive order. Then, the error signal at the output layer, i.e. the gradient of the objective function w.r.t. the output layer activations, is computed. In the backpropagation step, the error signals of the hidden layers are calculated recursively using chain-rule. The gradient can then be calculated from the activations and the error signals.

The most widely used optimization method for neural networks is stochastic gradient descent (SGD). SGD employs a stochastic approximation of the gradient, i.e. the gradient is only evaluated on a small random mini-batch $\mathcal{B} \subset \{1, \ldots, N\}$:

$$\nabla \mathcal{F}(\theta, \mathcal{B}) = \sum_{n \in \mathcal{B}} \nabla \mathcal{F}_n(\theta) \; . \tag{1.33}$$

In every iteration $i$, the parameters are updated by

$$\theta_i = \theta_{i-1} - \eta_i \nabla \mathcal{F}(\theta_{i-1}, \mathcal{B}_i) \; , \tag{1.34}$$

where $\eta_i > 0$ is the learning rate. The initialization $\theta_0$ can be set randomly or computed by a pre-training algorithm. Alternative algorithms for optimizing neural networks are the topic of Chapter 5 and 6.

In general, neural network training is a non-convex optimization problem. Using numerical optimization algorithms, only a local optimum of the objective can be determined. This local optimum depends on the initialization $\theta_0$ and the choice of the optimization algorithm and its hyperparameters such as the mini-batch size and the learning rates for SGD.

Deep networks, i.e. networks with many hidden layers, are sometimes initialized with a layer-wise restricted Boltzmann machine (RBM) pre-training [Hinton & Osindero$^+$ 06] or other unsupervised pre-training algorithms. However, it has been found that for large-scale tasks like speech recognition, unsupervised pre-training is not necessary. Already randomly initialized DNNs perform almost as well as RBM pre-trained models. The same or even larger improvements as with RBM pre-training are also obtained by a supervised layer-wise pre-training [Seide & Li$^+$ 11a].

### 1.9.3 Log-linear models

Log-linear (LL) models are a special case of neural networks with only a softmax output layer and no hidden layers. This specific structure of log-linear models has important consequences which distinguishes them from general neural networks.

The posterior probability of a class $c$ given an observation $x$ defined by a log-linear model is of the form:

$$p_{(\Lambda,\beta)}(c|x) = \frac{1}{Z(x)} \exp\left(\lambda_c^\top x + \beta_c\right) \quad \text{with} \quad Z(x) = \sum_{\bar{c}} \exp\left(\lambda_{\bar{c}}^\top x + \beta_{\bar{c}}\right) \; . \tag{1.35}$$

Here, $\Lambda = (\lambda_1; \ldots; \lambda_C) \in \mathbf{R}^{D \times C}$ and $\beta \in \mathbf{R}^C$ are the parameters of the log-linear model.

In contrast to neural networks, log-linear models imply only linear decision boundaries. Non-linear classification is achieved by mapping observations to a higher-dimensional feature space. A typical example is a log-linear model with quadratic features. Such a model has linear decision boundaries in the higher-dimensional feature space, but quadratic decision boundaries in the original feature space.

This approach is closely related to the concept of kernels, which are widely used in many machine learning algorithms, most prominently in support vector machines (SVMs) [Cortes & Vapnik 95]. While for log-linear models the features are explicitly computed, this is usually avoided in SVMs by means of the kernel trick [Aizerman & Braverman$^+$ 64, Boser & Guyon$^+$ 92].[3] For SVMs, this comes at the price of a training time complexity which is in general cubic in the number of training samples [Bottou & Lin 07].

The main advantage of log-linear models in comparison to general neural networks is that their training according to the cross-entropy criterion is a convex optimization problem. In principle, this allows for determining the global optimum of the objective function, independently of the initialization and the specific choice of the optimization algorithm. Mostly, log-linear models are optimized using general-purpose optimization algorithms such as L-BFGS [Liu & Nocedal 89].

Log-linear models are also motivated by the maximum entropy principle [Jaynes 57]. It can be shown that a probability distribution maximizing the entropy among all distributions satisfying *consistent constraints* must be of log-linear form [Darroch & Ratcliff 72]. Therefore, log-linear models are also known as maximum entropy models. Since the maximum entropy solution is identical to the conditional maximum likelihood solution of a log-linear model [Darroch & Ratcliff 72], the notion of maximum entropy can serve as a motivation but is not required in the following. We are primarily interested in the functional form of the model, therefore we use the term log-linear model.

### 1.9.4 The hybrid approach

The application of neural networks to LVCSR is not straightforward. The framework presented above is only suited for static input. The *hybrid approach* [Bourlard & Morgan 94] allows for using neural networks as emission models in the HMM framework.

In the hybrid approach, the neural network is trained with the emission model labels as classes and the acoustic observations as the input. In other words, the network is a model for the posterior probability $p_\theta(a|x)$ of a label $a$ given an acoustic observation $x$. In recognition, the class-conditional probability $p(x|a)$ of an emission model label $a$ is required. Using Bayes' rule, the class-conditional probability can be expressed in terms of the posterior probability:

$$p(x|a) = \frac{p(a|x)\, p(x)}{p(a)} \,. \tag{1.36}$$

[3] Note that it is possible to train log-linear models using the kernel trick as well [Kubo & Wiesler$^+$ 11]. Also, SVMs are sometimes trained without the kernel trick [Chapelle 07].

The prior probability $p(a)$ can be estimated easily as the relative frequency in an HMM state alignment. The marginal probability $p(x)$ can be discarded in recognition, because it does not change the maximizing argument in Eq. (1.15). The emission score used in hybrid NN-HMMs is obtained by plugging in the estimate of the neural network for the posterior probability:

$$g(x, s, w_1^N) = \frac{p_\theta(a|x)}{p(a)} \ , \tag{1.37}$$

where

$$\mathcal{A}(w_1^N, s) = a \ . \tag{1.38}$$

## 1.10 Related Work

In the next chapter, we will present the scientific goals of this work. In order to better understand the context, in which these goals have developed, we here give a brief overview on related publications for each of the sub-topics of this work.

**Log-linear modeling for speech recognition (Chapter 3)** Log-linear models, also known as maximum entropy models, are well known in the field of statistics [Jaynes 57, Darroch & Ratcliff 72] and are a general tool in machine learning. They have found a wide range of applications in the natural language processing domain [Rosenfeld 94, Berger & Pietra$^+$ 96, Ratnaparkhi et al. 96]. Of particular interest in the context of speech recognition are sequential models. The maximum entropy Markov model (MEMM) [McCallum & Freitag$^+$ 00] has been a first attempt in this direction, but due to its position-wise normalization it is limited by the *label bias problem* [Bottou 91]. Conditional random fields (CRFs) do not suffer from this problem [Lafferty & McCallum$^+$ 01] and have been applied successfully to many natural language processing tasks, e.g. [Sha & Pereira 03, Cohn 07]. Instead of using a discriminative sequence model like MEMMs or CRFs, log-linear models can also be used in combination with HMMs via the hybrid approach presented above [Bourlard & Morgan 94].

As linear discriminative models, maximum entropy models are also strongly related to SVMs and structured SVMs, their extension to structured, in particular sequential data [Tsochantaridis & Joachims$^+$ 05]. Using kernels for enabling non-linear classification is a standard technique in machine learning, especially for SVMs.

A number of researchers have worked on the application of log-linear models to speech recognition. This is also motivated by the observation that the posterior form of the commonly used Gaussian distribution is a log-linear model. Earlier works were carried out on small-scale tasks like digit recognition [Macherey & Ney 03, Heigold 10], where good results are already achieved with linear features.

Different approaches have evolved for scaling the models to larger tasks. Analogous to GMMs, latent variables can be introduced into the model, leading to log-linear mixture models and hidden CRFs (HCRFs) [Gunawardana & Mahajan$^+$ 05]. The drawback of this approach is that the convexity of the model is lost and an initialization with

a generative model is required. Another direction is based on exploiting a generative model within the discriminative framework. This can be done by using the sufficient statistics of the generative model as features [Jaakkola & Haussler 99, Layton & Gales 06] or directly using the output of the generative model on whole segments [Heigold & Zweig$^+$ 09, Zweig & Nguyen 09, Zhang & Ragni$^+$ 10, Ragni & Gales 11]. In such a framework, the log-linear model does not replace the generative model, but is trained on top of it and is mostly applied in a lattice rescoring step. The work most closely related to our approach is [Hifny & Renals$^+$ 05, Hifny & Renals 09], where a model with generic features is used directly in decoding.

In some works, neural network features have been employed in a CRF framework [Fosler-Lussier & Morris 08, Do & Artières 10]. Of course, the neural network is then the major part of the acoustic model. These papers already point into the direction of sequence-discriminatively trained hybrid DNN-HMMs, which we study in Chapter 7 and nowadays are considered as state-of-the-art.

**Analysis of convergence properties of log-linear training (Chapter 4)** The convexity of training could possibly lead to the misconception that optimization is of minor importance for log-linear models. This however, is not valid for large-scale tasks, which can be characterized by the fact that computation time is limited [Bottou & Bousquet 08].

Several researchers have empirically compared different optimization algorithms for log-linear training. While in earlier works on log-linear models the optimization problem has been solved with generalized iterative scaling (GIS) [Darroch & Ratcliff 72] or improved iterative scaling (IIS) [Lafferty & McCallum$^+$ 01], it has been found that gradient-based numerical optimizers are much faster, in particular the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) (L-BFGS) algorithm [Liu & Nocedal 89, Minka 01, Malouf 02, Sha & Pereira 03].

The theoretical analysis of the optimization problem is limited. [Salakhutdinov & Roweis$^+$ 03] derived a convergence analysis specifically for bound optimization algorithms like GIS and showed that GIS converges slowly when features are correlated and have a non-zero mean. In the context of neural networks, the analysis [LeCun & Kanter$^+$ 90] is often referenced. In fact, [LeCun & Kanter$^+$ 90] analyze the convergence properties of a linear regression model with mean squared error loss. Although they study a different combination of model and loss function, their analysis has similarities with ours. A comparison with their work is given in Chapter 4.

**Second-order optimization of neural networks (Chapter 5 and Chapter 6)** Learning deep neural networks is a difficult optimization problem [Glorot & Bengio 10]. This makes second-order optimization a promising research direction. Here, only a brief overview on the large number of papers on second-order optimization for deep learning is given.

Optimization algorithms for deep learning can be categorized into stochastic and batch algorithms. Stochastic algorithms are typically faster, while batch algorithms are

more accurate and can be parallelized more easily. For batch algorithms, the optimization can be regarded as a black box and numerical optimizers known from optimization literature can be employed, e.g. L-BFGS [Liu & Nocedal 89, Dean & Corrado$^+$ 12]. Recently, Hessian-free (HF), an algorithm specifically tailored for DNN training, has been proposed [Martens 10]. [Kingsbury & Sainath$^+$ 12] applied HF to sequence-discriminative training of acoustic models and reported improvements in recognition accuracy and wall-clock time.

Stochastic second-order algorithms usually make strong approximations of the Hessian matrix. For example, [LeCun & Bottou$^+$ 98] use a diagonal approximation. AdaGrad [Duchi & Hazan$^+$ 11] is another stochastic diagonal second-order algorithm, which has recently been applied to large-scale acoustic model training [Dean & Corrado$^+$ 12]. In the context of this thesis, the publication [Raiko & Valpola$^+$ 12] is of interest, which aims at improving convergence behavior by using an adaptive activation function.

**Sequence-discriminative training of neural networks (Chapter 7)** Typically, neural networks are trained according to the cross-entropy criterion, which allows for training models discriminatively from scratch. This criterion optimizes the decision on frame level although the decision on sequence level is required in recognition. Similar to Gaussian mixtures and log-linear models, the performance of neural networks can be improved by fine-tuning the models according to sequence-discriminative criteria.

Sequence-discriminative training of neural networks has first been proposed by [Kingsbury 09]. With the advent of deep learning, this work received more attention and has been extended from shallow networks to deep networks [Kingsbury & Sainath$^+$ 12]. Several heuristics to make sequence-discriminative training more robust have been proposed in [Su & Li$^+$ 13, Veselý & Ghoshal$^+$ 13]. According to [Heigold & McDermott$^+$ 14], creating word lattices on-the-fly improves the stability of training and makes such heuristics unnecessary.

## 1.11 Document Structure

The main chapters of this thesis are organized as follows. In the next chapter, the scientific goals of this thesis are presented. Chapter 3 presents our work on log-linear models for speech recognition. An analysis of the convergence behavior of log-linear training is given in Chapter 4. Chapter 5 and Chapter 6 deal with second-order optimization of neural networks. In Chapter 7, we investigate sequence-discriminative training of neural networks. The scientific contributions of this work are summarized in Chapter 8. Finally, an outlook on possible future directions related to this work is given in Chapter 9.

# Chapter 2

# Scientific Goals

Discriminative techniques are a major line of research in automatic speech recognition. At the beginning of this work, Gaussian mixture models were the de-facto standard for acoustic modeling. These *generative models* are trained according to the maximum likelihood criterion using the expectation maximization algorithm. Their results can be improved with discriminative training using the maximum likelihood model as initialization.

In this thesis, we study the use of *discriminative models*, i.e. models which directly parametrize the posterior probability required in recognition. Such models are trained discriminatively from scratch and do not rely on a suboptimal maximum likelihood initialization. In the first part of this thesis, log-linear models are investigated, which have a shallow structure. Their main advantage is the convexity of training. During the last years, it became apparent that deep neural networks are superior to both Gaussian mixtures and log-linear models for acoustic modeling due to their hierarchical structure. Therefore, the scope of this thesis has been extended in the direction of deep neural networks.

In the following, we summarize the main scientific goals of this thesis.

**Investigations on convex training of large-scale acoustic models (Chapter 3)**
The discriminative training of Gaussian mixtures is a non-convex optimization problem, therefore it can get stuck in local optima. Conventional discriminative training involves many approximations and heuristics. One goal of this work is the development of a discriminative training procedure which requires less engineering work than conventional discriminative training does.

Log-linear models are a principled method to overcome the problems of conventional discriminative training. The training of log-linear models is a convex optimization problem, thus it can not get stuck in local optima. In principle, the unique global optimum can be found from any initialization using any optimization algorithm with guaranteed convergence.

In this thesis, we investigate different design choices required for log-linear modeling, namely the model structure, the feature functions, the training criterion, and the optimization algorithm. We aim for applying log-linear models to large-scale speech recognition tasks, therefore special attention is paid to training efficiency and how the design choices interact with it. A topic of particular interest is the choice of the opti-

mization algorithm.

**Analysis of convergence properties of log-linear training (Chapter 4)** Despite of the convexity of log-linear training, optimization plays an important role in its application. In practice, computation time is limited. The global optimum is approximated using numerical optimizers. On large-scale tasks, the quality of this approximation is an important factor contributing to the accuracy of the model.

The difficulty of a convex optimization problem can be described formally in terms of the condition number of the Hessian matrix. In this work, we analyze the condition number of log-linear training theoretically and draw practical conclusions. We validate the convergence analysis empirically on handwriting and speech recognition tasks.

Such a convergence analysis is of interest beyond log-linear training itself. Log-linear models are a special case of neural networks with a softmax output layer.

**Development and analysis of optimization algorithms for neural networks (Chapter 5 and Chapter 6)** Neural networks are almost always optimized with stochastic gradient descent. Since optimization plays an important role already for convex models, it is natural to investigate more sophisticated algorithms for neural network training as well. Second-order algorithms use a search direction based on a quadratic approximation of the objective function. We investigate the feasibility of the Hessian-free algorithm [Martens 10] for large-scale cross-entropy training. This batch algorithm uses a full quadratic model of the objective function and is therefore very accurate but also expensive.

Incorporating second-order information into stochastic algorithms is more difficult. Many proposed stochastic second-order algorithms in the literature either cause a large computational overhead in comparison to stochastic gradient descent or are difficult to tune. We develop a simple stochastic second-order algorithm which does not suffer from these problems. The algorithm is motivated by our convergence analysis of log-linear models.

With more accurate optimization of neural networks, overfitting gets more severe. For this reason, we investigate the use of a linear bottleneck topology for deep neural networks as a regularization method. The linear bottleneck topology is also attractive for accelerating training and evaluation of the network.

**Investigation of sequence-discriminative training of neural networks (Chapter 7)** Acoustic models in hybrid neural-network-HMM speech recognition systems have typically been trained on frame-level. In a number of recent works, it has been shown that neural-network-HMMs can be improved substantially by training them according to the sequence-discriminative criteria. However, it is not possible to draw clear conclusions on the optimal training setup from current literature. Several authors observed problems with its stability and proposed a variety of solutions. In this work, we aim for getting more empirical insight into this technique. Further, our work on training linear bottleneck networks from scratch is extended.

# Chapter 3

# Convex Log-Linear Acoustic Models for Large-Scale Speech Recognition

Generative acoustic models based on Gaussian mixtures can be significantly improved by discriminative training. Conventional discriminative training is a non-convex optimization problem, therefore it can get stuck in local optima. In addition, it involves many heuristics and approximations, for example the initialization and smoothing with a suboptimal maximum likelihood model, and the use of word lattices. This leads to much engineering work in practice.

In this chapter, we explore log-linear models for acoustic modeling as a principled method to overcome the problems of discriminative training. Log-linear models are discriminative models. They directly model the posterior probability required in classification. This avoids the indirection of discriminative training of generative models with maximum likelihood initialization. The training of log-linear models is convex. Thus, the global optimum of the objective function is accessible. In principle, the global optimum can be found from any initialization using any optimization algorithm with guaranteed convergence.

We investigate different design choices required for log-linear modeling, namely the model structure, the training criterion, the feature functions, and the optimization algorithm. Our goal is to apply convex log-linear models to large-scale speech recognition tasks, therefore special attention is paid to training efficiency.

In our experiments on medium to large-scale speech recognition tasks, we show that log-linear models can be trained discriminatively from scratch. The performance of log-linear models is comparable to discriminatively trained Gaussian mixture HMMs, but with the advantage of a convex training.

The framework developed in this chapter forms also the basis for the remaining chapters of this thesis.

## 3.1 Preliminaries

Before we proceed with the definition of log-linear models, we clarify the notation used throughout this chapter. Our framework is the statistical approach to pattern recognition, where the task is to classify observations. If not specified otherwise, $x$ and $x'$ are elements from the observation space $\mathbf{R}^D$, and $c, c' \in \{1, \ldots, C\}$ are classes. In

general sequence classification tasks, the targets are sequences of labels from a label set $\{1, \ldots, Y\}$. Individual labels are denoted by $y, \psi$. The classification rule is based on a statistical model $p_\Lambda$ with parameters $\Lambda$, which are optimized on the training sample.

The classes in speech recognition are word sequences. Words are denoted by $w$ or $v$. The training data consists of transcribed utterances $(\mathbf{x}_r, \mathbf{w}_r)_{r=1,\ldots,R}$ . Each utterance consists of a feature sequence $\mathbf{x}_r = (x_{r,1}, \ldots, x_{r,T_r})$ and its transcription $\mathbf{w}_r = (w_{r,1}, \ldots, w_{r,N_r})$. The total number of time frames is $T = \sum_{r=1}^{R} T_r$. Often, it is assumed that a Viterbi alignment of the training data is available, which is given in form of a sequence of HMM states $\mathbf{s}_r = (s_{r,1}, \ldots, s_{r,T_r})$ for each utterance $r = 1, \ldots, R$. The corresponding sequence of emission model labels is denoted by $\mathbf{a}_r = (a_{r,1}, \ldots, a_{r,T_r})$.

Some speech recognition training criteria are based on word lattices, which represent the most likely word sequences of an utterance. We assume that the lattices include Viterbi alignments of all lattice arcs. The word lattice corresponding to utterance $r$ is denoted by $\mathcal{L}_r$. We write $\mathcal{L}(\mathbf{w})$ for the sub-lattice of $\mathcal{L}$ consistent with a word sequence $\mathbf{w}$. The lattice arcs are word-pronunciations, i.e. pairs of a word and its pronunciation. Word-pronunciation sequences are denoted by $\boldsymbol{\pi}$. The function $\omega(\boldsymbol{\pi})$ maps from a word-pronunciation sequence to the corresponding word sequence.

In this chapter, we deal with models with a convex training criterion. A function $f : \mathbf{R}^n \to \mathbf{R}$ is called *convex* if for all $x, y \in \mathbf{R}^n$, and $\alpha \in [0, 1]$, we have

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y) \; . \tag{3.1}$$

$f$ is *strictly convex* if the inequality holds strictly whenever $x \neq y$. If $f$ is twice differentiable, $f$ is convex if and only if the Hessian matrix $\nabla^2 f(x)$ is positive semi-definite for all $x \in \mathbf{R}^n$. If $\nabla^2 f(x)$ is positive definite for all $x \in \mathbf{R}^n$, then $f$ is strictly convex. Note that the converse direction of this statement does not hold [Boyd & Vandenberghe 09, p71]. The function $f$ is *strongly convex* if and only if its Hessian is uniformly lower-bounded by a multiple of the identity.

Convex functions are important in optimization. Any local optimum of a convex function is a global optimum [Boyd & Vandenberghe 09, p69] and the set of global optima is convex [Boyd & Vandenberghe 09, p75]. The global optimum of strictly convex functions is unique. In contrast to non-convex functions, the global optimum of convex functions can be found efficiently using numerical optimization algorithms.

## 3.2 Modeling

In this section, the definitions of log-linear models on *frame-level* and on *sequence-level* are derived.

### 3.2.1 Frame-level model

Recall from Section 1.9.3 that a log-linear model defines a posterior probability for a class given an observation. The decision boundaries of log-linear models are linear. Non-linear classification is achieved by mapping observations to a higher-dimensional feature

space. Commonly, the mapping to the feature space is made explicit by denoting it as

$$f : \mathbf{R}^D \to \mathbf{R}^J \ . \tag{3.2}$$

The components of $f$ are called *feature functions* or *kernel functions*. For ease of notation, we can assume that one component of the feature mapping is constant, i.e. $f_J(x) = 1$ for all $x \in \mathbf{R}^D$. Then, separate bias parameters are not required and the log-linear model simplifies to

$$p_\Lambda(c|x) = \frac{1}{Z(x)} \exp\left(\lambda_c^\top f(x)\right) \tag{3.3}$$

with normalization constant

$$Z(x) = \sum_{\bar{c}} \exp\left(\lambda_{\bar{c}}^\top f(x)\right) \ . \tag{3.4}$$

Here, $\Lambda = (\lambda_1; \ldots; \lambda_C) \in \mathbf{R}^{J \times C}$ are the parameters of the model. Note that the parameters of log-linear models are redundant. Shifting the parameters of all classes by a vector $\bar{\lambda} \in \mathbf{R}^J$ results in exactly the same posterior probabilities:

$$p_{(\lambda_1;\ldots;\lambda_C)}(c|x) = p_{(\lambda_1+\bar{\lambda};\ldots;\lambda_C+\bar{\lambda})}(c|x) \ . \tag{3.5}$$

This parameter transformation is known as an *invariance transformation* [Heigold 10, Chapter 4].

An alternative formulation of log-linear models uses class-dependent features and a class-independent parameter vector $\Lambda \in \mathbf{R}^{JC}$:

$$p_\Lambda(c|x) = \frac{1}{Z(x)} \exp\left(\Lambda^\top f(x,c)\right) \quad \text{with} \quad Z(x) = \sum_{\bar{c}} \exp\left(\Lambda^\top f(x,\bar{c})\right) \ . \tag{3.6}$$

It can easily be seen that both formulations are equivalent in the sense that for a model in one notation, there is a model with the same posterior probabilities in the other notation, see [Heigold 10, p48].

A log-linear model induces a decision rule via

$$r : \mathbf{R}^D \to \{1, \ldots, C\}, \quad x \mapsto \operatorname*{argmax}_c p_\Lambda(c|x) = \operatorname*{argmax}_c \lambda_c^\top x \ . \tag{3.7}$$

Note that the normalization term does not appear in the decision rule. The normalization is required in training. Furthermore, it simplifies the integration of log-linear models into more complex classifiers because it enables their statistical interpretation.

As discussed in Section 1.9.4, static log-linear models can be integrated into HMM speech recognizers via the hybrid approach. The log-linear model is trained with the HMM emission model labels as classes, and the acoustic observations as input. Let $w_1^N$ denote a word sequence, $x_1^T$ a feature sequence, $s_1^T$ an HMM state sequence, and

$$a_t = \mathcal{A}(w_1^N, s_t) \ , \quad t = 1, \ldots, T \tag{3.8}$$

the corresponding emission model labels. Assuming a bigram language model for simplicity, recall that the discriminant maximized in Viterbi search is of the form

$$p(w_1^N, s_1^T, x_1^T) = \prod_{n=1}^{N} p(w_n|w_{n-1}) \prod_{t=1}^{T} p(x_t|a_t)\; p(s_t|s_{t-1}, w_1^N)\;. \tag{3.9}$$

Inserting the log-linear model (3.3) via the hybrid approach yields

$$p(w_1^N, s_1^T, x_1^T) = C' \prod_{n=1}^{N} p(w_n|w_{n-1}) \prod_{t=1}^{T} \frac{p_\Lambda(a_t|x_t)}{p(a_t)}\; p(s_t|s_{t-1}, w_1^N) \tag{3.10}$$

$$= C'' \exp\Big( \sum_{n=1}^{N} \log p(w_n|w_{n-1}) + \sum_{t=1}^{T} \lambda_{a_t}^\top f(x_t) - \log p(a_t) + \log p(s_t|s_{t-1}, w_1^N) \Big)\;. \tag{3.11}$$

$C'$ and $C''$ are positive constants independent of the state and the word sequence. The resulting model is termed *log-linear HMM*. Note that dividing by the state prior probability is equivalent to subtracting the log-prior from the bias parameter.

### 3.2.2 Sequence-level model

Instead of using a log-linear model within the hybrid approach, one can define the log-linear model directly on sequence level. Such models are known as conditional random fields (CRFs). In order to ensure efficient decoding, usually a linear chain structure is assumed.

Following [Sutton & McCallum 12, Definition 2.2], a *linear-chain CRF* is a probability distribution over label sequences given an observation sequence of the form:

$$p_\Lambda(y_1^T|x_1^T) = \frac{1}{Z(x_1^T)} \exp\left( \sum_{t=1}^{T} \Lambda^\top f(y_t, y_{t-1}, x_t) \right)\;. \tag{3.12}$$

Again,

$$Z(x_1^T) = \sum_{\psi_1^T} \exp\left( \sum_{t=1}^{T} \Lambda^\top f(\psi_t, \psi_{t-1}, x_t) \right) \tag{3.13}$$

is the normalization term. Note that the observation and the label sequence are required to have the same length. The class-dependent features are now functions of two consecutive labels and the observation. Usually, HMM-like features are employed, which are of the form:

$$f_{\psi,\psi'}(y, y', x) = \delta_{\psi,y}\, \delta_{\psi',y'}\;, \tag{3.14}$$

$$f_{\psi,j}(y, y', x) = \delta_{\psi,y}\, \phi_j(x)\;. \tag{3.15}$$

Here, $\phi : \mathbf{R}^D \to \mathbf{R}^J$ is a class-independent kernel function. For notational convenience, the features and corresponding parameters are indexed with label pairs respectively label dimension-index pairs. The sequence-level model then becomes

$$p_\Lambda(y_1^T|x_1^T) = \frac{1}{Z(x_1^T)} \exp\left(\sum_{t=1}^{T} \lambda_{y_t,\cdot}^\top \phi(x_t) + \lambda_{y_t,y_{t-1}}\right) . \tag{3.16}$$

The parameter corresponding to feature $f_{\psi,\psi'}$ then takes the role of the log-probability of the transition from $\psi'$ to $\psi$. The parameter vectors $\lambda_{y,\cdot} = (\lambda_{y,1}, \ldots, \lambda_{y,J})$ correspond to the emission model.

In speech recognition, the posterior probability of a word and a state sequence given an observation sequence is required. This can be obtained by adding features depending on the word sequence to the model and letting the state sequence take the role of the label sequence. The resulting model with HMM parameter tying is of the form

$$p(w_1^N, s_1^T|x_1^T) = \frac{1}{Z(x_1^T)} \exp\left(\sum_{n=1}^{N} \mu_{w_n,w_{n-1}} + \sum_{t=1}^{T} \lambda_{a_t}^\top \phi(x_t) + \lambda_{s_t,s_{t-1},w_1^N}\right) \tag{3.17}$$

with normalization constant analogous to the exponent in Eq. (3.17). The posterior probability $p(w_1^N|x_1^T)$ can be obtained by marginalizing over the state sequences. This model is known as a hidden CRF (HCRF), i.e. a CRF with hidden variables [Gunawardana & Mahajan$^+$ 05]. In addition, HCRF allow hidden variables on emission level. Thus, HCRFs are the posterior form of Gaussian mixture HMMs.

Note that the functional form of Eq. (3.11) and Eq. (3.17) is very similar. It can easily be seen that any log-linear HMM can be converted into a CRF with appropriate choice of features and parameters. One could suspect that the converse direction does not hold. In contrast to the CRF parameters, the language model and transition model parameters of the log-linear HMM are constrained to define a proper probability distribution. But with the technique presented in [Heigold & Ney$^+$ 11], it can be shown that any linear-chain CRF can be converted into a log-linear HMM. This implies that the log-linear HMM and the sequence-level model are equally expressive.

In the next section, we will see that it is more appropriate to distinguish between frame-discriminative and sequence-discriminative training.

## 3.3 Training Criteria

Log-linear models can be trained according to all criteria involving frame- or utterance-posterior probabilities. This includes the criteria known from discriminative training of GHMMs [Heigold & Schlüter$^+$ 12]. In principle, the sequence-level model allows for training the emission, transition, and language model jointly. However in this thesis, we only deal with the emission model. Therefore, we use the notation from Eq. (3.3) for log-linear models.

### 3.3.1 Cross-entropy

Cross-entropy[1] is a frame-discriminative training criterion. As such, it is defined on individual acoustic observations and their corresponding emission model label given by an alignment:

$$\mathcal{F}^{(\mathrm{CE})} : \mathbf{R}^{J \times A} \to \mathbf{R}, \quad \Lambda \mapsto -\frac{1}{T} \sum_{r=1}^{R} \sum_{t=1}^{T_r} \log p_\Lambda(a_{r,t}|x_{r,t}) + \alpha_{\mathrm{reg}}\, \mathcal{R}(\Lambda) \ . \tag{3.18}$$

Here, $\mathcal{R} : \mathbf{R}^{J \times A} \to \mathbf{R}$ is a *regularization term* with *regularization constant* $\alpha_{\mathrm{reg}} > 0$. The purpose of the regularization term is to improve the generalization ability of the model. Appropriate choices of $\mathcal{R}$ are discussed in Section 3.3.4. The first summand of the objective function is the *loss term*.

Minimizing the cross-entropy objective is a convex optimization problem. This directly follows from the fact that linear functions are convex, the function $f(x) = \log \sum_i \exp(x_i)$ is convex, and the composition of convex functions is again convex [Boyd & Vandenberghe 09, Chapter 3]. The convexity implies that the global optimum can be found from any initialization using numerical optimization algorithms. This is beneficial for training the model $p_\Lambda$ from scratch. Heuristics such as using a maximum likelihood initialization for discriminative training of Gaussian mixture models or pre-training of deep neural networks are not required.

For further reference, we state the first and second partial derivatives of the unregularized cross-entropy objective function for $1 \leq a, \bar{a} \leq A$ and $1 \leq j, \bar{\jmath} \leq J$:

$$\frac{\partial \mathcal{F}^{(\mathrm{CE})}}{\partial \lambda_{a,j}}(\Lambda) = \frac{1}{T} \sum_{r=1}^{R} \sum_{t=1}^{T_r} \left( p_\Lambda(a|x_{r,t}) - \delta_{a,a_{r,t}} \right) f_j(x_{r,t}) \ , \tag{3.19}$$

$$\frac{\partial^2 \mathcal{F}^{(\mathrm{CE})}}{\partial \lambda_{a,j} \partial \lambda_{\bar{a},\bar{\jmath}}}(\Lambda) = \frac{1}{T} \sum_{r=1}^{R} \sum_{t=1}^{T_r} p_\Lambda(a|x_{r,t}) (\delta_{a,\bar{a}} - p_\Lambda(\bar{a}|x_{r,t}))\ f_j(x_{r,t}) f_{\bar{\jmath}}(x_{r,t}) \ . \tag{3.20}$$

In contrast to the maximum likelihood approach, the cross-entropy criterion takes the competing classes into account. Consider the log-posterior of a training example $(a, x)$:

$$\log p_\Lambda(a|x) = \lambda_a^\top x - \log \sum_{a'} \exp\left( \lambda_{a'}^\top x \right) \ . \tag{3.21}$$

It can be improved either by increasing the score of the correct state $\lambda_a^\top x$ or decreasing the scores of the other states. Thus, the correct state competes against the other

[1] Note that the cross-entropy criterion is also known as *frame-discriminative MMI criterion*. The criterion is also identical to the (frame-discriminative) conditional maximum likelihood criterion. Further, the term MMI is often used for the corresponding sequence-discriminative criterion. To avoid ambiguity and for consistency with recent neural network literature, we use the term cross-entropy for the frame-discriminative criterion and MMI for the sequence-discriminative criterion.

states. In this sense, the objective function (3.18) is discriminative. Note that the discriminative approach requires evaluating the scores of all classes, which makes it computationally demanding.

Frame-discriminative training optimizes only the emission model of isolated frames. The other knowledge sources of the speech recognition system – the transition model, the pronunciation lexicon, and the language model – are not taken into account. From this point of view, frame-discriminative training is suboptimal. On the other hand, other heuristics are mostly avoided due to its simplicity. In particular, the criterion does not require word lattices.

### 3.3.2 Sequence-discriminative maximum mutual information

Sequence-discriminative maximum mutual information training of log-linear models directly optimizes the posterior probability of the whole training utterance, thereby taking into account all knowledge sources of the speech recognition system:

$$\mathcal{F}^{(\mathrm{MMI})} : \mathbf{R}^{J\times A} \to \mathbf{R}, \quad \Lambda \mapsto -\frac{1}{R}\sum_{r=1}^{R} \log p_\Lambda(\mathbf{w}_r|\mathbf{x}_r) + \alpha_{\mathrm{reg}}\,\mathcal{R}(\Lambda)\ . \tag{3.22}$$

Again, $\mathcal{R}$ denotes a regularization term. The utterance-posterior probability requires the normalization over all competing word sequences. For LVCSR, it is not feasible to carry out this summation over all possible word sequences and one restricts it to the most likely ones represented by a word lattice [Valtchev & Odell$^+$ 97]. The posterior probability of utterance $(\mathbf{x}, \mathbf{w})$ with lattice $\mathcal{L}$ is then defined as

$$p_\Lambda(\mathbf{w}|\mathbf{x}) = \sum_{\boldsymbol{\pi}\in\mathcal{L}(\mathbf{w})} p(\boldsymbol{\pi}|\mathbf{x}) \tag{3.23}$$

$$= \frac{1}{Z(\mathbf{x})} \sum_{\boldsymbol{\pi}\in\mathcal{L}(\mathbf{w})} p(\mathbf{w})^\alpha q_\Lambda(\mathbf{x}|\boldsymbol{\pi})^\alpha\ . \tag{3.24}$$

Here, the normalization term $Z(\mathbf{x})$ is obtained by summing over all paths in the lattice:

$$Z(\mathbf{x}) = \sum_{\boldsymbol{\pi}\in\mathcal{L}} p\left(\omega(\boldsymbol{\pi})\right)^\alpha q_\Lambda(\mathbf{x}|\boldsymbol{\pi})^\alpha\ . \tag{3.25}$$

The *acoustic pseudo-probability* $q_\Lambda$ is commonly defined using Viterbi approximation on arc-level:

$$q_\Lambda(\mathbf{x}|\boldsymbol{\pi}) = \exp\left(\sum_{t=1}^{T} \lambda_{a_t}^\top f(x_t) + \log p(s_t|s_{t-1}, \omega(\boldsymbol{\pi}))\right)\ , \tag{3.26}$$

where, $s_1^T$ is the state sequence composed of the arc-alignments of $\boldsymbol{\pi}$, and $a_1^T$ is the corresponding emission model label sequence.

The *smoothing weight* $\alpha > 0$ is typically set to the inverse of the language model scale used in recognition. Smoothing does not have an effect in the computation of the single-best Viterbi path, but in the computation of the utterance posterior, it prevents the normalization term to be dominated by a single utterance [Woodland & Povey 02]. Using a *weak* language model, typically a unigram, is another commonly used heuristic to ensure that the set of competing hypothesis has enough variation [Schlüter & Müller$^+$ 99]. An initial model is used for creating the lattices and computing the arc-alignments. Usually, the lattices and the alignments are kept fix during training.

Note that in general the reference is represented by multiple paths in the lattice because of pronunciation variants and optional silence arcs between word-arcs. Therefore, sequence-discriminative MMI involves a sum in the numerator of the utterance-posterior, cf. Eq. (3.23). Because of this summation, sequence-discriminative MMI is not convex. The criterion can be transformed into a convex function by restricting the summation to the best path according to the initial model [Heigold & Rybach$^+$ 09].

Sequence-discriminative training is not well-suited for training a model from scratch, because it requires an initial model for creating the word lattices. Instead, one can first train a log-linear model using a frame-discriminative criterion and then further optimize it according to a sequence-discriminative criterion. The goal of this chapter is the development of an alternative to discriminative training of GMMs, which is easier to use. Therefore, the experiments in this chapter are restricted to frame-discriminative training. We will revisit sequence-discriminative training in the context of neural networks in Chapter 7.

### 3.3.3 Minimum Bayes risk

Minimum Bayes risk (MBR) training goes back to [Kaiser & Horvat$^+$ 00, Kaiser & Horvat$^+$ 02]. The MBR criterion optimizes the expected error on the training data, where the expectation is taken with respect to the model. Typically the error is defined using the Levenshtein distance or an approximation to it.

Let $\mathcal{C}$ denote a cost function, which assigns a cost to a word-pronunciation sequence depending on the reference. The MBR objective function with respect to this cost is defined as

$$\mathcal{F}^{(\mathrm{MBR})} : \mathbf{R}^{J \times A} \to \mathbf{R}, \quad \Lambda \mapsto \frac{1}{R} \sum_{r=1}^{R} \sum_{\boldsymbol{\pi} \in \mathcal{L}} p_\Lambda(\boldsymbol{\pi}|\mathbf{x}_r) \mathcal{C}_{\mathbf{w}_r}[\boldsymbol{\pi}_r] + \alpha_{\mathrm{reg}}\, \mathcal{R}(\Lambda) \; . \tag{3.27}$$

As for sequence-discriminative MMI, it is implicitly assumed that the posteriors are based on scaled scores and a weak language model.

If the error is chosen as the exact word-level Levenshtein distance to the reference, i.e.

$$\mathcal{C}_{\mathbf{w}}[\boldsymbol{\pi}] = \mathrm{Lev}(\omega(\boldsymbol{\pi}), \mathbf{w}) \; , \tag{3.28}$$

then the criterion optimizes the expectation of the typical evaluation criterion used in speech recognition [Kaiser & Horvat$^+$ 02]. For this reason, the criterion is considered to

be superior to MMI. To enable the use of efficient lattice algorithms, [Povey & Woodland 02] introduced a local approximation of the Levenshtein distance, which is based on the temporal overlap of lattice arcs. Further, [Povey & Woodland 02] observed that the criterion with the local Levenshtein distance on phoneme level performs better than the one defined on word level. This criterion is known as minimum phone error (MPE) and is commonly regarded as the method of choice for discriminative GHMM training.

Various modifications of the MPE criterion have been proposed. [Zheng & Stolcke 05] introduce a slightly different error measure, leading to a criterion known as minimum phone frame error (MPFE). The same criterion on HMM state level is called state-level minimum Bayes risk (sMBR) [Gibson & Hain 06] and has recently been used for sequence-discriminative training of neural networks [Kingsbury & Sainath$^+$ 12].

In contrast to MMI, the MBR objective function is bounded. Therefore, MBR training is more robust to outliers than MMI. On the other hand, the boundedness implies that the MBR objective is not convex.

### 3.3.4 Regularization

Regularization is a standard technique in statistics and machine learning to prevent overfitting. The most commonly used regularizer is the $\ell_2$-norm of the parameters:

$$\mathcal{R} : \mathbf{R}^{J \times A} \to \mathbf{R}, \quad \Lambda \mapsto \|\Lambda\|_2^2 = \sum_{a,j} \lambda_{a,j}^2 \ . \tag{3.29}$$

The $\ell_2$-regularization term penalizes models with extreme parameter values. It is differentiable and strictly convex.

Similarly, $\ell_1$-regularization [Tibshirani 96] penalizes parameters measured by their $\ell_1$-norm:

$$\mathcal{R} : \mathbf{R}^{J \times A} \to \mathbf{R}, \quad \Lambda \mapsto \|\Lambda\|_1^2 = \sum_{a,j} |\lambda_{a,j}| \ . \tag{3.30}$$

$\ell_1$-regularization is especially advantageous if many features are irrelevant [Ng 04]. It favors *sparse* models, i.e. most parameters are zero. In contrast to the $\ell_2$-regularization term, it is not differentiable and only convex, but not strictly convex.

Because of the non-differentiability of the $\ell_1$-regularization term, the optimization is more involved from a theoretical and algorithmical point of view. General gradient-based optimization algorithms need to be adapted to the special case of $\ell_1$-regularized objective functions. The optimum of a non-differentiable convex function can be characterized in terms of the *subgradient*, which is a generalization of the gradient. Let $f : \mathbf{R}^n \to \mathbf{R}$ denote a convex function. Then $g \in \mathbf{R}^n$ is a subgradient of $f$ at $x \in \mathbf{R}^n$ if

$$f(y) \geq f(x) + g^\top (y - x) \tag{3.31}$$

for all $y \in \mathbf{R}^n$. Note that when $f$ is differentiable, then the subgradient coincides with the gradient uniquely. From the definition of the subgradient, it can directly be seen that $x^*$ is the optimum of $f$ if and only if zero is a subgradient of $f$ at $x^*$.

*Early stopping* [Plaut & Nowlan$^+$ 86, Morgan & Bourlard 89] is a simple and widely used alternative to $\ell_2$- and $\ell_1$-regularization. Early stopping terminates training as soon as the error on a validation set increases. In terms of the objective function, early stopping is ill-defined. On the other hand, it does not require tuning of a regularization constant, which makes it well suited for large-scale problems.

## 3.4 Features

Log-linear models induce linear decision boundaries. Non-linear classification is achieved by mapping observations into a high-dimensional feature space. The choice of the feature representation is thus a crucial step.

The features considered in literature can be roughly divided into simple generic features [Kuo & Gao 06, Hifny & Renals 09, Yu & Deng$^+$ 09, Wiesler & Nußbaum$^+$ 09] and complex specialized features [Fosler-Lussier & Morris 08, Heigold & Zweig$^+$ 09]. The hidden part of a neural network with a softmax output layer can also be regarded as a complex feature extractor of a log-linear model. Complex features tend to be more powerful than simple approaches but with the drawback that the main algorithm is possibly outsourced into a non-convex feature extraction part. Since the goal of this chapter is to investigate the feasibility of convex training of acoustic models, we restrict this study to simple generic features.

### 3.4.1 Polynomial features

A polynomial feature of degree $k$ is of the form:

$$f_{(d_1,\ldots,d_k)} : \mathbf{R}^D \to \mathbf{R},\ x \mapsto x_{d_1} \cdot \ldots \cdot x_{d_k}\ , \tag{3.32}$$

with $(d_1, \ldots d_k) \in \{1, \ldots, D\}^k$. Polynomial features are commonly used with SVMs and other kernel methods. A log-linear model with polynomial features up to order one is referred to as a *first-order model*, a model with features up to order two as a *second-order model* and so on. Polynomial models can be interpreted from a statistical point of view. A first-order model is the posterior form of a generative Gaussian model with tied covariance matrices, a second-order model corresponds to a Gaussian model with class-specific covariance matrices. A third-order model also takes class-specific skewnesses of the distributions into account.

Polynomial features are global in the sense that their support is the whole observation space. Therefore they generalize well to unseen data. In principle Weierstrass's approximation theorem [Rudin 76, Chapter 7] states that any continuous function can be approximated arbitrarily well by polynomials on a compact set, which could indicate that polynomial feature functions are a good choice. But for high-dimensional problems, the use of high-order polynomial features is prohibitive, since their number grows exponentially in the order of the model. First and second-order models can be trained quite efficiently, training third-order models is already computationally very expensive. Going beyond third order is yet not feasible.

### 3.4.2 Clustering features

The densities of GMMs trained with the EM-algorithm correspond to localized dense subsets of the feature space. Due to this locality, GMMs can model the training data well. To carry over this idea to the log-linear framework, we can apply the EM algorithm to estimate a single Gaussian mixture distribution for the marginal probability

$$p(x) = \sum_{l=1}^{L} p(l)p(x|l) \ , \tag{3.33}$$

where $L$ is the number of densities. In the log-linear model, we use every density $p(x|l)$ of the GMM to define a feature function

$$f_l(x) = \frac{p(x|l)}{\sum_{l'} p(x|l')}, \ l = 1, \ldots, L \ . \tag{3.34}$$

Because of the exponential decay of Gaussian densities, most feature functions are close to zero for fixed $x$. By setting features below a small threshold to zero, sparse features are obtained, which strongly reduces the computational demands of the gradient accumulation. The number of features can be increased by concatenating features of neighboring time frames. We refer to these features as *clustering features* [Wiesler & Nußbaum$^+$ 09].

Similar features are used in fMPE [Povey & Kingsbury$^+$ 05]. In the log-linear framework, clustering features have first been introduced by [Hifny & Renals 09]. Conceptually, the features are also related to radial basis function kernels used for SVMs. With such local features, log-linear models can approximate the training data very well, but tend to overfit quickly. In terms of the classical bias/variance dilemma [Geman & Bienenstock$^+$ 92], log-linear models with clustering features have a low bias, but a high variance. Conversely, models with polynomial features have a high bias, but a lower variance.

### 3.4.3 Feature selection

Training log-linear models is computationally demanding. Their training time is directly proportional to the feature dimension. Feature selection algorithms aim at identifying a subset of relevant features in a separate pre-processing step and are therefore an attractive method for reducing training time. Moreover, feature selection can improve results by reducing overfitting.

In [Wiesler & Richard$^+$ 11], we empirically compared feature selection algorithms for log-linear models and found ReliefF [Kira & Rendell 92, Kononenko 94] to be a good choice. ReliefF is a state-of-the-art feature selection algorithm, which takes the interaction of features into account. The idea of the algorithm is to measure how well a feature separates neighboring training examples in the original feature space. A small subset of the training data is chosen randomly. For each instance in the subset, the $k$

nearest instances of the same class (*nearest hits*) and all other classes (*nearest misses*) are computed on the complete training data. When a feature of an observation and one of its nearest hits differs, the score of the feature is reduced. Conversely, the feature score is increased, if the feature of the training example and one of its nearest misses is different. The highest-scoring features are selected. The nearest hits and misses are calculated in the original feature space with $\ell_2$-norm.

## 3.5 Parameter Optimization

A major motivation for the use of log-linear models is the convexity of training. In principle, the global optimum can be found from any initialization and with any numerical optimization algorithm with guaranteed convergence. From a naive point of view, the optimization can therefore be viewed as a black box.

For large-scale tasks this point of view is an oversimplification. The global optimum can not be determined exactly, but is approximated using numerical optimizers. Large-scale tasks are characterized by the fact that the quality of this approximation impacts the error rate because of limited computation time [Bottou & Bousquet 08]. Therefore, even for convex problems, the choice of the optimization algorithm and the initialization are important.

In the following, we discuss optimization algorithms, which are relevant for log-linear training. Important properties of optimization algorithms are their convergence speed, memory requirements, parallelism, and practical aspects such as ease of implementation.

The algorithms can be categorized into batch and stochastic algorithms. The updates of batch algorithms are computed on the complete training data. In contrast, stochastic algorithms only require the statistics of a small random subset of the training data. Stochastic algorithms are in particular effective on large and redundant datasets. On the other hand, batch algorithms can be parallelized straightforwardly by distributing the computation of the gradient (data-parallelism).

Beyond of comparing different optimization algorithms, we formally analyze general convergence properties of log-linear training in Chapter 4. We already make use of this analysis in this chapter, and therefore summarize the main result here. Further, we derive a method for transforming features implicitly in the optimization algorithm, which is useful in the context of this convergence analysis.

### 3.5.1 Gradient descent and Newton's method

Although rarely used in practice, gradient descent and Newton's method are the basis of all gradient-based optimization algorithms. Gradient descent is the most simple batch optimization algorithm. In every iteration, the model is updated by

$$\Lambda_i = \Lambda_{i-1} - \eta_i \nabla \mathcal{F}(\Lambda_{i-1}) \;, \tag{3.35}$$

where $\eta_i > 0$ is the step size or learning rate. Gradient descent with an appropriate line-search (for example step-sizes which satisfy the Wolfe conditions [Wolfe 69]) is globally

convergent, i.e. it converges to a local optimum from any initialization [Nocedal & Wright 06, Theorem 3.2]. However, its convergence speed is very slow, cf. Theorem 4.1.

Newton's method is based on a second-order Taylor series approximation of the objective function:

$$\mathcal{F}(\Lambda_k + P) \approx \mathcal{F}(\Lambda_k) + P^\top \nabla \mathcal{F}(\Lambda_k) + \frac{1}{2} P^\top \nabla^2 \mathcal{F}(\Lambda) P \; . \tag{3.36}$$

Minimizing the second-order model of the objective function on the right hand side of Eq. (3.36) yields the Newton update rule:

$$\Lambda_{i+1} = \Lambda_i - \eta_i \nabla^2 \mathcal{F}(\Lambda)^{-1} \nabla \mathcal{F}(\Lambda_i) \; . \tag{3.37}$$

Newton's method converges rapidly in the neighborhood of the solution in terms of the number of iterations. Note that the size of the Hessian matrix is in the order of the square of the dimension of $\Lambda$. This makes Newton's method prohibitive for high-dimensional problems such as all machine learning tasks considered in this work. Practical second-order algorithms use approximations of the inverse Hessian matrix. The general form of such a second-order batch algorithm is

$$\Lambda_{i+1} = \Lambda_i - \eta_i B_i \nabla \mathcal{F}(\Lambda_i) \; , \tag{3.38}$$

where $B_i$ is a positive-definite matrix.

### 3.5.2 L-BFGS

L-BFGS [Liu & Nocedal 89] is the most widely used general-purpose algorithm for large-scale unconstrained optimization. It is generally considered as the best batch algorithm for log-linear training [Malouf 02, Sha & Pereira 03, Wallach 03]. L-BFGS incrementally builds a model of the inverse Hessian matrix using a limited history of the previous gradients and iterates. The algorithm is matrix-free, i.e. rather than storing the matrix $B_i$ explicitly, L-BFGS directly computes its product with the gradient. Thus, the memory requirement of L-BFGS is linear in the number of parameters. No additional statistics have to be accumulated in comparison to gradient descent. Only the parameter update is computationally more expensive, but these costs are negligible in the batch case. However, L-BFGS converges much faster than gradient descent.[2] In addition, a line-search can be avoided in most iterations, because L-BFGS has a natural step size of 1.0. There exists a well-founded convergence theory for L-BFGS, see e.g. [Nocedal & Wright 06, Chapter 8].

### 3.5.3 Rprop

Resilient backpropagation (Rprop) [Riedmiller & Braun 93] is a batch algorithm well-known in the field of neural networks. Rprop uses separate learning rates for all parameters, which corresponds to a diagonal second-order model of the objective function.

[2] As gradient descent, L-BFGS converges with a linear convergence rate, but the constant has a weaker dependence on the condition number, cf. Chapter 4.

The parameter-specific learning rates are computed from sign changes of the gradient. There exist slightly different variants of the basic Rprop algorithm. In this work, we use the iRprop$^+$ variant proposed in [Igel & Hüsken 03]. In order to guarantee Rprop's convergence, a line search has to be employed [Anastasiadis & Magoulas$^+$ 05]. In practice, this is not necessary when a sufficiently small initial learning rate is chosen. Rprop has only few tuning parameters, is simple to implement, and shows good empirical results.

### 3.5.4 Orthant-wise Rprop for $\ell_1$-regularized training

The optimization of $\ell_1$-regularized objective functions requires special care, because the regularization term is non-differentiable. General gradient-based optimization algorithms need to be adapted to the special case of $\ell_1$-regularized objective functions. In [Wiesler & Richard$^+$ 11], we introduced orthant-wise Rprop (OW-Rprop), a modification of Rprop suitable for optimizing $\ell_1$-regularized functions. The idea of our algorithm is analogous to that of orthant-wise L-BFGS proposed in [Andrew & Gao 07].

The algorithm is based on the definition of the *pseudo-gradient* of the objective function. Let $\widetilde{\mathcal{F}}$ denote the loss term of the objective function. Then the pseudo-gradient $\diamond\mathcal{F}$ is defined by

$$\diamond_{s,i}\mathcal{F}(\Lambda) = \begin{cases} \partial_{s,i}\widetilde{\mathcal{F}}(\lambda) + \alpha_{\text{reg}}\,\text{sgn}(\lambda_{s,i}) & , \text{ if } \lambda_{s,i} \neq 0 \\ \partial_{s,i}\widetilde{\mathcal{F}}(\lambda) + \alpha_{\text{reg}} & , \text{ if } \lambda_{s,i} = 0 \text{ and } \partial_{s,i}\widetilde{\mathcal{F}} < -\alpha_{\text{reg}} \\ \partial_{s,i}\widetilde{\mathcal{F}}(\lambda) - \alpha_{\text{reg}} & , \text{ if } \lambda_{s,i} = 0 \text{ and } \partial_{s,i}\widetilde{\mathcal{F}} > \alpha_{\text{reg}} \\ 0 & , \text{ if } \lambda_{s,i} = 0 \text{ and } -\alpha_{\text{reg}} \leq \partial_{s,i}\widetilde{\mathcal{F}} \leq \alpha_{\text{reg}} \end{cases} .$$

Here, $\text{sgn}(x)$ is the sign function. It can be verified elementarily that the pseudo-gradient is a subgradient.[3] Among all subgradients, it is the vector with the maximal number of zero-components. This implies that a parameter vector minimizes the objective function if and only if its pseudo-gradient is zero.

The idea of the algorithm is that the objective function is differentiable, when it is restricted to the orthant (a multidimensional quadrant) containing the current iterate and into which the pseudo-gradient leads. OW-Rprop's parameter update is calculated in the same way as that of Rprop, but with the pseudo-gradient instead of the gradient. Furthermore, the update is constrained to the current orthant, i.e. whenever for Rprop the sign of a parameter changes from one iteration to the next, OW-Rprop sets it to zero.

[3] Use the characterization of the subgradient of the $\ell_1$-norm given in [Boyd & Vandenberghe 14]:

$$g \text{ subgradient of } x \mapsto \|x\|_1 \text{ at } x_0 \Leftrightarrow \|g\|_\infty = 1 \text{ and } g^\top x_0 = \|x_0\|_1$$

### 3.5.5 Growth transformations

In earlier works on log-linear models, the optimization problem has been solved with generalized iterative scaling (GIS) [Darroch & Ratcliff 72] or improved iterative scaling (IIS) [Lafferty & McCallum$^+$ 01]. Later, it has been found that these algorithms converge extremely slowly in comparison to gradient-based numerical optimizers such as L-BFGS [Minka 01, Malouf 02, Sha & Pereira 03]. Therefore algorithms of this type are not considered in this work.

### 3.5.6 Stochastic gradient descent

Stochastic gradient descent is the stochastic version of the gradient descent algorithm discussed above. It is the de-facto standard for neural network training, see Section 1.9.2. For convex optimization, batch algorithms are employed traditionally. Only in recent years, stochastic algorithms have received considerable attention for convex problems [Vishwanathan & Schraudolph$^+$ 06, Shalev-Shwartz & Singer$^+$ 07, Bordes & Bottou$^+$ 09, Schraudolph & Yu$^+$ 07].

It can be shown that if the objective function is strongly convex, the training examples are independent and identically distributed (i.i.d.) and the learning rates fulfill the Robbins-Monroe conditions [Robbins & Monro 51]:

$$\sum_{i=1}^{\infty} \eta_i = \infty \quad \text{and} \quad \sum_{i=1}^{\infty} \eta_i^2 < \infty \,, \tag{3.39}$$

then stochastic gradient descent converges almost surely towards the global optimum, see e.g. [Bottou 98]. In our experiments with log-linear models, we use learning rates defined by

$$\eta_i = \frac{\tau}{\tau + i} \eta_0 \,, \tag{3.40}$$

where $\eta_0$ is the initial learning rate, and $\tau$ controls how fast the learning rate decays over time. I.i.d. training examples are simulated by shuffling the dataset. This requires that all training examples have to be loaded into memory. Note that this is not required for batch algorithms, because they do not depend on the order of the training sample.

Stochastic gradient descent's main advantage is that it frequently updates the model. This is in particular beneficial when working with very large datasets. While batch algorithms can be parallelized easily, this is much more difficult for stochastic algorithms. Data-parallelism is only possible within one mini-batch, which is typically of the size in the order of a few hundred to a few thousands training examples. This computation can be performed efficiently on a single GPU. Recently, there has been success in parallelizing stochastic gradient descent to multiple compute nodes by exploiting its robustness to delayed model updates [Dean & Corrado$^+$ 12, Seide & Fu$^+$ 14, Strom 15]. These approaches are of great practical relevance, but require a carefully optimized implementation and a suitable compute infrastructure.

Stochastic algorithms require careful tuning of learning rates. Usually, many runs of SGD with different learning rate settings have to be performed. Since SGD does not evaluate the objective function on the complete training data, it can not detect that a local optimum has been reached.

### 3.5.7 Stochastic second-order algorithms

In contrast to L-BFGS and Rprop, SGD is a first-order algorithm. Since second-order information is crucial in batch algorithms, it is natural to investigate whether stochastic algorithms can benefit from second-order information as well. The update of stochastic second-order algorithms is of the same form as in the batch case

$$\Lambda_{i+1} = \Lambda_i - \eta_i \, B_i \nabla \mathcal{F}(\Lambda_i, \mathcal{B}_i) \; , \tag{3.41}$$

but with the stochastic gradient $\nabla \mathcal{F}(\Lambda_i, \mathcal{B}_i)$ on a mini-batch $\mathcal{B}_i$ instead of the exact gradient. $B_i$ is again a positive-definite matrix, which is updated in every iteration using information from mini-batch $\mathcal{B}_i$.

[LeCun & Bottou$^+$ 98] proposed to approximate the Hessian by its diagonal. For log-linear models, the diagonal of the Hessian can be computed exactly with only small additional costs. In order to compensate for stochastic noise, the diagonal stochastic Hessian matrix is smoothed using the estimate of the previous iteration and interpolated with the identity matrix.

The wide use of L-BFGS suggests generalizing it to the stochastic case. This algorithm is known as online L-BFGS (oLBFGS) [Schraudolph & Yu$^+$ 07]. As L-BFGS, it uses an approximation to the inverse of the Hessian matrix, which is built from differences of subsequent gradients and models. oLBFGS requires that the gradients which are used for computing the difference are computed on the same mini-batch. This means that two gradient evaluations have to be performed on each mini-batch instead of one as in stochastic gradient descent. Because of the frequent parameter updates in the stochastic case, the higher costs of the update step impact the run-time too (depending on the history size). Convergence of oLBFGS in the convex setting has been proven in [Sunehag & Trumpf$^+$ 09].

### 3.5.8 Convergence properties

The choice of the optimization algorithm is an important factor contributing to the convergence speed of log-linear training. Furthermore, the convergence speed depends on the general difficulty of the optimization problem. In Chapter 4, we analytically study the *condition number* of log-linear training, which gives an accurate description of the difficulty of the problem. The analysis shows that log-linear training can be highly ill-conditioned. The conditioning of the optimization problem can be dramatically improved by affine feature transformations. Note that invertible affine feature transformations do not have an effect on log-linear models when convergence issues and the regularization penalty are not taken into account – for every invertible affine feature

transformation there exists an equivalent model with the same posterior probabilities. According to our analysis, best convergence behavior can be expected with decorrelated features with normalized mean and variance. In the experiments in this chapter, we already make use of this result, but refer to Chapter 4 for the formal analysis.

### 3.5.9 Implicit feature transformation

In principle, the features used for log-linear training can be generated once and then stored to disk. Feature transformations motivated by the above-mentioned convergence properties have therefore only negligible costs in comparison to the optimization. However, often very high-dimensional features are used for log-linear models, which are derived from low-dimensional observations, for example polynomial features. Storing these features is prohibitive. The features have to be computed on-the-fly in every iteration. In particular, decorrelation requires an expensive high-dimensional matrix product. Here, we show that these computations can be avoided by performing the transformation on model side [Wiesler & Schlüter$^+$ 12].

Let $M \in \mathbf{R}^{J \times \tilde{J}}$ and $b \in \mathbf{R}^{\tilde{J}}$ denote the parameters of an affine feature transformation. We use the notation of log-linear models with separate bias parameters from Section 1.9.3. The scores of the log-linear model are

$$\lambda_a^\top (M f(x) + b) + \beta_a = (M^\top \lambda_a)^\top f(x) + \lambda_a^\top b + \beta_a \ . \tag{3.42}$$

Thus, the posterior probabilities of transformed features can be calculated by transforming the model instead of the features. Let

$$(\widetilde{\Lambda}; \widetilde{\beta}) = (M^\top \Lambda; \Lambda^\top b + \beta) \tag{3.43}$$

denote the transformed model parameters. Then the gradient (3.19) can be written as

$$\begin{aligned}
\nabla_{\lambda_a} \mathcal{F}^{(\mathrm{CE})}(\Lambda; \beta) &= \frac{1}{T} \sum_{r=1}^{R} \sum_{t=1}^{T_r} \left( p_{(\Lambda;\beta)}(a | M f(\mathbf{x}_{r,t}) + b) - \delta_{a,a_t} \right) (M f(\mathbf{x}_{r,t}) + b) \\
&= M \ \frac{1}{T} \sum_{r=1}^{R} \sum_{t=1}^{T_r} \left( p_{(\widetilde{\Lambda};\widetilde{\beta})}(a | \mathbf{x}_{r,t}) - \delta_{a,a_t} \right) f(\mathbf{x}_{r,t}) \\
&+ \frac{1}{T} \sum_{r=1}^{R} \sum_{t=1}^{T_r} \left( p_{(\widetilde{\Lambda};\widetilde{\beta})}(a | \mathbf{x}_{r,t}) - \delta_{a,a_t} \right) b \\
&= M \nabla_{\lambda_a} \mathcal{F}^{(\mathrm{CE})}(\widetilde{\Lambda}; \widetilde{\beta}) + \nabla_{\beta_a} \mathcal{F}^{(\mathrm{CE})}(\widetilde{\Lambda}; \widetilde{\beta}) \ b \ .
\end{aligned} \tag{3.44}$$

An analogous identity holds for the gradient with respect to $\beta_a$. Thus, the gradient with the transformed features can be accumulated using the original features and the transformed model (3.43) and then transforming the gradient via (3.44). This procedure reduces the complexity of the affine feature transformation from $\mathcal{O}(TJ\bar{J})$ to $\mathcal{O}(AJ\bar{J})$ and is therefore beneficial when the number of data points is larger than the number of classes.

**Table 3.1.** Results on the WSJ corpus. The 5 120-dimensional sparse features were obtained from the original 9 216-dimensional sparse features with a feature selection algorithm.

| Model | Criterion | Poly. order | #Sparse feat. | WER [%] | |
|---|---|---|---|---|---|
| | | | | Orig. feat. | Norm. feat. |
| GMM | ML | - | - | 3.5 | - |
| | MPE | - | - | **3.2** | - |
| LL | CE | 1 | - | 7.9 | 8.0 |
| | | 2 | - | 4.2 | 4.0 |
| | | 3 | - | 3.7 | 3.5 |
| | | - | 9216 | 4.4 | - |
| | | 2 | 9216 | 3.6 | **3.2** |
| LL | CE | 2 | 5120 | 3.6 | - |
| | CE + $\ell_1$ | | 9216 | 3.9 | - |
| | CE + $\ell_1$ | | 5120 | 4.1 | - |

## 3.6 Experimental Results

In this section, we present experimental results with hybrid log-linear HMMs on medium to large-scale speech recognition tasks. In a first set of experiments on the medium-scale Wall Street Journal (WSJ) task, we compare the features which we introduced above. The training scheme is then scaled to a recent LVCSR task. Finally, we present an empirical comparison of optimization algorithms.

### 3.6.1 Comparison of features

As a first step towards a log-linear LVCSR system, we experimented with the choice of features on the Wall Street Journal corpus. The dataset and the Gaussian mixture baseline system are described in detail in Appendix A.1. Often, the Wall Street Journal corpus is considered as an LVCSR task, but with a recognition vocabulary of 5k words and 15 hours training data, it is quite small in comparison to recent corpora. The results on this corpus need to be interpreted with care, because the evaluation corpus has only 5k running words. This means, that a difference of 0.1 percent word error rate corresponds to only five errors. Nevertheless, the corpus is a good choice for first experiments, because it requires the most important LVCSR techniques and it allows for performing a large number of comparative experiments.

As a proof-of-concept, we trained the log-linear system on the Wall Street Journal task completely from scratch, i.e. even without an alignment from a Gaussian mixture system. We first assumed a linear time alignment and performed a realignment with the converged model. After several cycles of training and realignment, we obtained a

reasonable alignment, which has been used for all reported experiments.

All log-linear systems have been trained with Rprop with a maximum of 100 iterations according to the cross-entropy criterion. We used the state tying of the Gaussian mixture baseline system. The best iterate has been selected on the development set. The state priors were set to the relative frequencies calculated from the alignment used in training.

In our experiments, we observed that the log-linear emission scores have a different range than the Gaussian mixture scores. Therefore, we scaled the emission scores in Eq. (1.37) with a factor $\kappa > 0$. The optimized values for $\kappa$ in our experiments range from 0.3 to 5.0. Note that one can equivalently retune the transition probabilities and the language model scale. We have not observed improvements from optimizing the prior scale and set it to 1.0 in all recognitions.

The polynomial features were constructed from the same LDA-transformed features, which have been used for the Gaussian mixture baseline system. We tested first-, second-, and third-order polynomial features. The features have a dimension of 33, 594, and 7 139. The training of the third-order model has been initialized with the second-order model, because this training is computationally expensive even on this moderately-sized task.

The clustering features were constructed from a Gaussian mixture with tied diagonal covariance matrices. The Gaussian mixture model has been trained with the EM algorithm with a splitting procedure. Note that the EM algorithm has been applied to the acoustic training vectors without considering class labels. This means we did not rely on an existing GHMM system. In preliminary experiments [Wiesler & Nußbaum$^+$ 09], we found that the error rate saturates with clustering features obtained after ten splits and a temporal context of nine. The resulting sparse feature dimension was 1 024, respectively 9 264 after context expansion. On average, only 30.6 components of the 9 264-dimensional sparse feature vector were non-zero, which strongly accelerated the gradient accumulation.

The recognition results are shown in Table 3.1. The baseline system achieves a word error rate of 3.5 percent with maximum likelihood training. The result is improved to 3.2 percent word error rate after MPE training.

The log-linear system with first-order polynomial features has an error rate of 7.9 percent. By increasing the polynomial order, the error rate decreases to 4.2 (second-order features) and 3.7 percent (third-order features). Training a forth-order model with 66 045 dimensional features is not only currently infeasible. Considering the moderate improvement from second to third-order features, one can not expect substantial further improvements from increasing the polynomial order.

Normalizing the features according the convergence analysis mentioned in the previous section consistently improves the optimization behavior. For these experiments, we normalized the mean and variance of the polynomial features. The second-order features were additionally decorrelated. The original first-order features were already decorrelated. We have not investigated the effect of decorrelation of the third-order features because of their high dimension. The feature normalization improves the recognition

accuracy of polynomial models. For the first-order model, optimization is not an issue. The second-order and third-order models are both improved by 0.2 percent word error rate. The error rate of the third-order model is still worse than the discriminatively trained baseline system, but it already reaches the error rate of the maximum likelihood baseline.

The log-linear model with only sparse features performs worse than the model with polynomial features. But the combination of second-order features and sparse features outperforms the third-order model. The log-linear system now achieves the same error rate as the MPE-trained Gaussian mixture system.[4] The feature normalization not only improves the recognition accuracy. The best result is already obtained after only 16 Rprop iterations in contrast to 68 with the unnormalized features.

### 3.6.2 Feature selection

In order to speed up training, we investigated feature selection methods on the log-linear Wall Street Journal system. We applied the ReliefF feature selection algorithm described in Section 3.4.3 to the clustering features. Further, we experimented with $\ell_1$-regularization using the OW-Rprop algorithm described in Section 3.5.4.

The training subset required for ReliefF was set to 650 training examples. The number of nearest hits and misses was set to 50. We observed that the algorithm is robust to the choice of the tuning parameters. The results in Table 3.1 show that ReliefF allows for reducing the number of sparse features moderately from 9216 to 5120 without degradation in word error rate.

For the experiments with $\ell_1$-regularization, we set the regularization constant to $\alpha_{\text{reg}} = 100/T = 1.85\text{e-}5$. We only penalized the parameters corresponding to the polynomial features. We observed a similar convergence speed of OW-Rprop and Rprop. Roughly sixty percent of the polynomial-feature parameters of the final model were zero. However, the results in Table 3.1 show that although OW-Rprop works well as an optimization algorithm, the $\ell_1$-penalty degrades recognition accuracy. Of course, one could set the regularization constant to an insignificant value, but then it is not possible to exploit model sparsity for accelerating training.

Overall, the experience from our experiments is that there are simpler and more effective methods for accelerating training than general feature selection methods. In particular, improving the convergence behavior by feature normalization reduced the number of Rprop iterations by a factor of four. Therefore, we have not have not investigated feature selection methods further.

### 3.6.3 Experiments on LVCSR

The findings on the Wall Street Journal corpus allowed us to scale the log-linear system to larger tasks. We chose the English Quaero 2010 corpus for the evaluation of the

[4] The result reported here is slightly better than the one in [Wiesler & Schlüter$^+$ 11]. The reason is that we previously used incorrect state priors, which have been calculated on a different alignment.

**Table 3.2.** Results on the English Quaero 2010 corpus. The last row is the result of the system combination with the log-linear and the MPE-trained GMM system.

| **Model** | **Criterion** | **WER** [%] | | |
|---|---|---|---|---|
| | | Dev2010 | Eval2010 | Eval2011 |
| GMM | ML | 25.5 | 25.1 | 32.2 |
| | MPE | 24.0 | 24.0 | 30.6 |
| LL | CE | 24.2 | 24.0 | 30.8 |
| System combination | - | 22.2 | 22.3 | 28.9 |

log-linear approach. The task of the Quaero corpus is the transcription of English conversational data. It is a challenging task due to the variability of the audio data, the speaking style, and frequent speaker changes. The training set contains 103 hours of speech data. The baseline system is a simplified version of the evaluation system used in the Quaero 2010 evaluation [Sundermeyer & Nußbaum-Thom$^+$ 11]. The corpus and the GHMM baseline system are described in detail in Section A.2.1.

Similarly to the Wall Street Journal setup, we used a combination of second-order polynomial features and sparse clustering features with context expansion. The LDA-transformed features used for the Gaussian mixture are 45-dimensional. The second-order polynomial features derived from these features have a dimension of 1 080. The polynomial features were decorrelated and their mean and variance were normalized. For the sparse clustering features, we trained a Gaussian mixture for the marginal probability with 12 splits. The resulting sparse features with a context size of nine have a dimension of 36 864.

In contrast to the Wall Street Journal setup, the log-linear training on Quaero was based on the maximum likelihood alignment. The log-linear model has been trained according to the cross-entropy criterion using Rprop. The gradient accumulation has been accelerated using the implicit feature transformation derived in Section 3.5.9. The best iterations for the log-linear training and the MPE training of the Gaussian mixture system have been selected on the development data. The best log-linear model has been obtained after 16 iterations, the best GMM after 20 iterations. Using a CPU-based implementation, the computation time of one Rprop iteration with the log-linear model is roughly five times larger than one GMM/MPE iteration. On the other hand, the frame-discriminative log-linear approach does not require the generation of word lattices. Since the training scheme can be parallelized easily, the proposed training scheme is scalable to large tasks.

The recognition results are summarized in Table 3.2. The log-linear model and the MPE-trained Gaussian mixture system perform equally well. In addition, we performed a confusion network combination [Evermann & Woodland 00] of the log-linear and the

**Table 3.3.** Results on the English Quaero 2011 / Train-50h corpus

| Model | Criterion | WER [%] | |
|---|---|---|---|
| | | Dev2011 | Eval2011 |
| GMM | ML | 24.4 | 31.6 |
| | MPE | 23.6 | 30.2 |
| LL | CE | 23.6 | 30.6 |

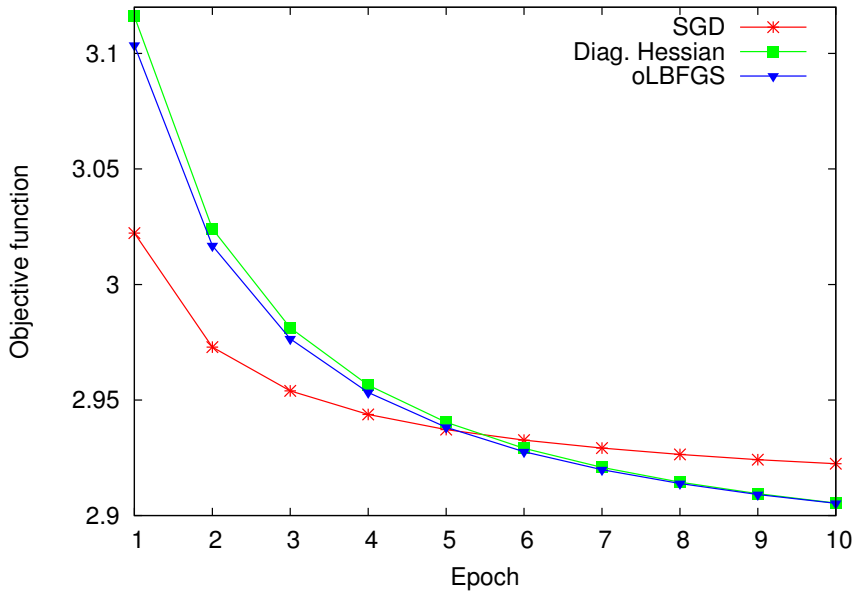


**Figure 3.1.** Objective function of SGD, oLBFGS, and the diagonal Hessian method on the training data

MPE-trained Gaussian mixture system. Strong improvements are obtained, although both systems use the same acoustic front end and the same state tying.

In order to be able to compare our approach to more recent results, we trained a log-linear model on the 50 hour subset of the Quaero 2011 corpus, cf. Appendix A.2.2. The model has been trained analogously to the one on the Quaero 2010 corpus. We obtained a slight gain of 0.3 percent word error rate by increasing the context size of the clustering features from 9 to 17. The recognition results are summarized in Table 3.3. Again, the log-linear model performs about as well as the discriminatively trained Gaussian mixture system.

**Table 3.4.** Comparison of optimization algorithms on the English Quaero 2010 corpus. Recognition results on the Dev2010 corpus and the training objective function values for trainings with original and normalized features

| | | Normalized Features | | Original Features | |
|---|---|---|---|---|---|
| **Algorithm** | **Iteration** | **WER [%]** | **Obj. fct.** | **WER [%]** | **Obj. fct.** |
| Rprop | 50 | 25.6 | 2.985 | - | - |
| | 100 | 25.3 | 2.935 | - | - |
| L-BFGS | 50 | 27.8 | 3.194 | - | - |
| | 100 | 26.1 | 3.004 | - | - |
| SGD | 5 | 25.3 | 2.937 | 26.6 | 3.050 |
| | 10 | 25.2 | 2.922 | 26.2 | 3.017 |
| oLBFGS | 5 | 25.1 | 2.938 | 33.1 | 4.109 |
| | 10 | 24.9 | 2.905 | 32.3 | 4.055 |
| Diagonal Hessian | 5 | 25.1 | 2.940 | 25.7 | 3.020 |
| | **10** | **24.8** | **2.905** | 25.8 | 3.007 |

### 3.6.4 Comparison of optimization algorithms

The experiments on the Wall Street Journal corpus show the importance of the performance of the optimizer for log-linear training. In the previous sections, all log-linear trainings have been performed with Rprop. In this section, an empirical comparison of optimization algorithms on the Quaero 2010 corpus is presented [Wiesler & Richard$^+$ 13].

More specifically, we compare Rprop to L-BFGS, which is commonly regarded as the best batch algorithm for log-linear training [Malouf 02], and to stochastic optimization algorithms, which are commonly used for neural network training. In particular, we evaluate whether stochastic second-order algorithms, which recently received attention in the machine learning literature, are beneficial in a large-scale setting.

The log-linear system has the same state tying as the Gaussian mixture system. We used second-order polynomial features, which were derived from the baseline features of the Gaussian mixture system. Mean and variance of the second-order features were normalized to improve the convergence behavior. The dimension of the second-order features is 1 080. The number of parameters of the log-linear model is 4.9 million, and thus only about one tenth of the Gaussian mixture model. We chose this relatively small model, because overfitting does not occur with this configuration. This allows for a more direct comparison of the optimization algorithms.

The history size of L-BFGS was set to 20. For the experiments with stochastic

algorithms, we used decaying learning rates of the form in Eq. (3.40). The learning rate parameters $\eta_0$ and $\tau$ were optimized using the first one million frames. We used mini-batches of size 4 500. We tested smaller mini-batch sizes as well, but the results were slightly worse. In the experiments, we observed that it is important to use a smaller learning rate for the bias parameters than for the other parameters (0.1 times the standard learning rate in our experiments). The computation of the batch-gradient has been distributed to 100 CPUs. For stochastic algorithms, multi-threading with eight threads has been used.

The experimental results are shown in Table 3.4. First, notice that Rprop is more than two times faster than the widely used L-BFGS algorithm.

Rprop takes about 38 iterations to reach the objective function value of SGD after only a single pass over the data. However, in the late phase of convergence, SGD is quite slow. Although Rprop's overall computation time is much higher than that of SGD, Rprop is competitive to SGD in wall clock time due to the better parallelizability. Which algorithm is preferable depends on the size of the dataset and on the amount of computational resources, which one is willing to spend.

In the next series of experiments, we compared the performance of SGD with the stochastic second-order algorithms oLBFGS and the diagonal Hessian method, which have been described in Section 3.5.7. The investigated stochastic second-order algorithms converge slightly faster than SGD in terms of the number of iterations, see Fig. 3.1. During the first epochs, the second-order algorithms perform worse than SGD. But closer to the solution, they converge faster than SGD. Note that the second-order algorithms require additional computations. In particular, oLBFGS requires two gradient evaluations per batch, hence doubling the computational demands. In contrast, the additional computational costs of the diagonal Hessian method are low. The best recognition result is achieved with the diagonal Hessian method after ten epochs, see Table 3.4. The word error rate of this simple second-order model is between that of the maximum likelihood trained Gaussian mixture system and the discriminatively trained Gaussian mixture system, compare Table 3.2.

Finally, the results in Table 3.4 also show that normalizing the features for improving the convergence behavior is not only important important for batch, but also for stochastic optimization algorithms. Without feature normalization, SGD and the diagonal Hessian method degrade by about one percent word error rate, and oLBFGS even much more.

## 3.7 Discussion

In this chapter, we presented an approach to acoustic modeling based on log-linear models. The frame-discriminative training of the log-linear acoustic model is convex. Thus, the model can be trained discriminatively from scratch, i.e. without using a generative model as initialization. The training does not require word lattices, which makes the approach algorithmically simple and saves engineering work. In our experiments on

medium to large-scale speech recognition tasks, the log-linear model achieved about the same performance as the sequence-discriminatively trained Gaussian mixture baseline system.

Two aspects have turned out to be critical for reaching this performance. The first one is the choice of features for the log-linear model. Here, we concentrated on generic features, which are restricted to the emission model. The second one is the numerical optimization of the log-linear parameters. We observed that the choice of the optimization algorithm as well as the conditioning of the optimization problem have an important impact on the recognition error rate.

In contrast to other works on log-linear acoustic modeling, our approach is scalable to large tasks without losing the convexity of the training. Most works on log-linear models have been conducted on small-scale tasks, in particular [Macherey & Ney 03, Heigold & Rybach$^+$ 09, Hifny & Renals 09, Ragni & Gales 11, van Dalen & Ragni$^+$ 13]. The effect of both, the choice of features and the convergence behavior, differs strongly on small-scale and large-scale tasks. On small-scale tasks, already linear features provide good results. Optimization is not relevant, because the optimization can be run until the unique global optimum is found with arbitrary precision.

Log-linear models with latent variables allow for increasing the model complexity straightforwardly [Gunawardana & Mahajan$^+$ 05]. Therefore, this approach is also well-suited for large-scale tasks [Heigold & Ney$^+$ 11]. However, the convexity of training is then lost. It is difficult to train such models without using a generative model as initialization. For this reason, log-linear models with latent variables are strongly similar to conventional discriminatively trained GHMMs.

In parallel to our work, large performance gains over Gaussian mixture models and thus also over log-linear models have been achieved with neural network-based acoustic models [Mohamed & Dahl$^+$ 09, Dahl & Ranzato$^+$ 10, Seide & Li$^+$ 11b, Sainath & Kingsbury$^+$ 11, Dahl & Yu$^+$ 12, Jaitly & Nguyen$^+$ 12]. This approach is strongly related to the log-linear model studied here. The neural networks have a log-linear output layer and are typically trained according to the cross-entropy criterion. In recognition, the hybrid approach is used. The essential difference between log-linear models and neural networks is that neural networks jointly learn the classifier and the feature space, whereas the features for log-linear models are specified manually. The drawback of neural networks is that their training is not convex and the global optimum is not accessible. However, it has turned out that it is more important to enable the model to learn the features than enforcing convexity, especially on larger datasets. For instance, already a neural network with only a single hidden layer outperforms the log-linear model developed here.[5]

There are two natural extensions of the work presented in this chapter. Since the performance of the optimizer is already important for simple convex models, can neural networks be improved with better optimization algorithms as well? This question is addressed in Chapter 5 and Chapter 6. Secondly, we only considered frame-discriminative

[5] compare Table 3.3 and Table 7.3

training of log-linear models here. One can expect that these models can be further improved with sequence-discriminative training. We study this technique in the context of neural networks in Chapter 7.

## 3.8 Publications and Joint Work

This work builds on G. Heigold's work on log-linear models [Heigold 10]. In particular the initial implementation of the hybrid log-linear approach is by G. Heigold. The focus of G. Heigold's work in the log-linear framework was on training criteria, which have mostly been studied on small-scale tasks such as a German digit string recognition task (SieTill) [Heigold & Rybach$^+$ 09]. For large-scale tasks, non-convex log-linear models with latent variables have been employed [Heigold & Deselaers$^+$ 08, Heigold & Wiesler$^+$ 10, Heigold & Ney$^+$ 11].

The novel contribution of the work presented in this chapter is the extension of the *convex* log-linear approach to large-scale tasks by investigating the choice of feature functions and improving the optimization [Wiesler & Nußbaum$^+$ 09, Wiesler & Richard$^+$ 11, Wiesler & Schlüter$^+$ 12, Wiesler & Richard$^+$ 13]. Detailed author contributions for these publications are given in Appendix E. The proof-of-concept of training a log-linear model directly from a linear time alignment without an existing GHMM system has been contributed by M. Nußbaum-Thom. A part of the experiments has been performed by A. Richard under the author's supervision.

# Chapter 4

# Convergence Analysis of Log-Linear Training

In this chapter, we present a theoretical analysis of the convergence behavior of log-linear training. Our analysis shows that log-linear training can be highly ill-conditioned. In this case, the global optimum of the objective function can not be found efficiently. Conversely, the conditioning of the optimization problem can be improved strongly by feature normalizations. We verify our findings on two handwriting recognition tasks.

## 4.1 Introduction

The convexity of log-linear training could possibly lead to the misconception that optimization is of minor importance for log-linear models. However, this is not valid for large-scale problems. The solution of log-linear training is determined using iterative numerical optimization algorithms. Large-scale problems can be characterized by the fact that computation time is limited, which only allows for an approximate solution of the optimization problem [Bottou & Bousquet 08]. In this case, the test error depends on the error caused by this approximation. This is in particular relevant, when the optimization problem is difficult.

From optimization theory it is known that the convergence speed of a wide range of optimization algorithms can be characterized in terms of the condition number of the Hessian matrix at the optimum, i.e. by the ratio of the largest and the smallest eigenvalue of the matrix. The following theorem describes the convergence behavior of gradient descent applied to quadratic functions. It can be found in many textbooks, for example in [Luenberger & Ye 08, Chapter 8].

**Theorem 4.1.** *Let*

$$f : \mathbf{R}^n \to \mathbf{R}, \quad x \mapsto x^\top A x + b^\top x + c \tag{4.1}$$

*be a strictly convex quadratic function with $A \in \mathbf{R}^{n \times n}$ and $b, c \in \mathbf{R}^n$. Let $f^*$ be the minimum of $f$. Let $\kappa$ denote the condition number of $A$, i.e. the ratio of the largest and smallest eigenvalue of $A$. Then the iterates generated by gradient descent with exact line search satisfy*

$$f(x_{k+1}) - f^* \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^2 (f(x_k) - f^*) \,. \tag{4.2}$$

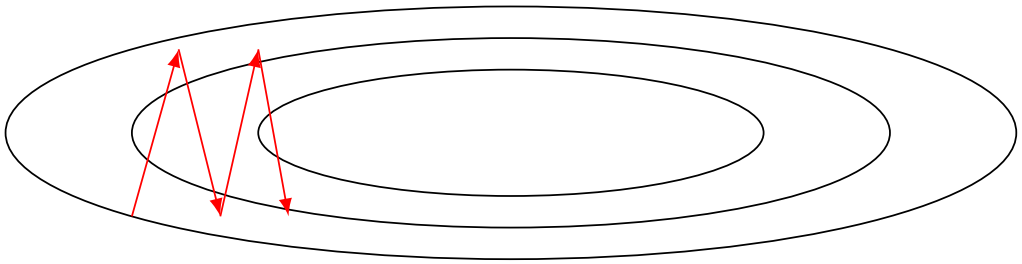

**Figure 4.1.** Illustration of the "zig-zag" behavior of gradient descent on a two-dimensional quadratic optimization problem

The constant which describes the improvement of gradient descent, is bounded by

$$\frac{\kappa - 1}{\kappa + 1} \leq 1 - \frac{1}{\kappa} \,. \tag{4.3}$$

For large condition numbers, the constant is close to one, thus convergence is slow. The bound in Eq. (4.2) is tight in the sense that there exists a starting point for which Eq. (4.2) is an equality. The role of the condition number for gradient descent is illustrated in Fig. 4.1. On poorly conditioned problems, gradient descent shows a characteristic "zig-zag" behavior, leading to slow convergence. Theorem 4.1 can be generalized to non-quadratic strictly convex functions. In this case, the asymptotic behavior of gradient descent is determined by the condition number of the Hessian matrix at the optimum, see [Nocedal & Wright 06, Theorem 3.4].

Gradient descent exhibits the strongest dependence of the convergence speed on the condition number. More sophisticated gradient-based optimization algorithms as conjugate gradient (CG) and L-BFGS depend on the condition number and other properties of the eigenvalue spectrum [Nocedal & Wright 06, Chapter 5.1 and Chapter 9.1]. Letting apart that Newton's method is not feasible for high-dimensional problems, its convergence behavior is in principle completely independent of the condition number. In practice, even Newton's method depends on the condition number, because computing the Newton direction requires solving a system of linear equations, which is more difficult for problems with high condition number [Boyd & Vandenberghe 09, Chapter 9.5].

In this chapter, we derive bounds for the condition number of the cross-entropy objective function for log-linear models. Our analysis shows that convergence can be accelerated by feature transformations. We verify our analytic results on two classification tasks. One is a small digit recognition task, the other a large-scale continuous handwriting recognition task. The experiments show that in extreme cases, log-linear training can be so ill-conditioned that a useful model can only be found from a reasonable initialization. On the other hand, when care is taken, we obtain good results with a conceptually simple and generic approach.

## 4.2 Formal Analysis

This section contains our theoretical result. We derive bounds for the eigenvalues of the Hessian of the cross-entropy objective function, which determine the convergence behavior of gradient-based optimization algorithms. First, we relate the eigenvalues of the Hessian to those of the uncentered covariance matrix. Then, bounds for the condition number of the uncentered covariance matrix are derived.

### 4.2.1 Preliminaries

To emphasize the generality of our analysis, we use the general notation for log-linear models from Chapter 1 instead of the speech recognition specific notation. We avoid using explicit bias parameters by extending the feature vector by a component which is constantly 1.0. The log-linear model is then of the form

$$p_\Lambda(c|x) = \frac{1}{Z(x)} \exp\left(\lambda_c^\top x\right) \quad \text{with} \quad Z(x) = \sum_{\bar{c}} \exp\left(\lambda_{\bar{c}}^\top x\right) . \tag{4.4}$$

The parameters of the model are of the form $\Lambda = (\lambda_1; \ldots; \lambda_C) \in \mathbf{R}^{(D+1)\times C}$. The $(D+1)$-th feature component is the constant one.

Let $(x_n, c_n)_{n=1,\ldots,N}$ denote the training sample. Our analysis concerns the $\ell_2$-regularized cross-entropy objective function:

$$\mathcal{F} : \mathbf{R}^{(D+1)\times C} \to \mathbf{R}, \quad \Lambda \mapsto -\frac{1}{N} \sum_{n=1}^{N} \log p_\Lambda(c_n|x_n) + \alpha_{\text{reg}} \|\Lambda\|_2^2 . \tag{4.5}$$

Although the objective function is convex, it is not guaranteed that it attains the infimum. For example if the data is linearly separable and no regularization is used, the parameters can get arbitrarily large. In the following, we restrict ourselves to the non-degenerate case, where the minimum is attained, i.e. we assume that there exists a $\Lambda^*$ with

$$\mathcal{F}(\Lambda^*) = \inf_\Lambda \mathcal{F}(\Lambda) . \tag{4.6}$$

The second derivatives of the objective function are

$$\frac{\partial^2 \mathcal{F}}{\partial \lambda_{c,j} \partial \lambda_{\bar{c},\bar{j}}}(\Lambda) = \frac{1}{N} \sum_{n=1}^{N} p_\Lambda(c|x_n)(\delta_{c,\bar{c}} - p_\Lambda(\bar{c}|x_n))\, x_{n,j} x_{n,\bar{j}} + \alpha_{\text{reg}}\, \delta_{c,\bar{c}}\, \delta_{j,\bar{j}} . \tag{4.7}$$

The gradient and the Hessian matrix of $\mathcal{F}$ are defined in terms of the vectorized parameters.

In the following, we require basic definitions and results from matrix analysis. A comprehensive reference is [Horn & Johnson 05]. Recall that the eigenvalues of a symmetric real matrix are real. We denote the eigenvalues of a symmetric real matrix $A \in \mathbf{R}^{n\times n}$ in ascending order by

$$\vartheta_1(A) \le \vartheta_2(A) \le \ldots \le \vartheta_n(A) . \tag{4.8}$$

The *spectrum* of the matrix is the set of all eigenvalues:

$$\sigma(A) = \{\vartheta_1(A), \ldots, \vartheta_n(A)\} \; . \tag{4.9}$$

The *condition number* of a positive definite matrix $A \in \mathbf{R}^{n \times n}$ is defined as the ratio between the largest and the smallest eigenvalue:

$$\kappa(A) = \frac{\vartheta_n(A)}{\vartheta_1(A)} \; . \tag{4.10}$$

Below, we will exploit that the calculation of the spectrum is simplified if the matrix has a Kronecker product structure, cf. [Horn & Johnson 94, Chapter 4]. Given two matrices $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{p \times q}$, the Kronecker product is defined as

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix} \in \mathbf{R}^{mp \times nq} \; . \tag{4.11}$$

### 4.2.2 The case without regularization

We begin with the analysis of the loss term, i.e. we assume that $\alpha_{\text{reg}}$ is zero. Only this term shows a complex behavior. The Hessian matrix of the objective function at the optimum depends on the posterior probabilities $p_{\Lambda^*}(c|x)$, which are of course unknown. In the following, we consider a simpler problem. We derive the eigenvalues of the Hessian at $\Lambda_0 = 0$. If the quadratic approximation of $\mathcal{F}$ at $\Lambda_0$ is good, the Hessian does not change strongly from $\Lambda_0$ to $\Lambda^*$ , and the eigenvalues of $\nabla^2\mathcal{F}(\Lambda_0)$ are close to those of $\nabla^2\mathcal{F}(\Lambda^*)$. This enables us to draw conclusions about the convergence behavior of gradient-based optimization algorithms. The experiments in Section 4.3 justify our assumption. All experimental results are in accordance to the theoretical results.

The following definition plays a central role in the convergence analysis:

**Definition 4.1.** The *uncentered (empirical) covariance* matrix is defined as

$$V = \frac{1}{N} \sum_{n=1}^{N} x_n x_n^\top \in \mathbf{R}^{(D+1) \times (D+1)} \; . \tag{4.12}$$

In the following, we assume that the covariance matrix of the non-trivial features $x_1, \ldots, x_D$ is positive definite, which is the case if there are no linear dependencies in the features and the number of training sample is sufficiently large. In the next section, we show that this also implies that $V$ is non-singular.

Now we can formulate the main result of this section.

**Theorem 4.2.** *The spectrum of the Hessian matrix $\nabla^2\mathcal{F}(\Lambda_0)$ is determined by the spectrum of the uncentered empirical covariance matrix:*

$$\sigma(\nabla^2\mathcal{F}(\Lambda_0)) = \{0\} \cup \{C^{-1}\vartheta_1(V), \ldots, C^{-1}\vartheta_{D+1}(V)\} \; . \tag{4.13}$$

*Further, the rank of $\nabla^2\mathcal{F}(\Lambda_0)$ is $(C-1)(D+1)$.*

*Proof of Theorem 4.2.* The proof is based on the Kronecker product structure of the Hessian at $\Lambda_0$.

For $\Lambda_0 = 0$, the posterior probabilities are uniform, i.e.

$$p_{\Lambda_0}(c|x) = C^{-1} \tag{4.14}$$

for all observations $x$ and classes $c$. Hence, the second derivatives of $\mathcal{F}$ are

$$\frac{\partial^2 \mathcal{F}}{\partial \lambda_{c,j} \partial \lambda_{\bar{c},\bar{j}}}(\Lambda_0) = C^{-1} \left(\delta_{c,\bar{c}} - C^{-1}\right) \frac{1}{N} \sum_{n=1}^{N} x_{n,j} x_{n,\bar{j}} \,. \tag{4.15}$$

Let $I_C \in \mathbf{R}^{C \times C}$ denote the $C$-dimensional identity matrix and $\mathbf{1}_C \in \mathbf{R}^{C \times C}$ the matrix, where all entries are equal to one. With the definition

$$S = C^{-1} \left(I_C - C^{-1} \mathbf{1}_C\right) \,, \tag{4.16}$$

the Hessian matrix can be written as a Kronecker product:

$$\nabla^2 \mathcal{F}(\Lambda_0) = S \otimes V \,. \tag{4.17}$$

The eigenvalues of $S$ can be computed easily:

$$\sigma(S) = \{0, C^{-1}\} \,. \tag{4.18}$$

The rank of $S$ is $C - 1$. The spectrum of the Kronecker product is given by [Horn & Johnson 94, Theorem 4.2.12]

$$\sigma(S \otimes V) = \{\vartheta_i(S)\vartheta_j(V) | 1 \leq i \leq C, 1 \leq j \leq D+1\} \,, \tag{4.19}$$

which proves Eq. (4.13). Further, the rank of the Kronecker product is the product of the rank of $S$ and $V$ [Horn & Johnson 94, Chapter 4.2], which concludes the proof. □

A difficulty in the analysis of the unregularized case is that the objective function is only convex, but not strictly convex. The Hessian matrix is singular, thus the condition number is not defined. From Eq. (4.17) it follows that the null space of the Hessian is spanned by the $(D+1)$ columns of

$$U = \frac{1}{\sqrt{C}} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \otimes I_{D+1} \in \mathbf{R}^{C(D+1) \times (D+1)} \,. \tag{4.20}$$

Note that these basis vectors exactly correspond to the invariance of log-linear models to parameter shifts, cf. Section 3.2.1. Any change in the direction of a column of $U$ does not change the objective function, and the directional derivative in such a direction is zero. Therefore, the optimization problem is essentially a $(C-1)(D+1)$-dimensional problem. The convergence behavior of the problem is determined only by the non-zero

eigenvalues. In our context it is therefore appropriate to define the condition number of a positive semi-definite matrix as the the ratio of the largest and the smallest non-zero eigenvalue.

$$\kappa(A) = \frac{\vartheta_n(A)}{\min_{i:\vartheta_i(A)\neq 0} \vartheta_i(A)} \ . \tag{4.21}$$

With this definition of the condition number, Theorem 4.1 can directly be generalized to the singular case. An analog result was shown by [Notay 90] for the application of CG for solving systems of linear equations, which is equivalent to the minimization of quadratic functions. All results about the convergence behavior of conjugate gradient extend to the singular case, if instead of the complete spectrum only the non-zero eigenvalues are considered. In the next section, we analyze the condition number of the uncentered covariance matrix.

### 4.2.3 Spectrum of the uncentered covariance matrix

The dependence of the convergence behavior on the properties of the uncentered covariance matrix is in accordance to experimental observations. Other researchers have noted before, that the use of correlated features leads to slower convergence, e.g. [Sha & Pereira 03]. [Minka 01] noted that convergence is slower when adding a constant to the features, because this "introduces correlation, in the sense that" the uncentered covariance matrix "has significant off-diagonals.". How can these these findings be verified formally?

The following theorem addresses the case of uncorrelated features. Note that we need to distinguish between the non-trivial first $D$ components of the feature vector and the constant feature component, corresponding to the bias parameters.

**Theorem 4.3.** *Suppose the features $x_1, \ldots x_D$ are uncorrelated with respect to the empirical distribution. Let $\mu, \sigma^2 \in \mathbf{R}^D$ denote the empirical mean and variance vector of $(x_1, \ldots x_D)$. Without loss of generality, we assume that the first $D$ features $x_1, \ldots x_D$ are ordered such that*

$$0 < \sigma_1^2 \leq \ldots \leq \sigma_D^2 \ . \tag{4.22}$$

*The extremal eigenvalues of the uncentered empirical covariance matrix are bounded by:*

$$\max\left\{0, \min\left\{1, \sigma_1^2\right\} - \|\mu\|\right\} \leq \vartheta_1(V) \leq \sigma_1^2 \tag{4.23}$$

$$\max\left\{\|\mu\|^2 + 1, \sigma_D^2 + \mu_D^2\right\} \leq \vartheta_{D+1}(V) \leq \sigma_D^2 + \|\mu\|^2 + 1 \ . \tag{4.24}$$

*Furthermore, the minimal eigenvalue $\vartheta_1(V)$ is strictly positive.*

Theorem 4.3 shows that even with uncorrelated features, the optimization problem can be arbitrarily poorly conditioned. A non-zero feature mean causes a large maximal eigenvalue. With zero mean, the matrix can also be ill-conditioned due to a different scaling of the features.

The proof of Theorem 4.3 is based on Weyl's inequalities, see [Horn & Johnson 05, Theorem 4.3.7]:

**Theorem 4.4** (Weyl's inequalities)**.** *Let $A, B \in \mathbf{R}^{n\times n}$ denote symmetric real matrices. Then, the following bounds on the eigenvalues of $A + B$ hold for all $1 \leq j, k \leq n$:*

$$\lambda_{j+k-n}(A+B) \leq \lambda_j(A) + \lambda_k(B) \,, \tag{4.25}$$
$$\lambda_{j+k-1}(A+B) \geq \lambda_j(A) + \lambda_k(B) \,. \tag{4.26}$$

*Proof of Theorem 4.3.* Since the features are uncorrelated, we have

$$V = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_\mathrm{D}^2, 0) + \bar{\mu}\bar{\mu}^\top \,, \tag{4.27}$$

with

$$\bar{\mu} = \begin{pmatrix} \mu \\ 1 \end{pmatrix} \in \mathbf{R}^{D+1} \,. \tag{4.28}$$

Denote the left summand in Eq. (4.27) as $A$ and the right summand as $B$. The eigenvalues of $A$ are the diagonal elements. $B$ is a rank-one matrix with eigenvalues $\vartheta_{D+1}(B) = 1 + \|\mu\|^2$ and $\vartheta_j(B) = 0$ for $1 \leq j \leq D$.

Now Weyl's inequalities can be applied. The upper bound on $\vartheta_1(V)$ follows from Eq. (4.25) with $j = 2$ and $k = D$. Similarly, the upper bound on $\vartheta_{D+1}(V)$ follows with $j = k = D+1$. The lower bound $\vartheta_{D+1}(V) \geq \|\mu\|^2+1$ is a consequence of Eq. (4.26) with $j = 1$ and $k = D + 1$. This bound is sharpened by using the fact that every diagonal element of $V$ is a lower bound for the largest eigenvalue [Horn & Johnson 05, p181]).

The smallest eigenvalue of $V$ is hard to bound. The uncentered covariance can also be written as

$$V = \begin{pmatrix} \mathrm{diag}(\sigma^2) & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} \mu\mu^\top & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & \mu \\ \mu^\top & 0 \end{pmatrix} \,. \tag{4.29}$$

The eigenvalues of the right-most matrix are $\{-\|\mu\|, 0, \|\mu\|\}$. Applying Eq. (4.26) repeatedly, yields

$$\vartheta_1(V) \geq \min\left\{\sigma_1^2, 1\right\} + 0 - \|\mu\| \,. \tag{4.30}$$

Finally, the positive definiteness of $V$ and thus the bound $\vartheta_1(V) > 0$ follows elementary from

$$y \neq 0 \Rightarrow y^\top V y > 0 \,. \tag{4.31}$$

To prove this statement, write $y = (v, w)$ with $v \in \mathbf{R}^D$ and $w \in \mathbf{R}$. Then,

$$0 = y^\top V y = v^\top \mathrm{diag}(\sigma^2) v + (\mu^\top v + w)^2 \tag{4.32}$$

implies $v = 0$ due to the positive definiteness of $\mathrm{diag}(\sigma^2)$ and thus also $w = 0$. □

Analyzing the general case of correlated and unnormalized features is more difficult. The idea of the following theorem is to regard the off-diagonals as a perturbation of the diagonal matrix. This case can be analyzed with Geršgorin's circle theorem [Horn & Johnson 05, Theorem 6.1.1], which states that all eigenvalues lie in discs around the diagonal entries of the matrix.

**Theorem 4.5.** *Let $\mu, \sigma^2 \in \mathbf{R}^D$ denote the empirical mean and variance vector of $(x_1, \dots x_D)$ and $\Sigma \in \mathbf{R}^{D \times D}$ the covariance matrix. Assume that $\Sigma$ is positive definite and $\sigma_1^2 \leq \dots \leq \sigma_D^2$. Let*

$$R_i = \sum_{\substack{1 \leq j \leq D \\ j \neq i}} |\Sigma_{i,j}| \tag{4.33}$$

*denote the radius of the $i$-th Geršgorin circle. Then the extremal eigenvalues of the uncentered empirical covariance matrix are bounded by:*

$$\max\left\{0, \min\left\{1, \sigma_1^2\right\} - R_1 - \|\mu\|\right\} \leq \vartheta_1(V) \leq \sigma_1^2 + \mu_1^2 \tag{4.34}$$

$$\max\left\{\|\mu\|^2 + 1, \sigma_D^2 + \mu_D\right\} \leq \vartheta_{D+1}(V) \leq \sigma_D^2 + R_D + \|\mu\|^2 + 1 \,. \tag{4.35}$$

*Furthermore, the minimal eigenvalue $\vartheta_1(V)$ is strictly positive.*

*Proof of Theorem 4.5.* The upper bound on $\vartheta_1(V)$ follows from $\sigma_1^2 + \mu_1^2$ being a diagonal element of $V$. The other bounds obtained by Weyl's inequalities follow analogously to Theorem 4.3. In contrast to the case with uncorrelated features, the eigenvalues of $\Sigma$ are not known and are therefore bound using Geršgorin's theorem. To show the positive definiteness of $V$, note that to show the positive definiteness in Theorem 4.3, we have not used that the features are uncorrelated. □

The conclusions from Theorem 4.5 are the same as in the case of uncorrelated features. The lower bound for the condition number is almost unchanged, which shows that a non-zero mean and different scaling of features causes ill-conditioning. Further, with correlated features, the condition number can even be much worse. In particular, if the mean is zero and the variances are all one, Theorem 4.5 implies:

$$1 \leq \kappa(V) \leq \frac{2 + R_D}{\max\{0, 1 - R_1\}} \,. \tag{4.36}$$

Hence, ill-conditioning can also be caused from correlation of the features alone.

Our analysis shows that log-linear training can be ill-conditioned if the feature representation is fixed. Conversely, the analysis shows that log-linear training can be accelerated by decorrelating the features and normalizing their means and variances.

### 4.2.4 The case with regularization

How does the convergence analysis change when regularization is used? The Hessian of the $\ell_2$-regularization term is a multiple of the identity. Therefore, the eigenvalue $\alpha_{\text{reg}}$ of the regularization term adds to the eigenvalues of the loss term. This has an important consequence. The zero eigenvalue of the Hessian changes to $\alpha_{\text{reg}}$ and the Hessian is then strictly positive definite. Thus, the condition number depends only on the largest eigenvalue of $V$:

$$\kappa(\nabla^2 \mathcal{F}(\Lambda_0)) = \frac{C^{-1} \vartheta_{D+1}(V) + \alpha_{\text{reg}}}{\alpha_{\text{reg}}} \,. \tag{4.37}$$

**Table 4.1.** Results on the USPS task with different feature transformations and regularization parameters $\alpha_{\text{reg}}$

| **Feat. normalization** | $\alpha_{\text{reg}}N$ | **Train error [%]** | **#Iterations** |
|---|---|---|---|
| None | 0.0 | 0.0 | 513 |
| Mean and var. | 0.0 | 0.0 | 116 |
| Decor., mean and var. | 0.0 | 0.0 | 66 |
| None | 0.01 | 0.03 | 731 |
| | 0.1 | 0.43 | 358 |
| | 1.0 | 2.08 | 174 |
| | 10.0 | 4.29 | 100 |

This shows that for large regularization parameters, the condition number is close to one and convergence is fast. But if the regularization constant is small, the condition number gets large, even if $V$ is well-conditioned.

On first glance, it seems paradoxical that a small modification of the objective function can change the convergence behavior completely. But the objective function with a small regularization constant has a very flat optimum instead of being constant in these directions. Finding the exact optimum is indeed very hard.

On the other hand, the optimization is dominated by the loss-term in this case. If this term is well-conditioned, the optimization quickly approaches an optimum of the loss-term. Since the regularization term is only small, the iterates already correspond to good models according to the objective function. Only the final phase of convergence is slow.

## 4.3 Experimental Results

In this section, we validate our theoretical result on two classification tasks. We contrast a small-scale and a large-scale learning task in the sense of the characterization by [Bottou & Bousquet 08]. The small-scale task is the well-known U.S. Postal Service (USPS) task for handwritten digit recognition. The second task, IAM, is a continuous handwriting recognition task. Our main interest is the large-scale task IAM.

### 4.3.1 Handwritten digit recognition

The training set of the USPS dataset consists of 7 291 images from ten classes of handwritten digits, see Section A.3. We trained a log-linear classifier directly on the whole image with 16×16 pixels.

We used L-BFGS for optimization. For all experiments, we used a backtracking line search and a history size of ten, which is a standard value given in literature [Malouf 02,

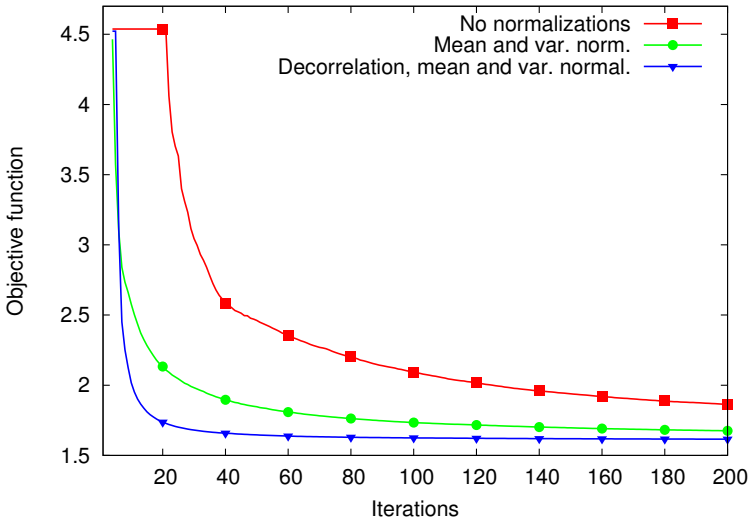


**Figure 4.2.** Training objective function on IAM with second-order features and different feature normalizations. The parameters are initialized with zero.

Sha & Pereira 03]. We initialized the models with zero and terminated the optimization, when the relative change in the objective was below $\epsilon =$ 1e-5 , i.e. when

$$\frac{\mathcal{F}(\Lambda_{i-1}) - \mathcal{F}(\Lambda_i)}{\mathcal{F}(\Lambda_i)} < \epsilon \; . \tag{4.38}$$

Table 4.1 contains the results on the USPS task. The results reflect our analysis of the condition number. Without normalizing mean and variance, the optimization problem is not well-conditioned. It requires more than 500 iterations until the termination criterion is satisfied. The optimization takes even longer, when a very small non-zero regularization constant is used. This is what we expected by analyzing the condition number – the objective function has a very flat optimum, which slows down convergence. On the other hand, for higher regularization parameters, the convergence speed is much faster.

We applied the normalizations only to the unregularized models because the feature transformations affect the regularization term. Therefore, results with regularization are not exactly comparable when feature transformations are applied. The mean and variance normalization reduces the computational costs greatly, from 513 to 116 iterations. Decorrelation reduces the number of iterations further to 66.

### 4.3.2 Handwritten text recognition

Our second task is the IAM English handwriting recognition task, which is described in Appendix A.4. The IAM task is a *large-scale learning problem* in the sense that it is not feasible to run the optimization until convergence as on USPS, thus the test error is strongly influenced by the optimization accuracy. The corpus has a predefined subdivision into a training, a development, and a test set. The training set contains lines of handwritten text with 54k words in total. With our feature extraction, this corresponds to roughly 3.6 million observations. The development and test set contain 9k respectively 25k words. We use the IAM 2011 system described in A.4.1 as our GHMM baseline. With maximum likelihood training, the baseline system achieves an error rate of 39.4 percent on the test data. MPE training improves the result to 29.3 percent word error rate.

We built a hybrid log-linear HMM handwriting recognition system, analogous to the log-linear speech recognition systems from Chapter 3. The log-linear model has been trained according to the cross-entropy criterion using the alignment generated with the maximum likelihood baseline system. The objective function has been optimized with L-BFGS with a history size of ten. For all experiments, we assumed a limited training budget, which allows for performing 200 iterations. We used polynomial features of degree one ($D = 30$), two ($D = 495$) and three ($D = 5\,455$). In contrast to USPS, where the training frame error without regularization was zero, the training frame error on IAM ranges from forty to sixty percent. Thus, the impact of regularization is only minor. In preliminary experiments, we did not obtain any improvements by regularization. Therefore, all reported results are obtained without regularization.

Figure 4.2 shows the optimization behavior of the second-order models with different feature preprocessings. The observed convergence behavior is again in accordance to our theoretical analysis. Note that the objective function of the training with unnormalized features remains constant for the first iterations, because L-BFGS performs a backtracking.

The recognition results are shown in Table 4.2. The first-order features are already decorrelated, but without normalizing their mean and variance, the convergence is slower, resulting in a worse word error rate on the development and test set. The difference is moderate, when the parameters are initialized with zero. In a next experiment, we initialized all parameters randomly with plus or minus one. This results in a huge degradation with the unnormalized features and has only a minor impact with normalized features and exactly the same random initialization.

The differences are even larger in the second-order feature experiments. This can be expected, since mean and variance take on more extreme values when the features are squared. In addition, the second-order features are correlated, which further worsens the condition of the problem. The improvement from mean and variance normalization is only moderate in word error rate with zero initialization. With unnormalized features and random initialization, the optimization did not lead to a usable model for recognition at all. Fastest convergence speed and best recognition results were obtained by

**Table 4.2.** Results on the IAM database for polynomial feature spaces of degree $m \in \{1, 2, 3\}$ with different initializations and preprocessings.

| $m$ | **Feature Normalization** | **Initialization** | **WER** [%] Development | Test |
|---|---|---|---|---|
| 1 | None | Random | 68.3 | 75.5 |
| | | Zero | 49.9 | 60.1 |
| | Mean and var. | Random | 48.9 | 58.5 |
| | | Zero | 49.7 | 58.9 |
| 2 | None | Random | >100.0 | >100.0 |
| | | Zero | 32.4 | 40.2 |
| | Mean and var. | Random | 34.4 | 41.3 |
| | | Zero | 30.2 | 38.5 |
| | | 1st-order | 26.8 | 33.1 |
| | Decorrelation, mean and var. | Random | 25.9 | 32.3 |
| | | Zero | 25.1 | 31.6 |
| 3 | Mean and var. | 2nd-order | 23.0 | 27.4 |

additionally decorrelating the features. In addition, the influence of the initialization is the smallest in this case.

The estimation of the whitening transformation for third-order features is already computationally very expensive because of the high feature dimension. Therefore, we only normalized mean and variance of the third-order features, but initialized the models incrementally from first to second to third-order features. In this manner, we obtain our best result of 27.4 percent word error rate. Thus, the log-linear system outperforms the MPE trained baseline system.

Although the main purpose of our experiments is to validate our theoretical convergence analysis, it is of interest to compare our result to others reported in literature. Our system is competitive to other systems with comparable preprocessing and architecture. [Bertolami & Bunke 08] obtain 32.9 percent word error rate on the test set with an ensemble-based HMM approach. [Dreuw & Heigold$^+$ 11] obtain 29.0 percent word error rate with an additional discriminative adaptation method. The system of [Graves & Liwicki$^+$ 09], which is based on recurrent neural networks with a long short-term memory (LSTM) structure, outperforms our system with 25.9 percent word error rate.

More recently, strong improvements have been obtained on the IAM database. Currently, the best published result on IAM is 12.2 percent word error rate [Doetsch & Kozielski$^+$ 14], which is also based on LSTM-RNNs. The largest part of the gains are due to a better preprocessing, an open vocabulary recognition approach, and other improvements, which are not related to the emission model [Kozielski & Doetsch$^+$ 13], cf.

also the system description in Appendix A.4.2.

## 4.4 Relation to Prior Work

The convergence behavior of log-linear training has been studied empirically in several works. Most of them focused on the empirical comparison of optimization algorithms [Minka 01, Malouf 02, Wallach 03, Sha & Pereira 03], confirming Minka's observation that bound optimization algorithms such as GIS and IIS converge extremely slowly in comparison to gradient-based numerical optimizers. Minka also observed that a non-zero feature mean is generally harmful for log-linear training.

The theoretical analysis of the optimization problem is limited. In [Salakhutdinov & Roweis$^+$ 03], a general convergence analysis of bound optimization algorithms is derived. As a special case, Salakhutdinov et al. discuss the application of GIS to log-linear training and conclude that GIS converges slowly when features are correlated and have a non-zero mean. The disadvantage of their analysis is that it concerns only GIS which is today known to perform very badly in practice.

Our analysis shares several similarities with the study by LeCun et al. [LeCun & Kanter$^+$ 90, LeCun & Kanter$^+$ 91], which has been derived in the context of neural networks. The most important difference to our study is that LeCun et al. consider a different combination of model and training criterion. LeCun et al. study a linear regression model with a one-dimensional output trained according to the mean squared error criterion. The model does not have a bias parameter. In this simpler case, the cost function is quadratic and its Hessian matrix is the uncentered covariance matrix. The properties of the uncentered covariance matrix are analyzed under quite restrictive assumptions. It is assumed that all feature components are independent and identically distributed. Thus, the effect of correlation and different feature scales is not addressed. Furthermore, they fix the ratio of the number of model parameters and the number of training examples. The derivation of the spectrum is then performed in the limit of infinite training data, leading to a continuous spectrum. In contrast, we assume a fixed training set. In [LeCun & Bottou$^+$ 98], informal arguments are given, which suggest that the features should be scaled to have the same variance and decorrelated. Thus, we confirm the findings of LeCun et al. for a different training criterion and under more general assumptions.

## 4.5 Discussion

The convexity of training guarantees that a large class of well-understood numerical optimization algorithms converge to the global optimum of the objective function. For large-scale applications like speech recognition, considerations in the limit of infinite iterations of the optimization algorithm are not sufficient. Assuming a limited training budget, the choice of the optimization algorithm and the initialization determine the quality of the approximation of the global optimum. This rather obvious fact becomes

critical when the optimization problem is not well-posed. The analysis derived in this chapter specifies the conditioning of the optimization problem in terms of elementary feature statistics. With unfavorable feature properties, log-linear training can be highly ill-conditioned and the test error is strongly influenced by the optimization error. Conversely, our analysis shows that the optimization problem can be preconditioned by feature transformations.

A rather unexpected result of our analysis is the effect of the regularization term. We showed that a weak regularization can hurt the performance of the optimizer. A small regularization constant causes the objective function to have a very flat optimum instead of being constant in one direction of the parameter space. This can be regarded as a disadvantage of $\ell_2$-regularization for log-linear models. The regularization term should not penalize models differently if they induce the same posterior probabilities.

The limitation of our analysis is the assumption that the second-order properties of the objective function do not change strongly when deviating from the zero-initialization. Although our experiments justify this assumption, it would be desirable to formulate conditions under which it is valid.

From the difficulty of training log-linear models, we can draw a general conclusion. On the one hand, convexity is highly desirable, because it guarantees that training can not get stuck in local optima and it facilitates theoretical analysis. On the other hand, on large-scale tasks, it is inevitable to deal with the difficulties of optimization. Influenced by the dominance of convex approaches in the machine learning community, one aim of convex log-linear modeling was to get rid of the dependence on the initialization and the optimization algorithm and thus to make training easier to use. This initial hope, which has been formulated in our work [Wiesler & Nußbaum$^+$ 09] and others [Heigold 10] appears to be overly optimistic.

While our analysis emphasizes the role of optimization in the convex setting, it is relevant beyond of that. Nowadays, neural networks are commonly used with a log-linear output layer[1] and trained according to the cross-entropy criterion. Our convergence analysis therefore describes the theoretically amenable case of the most common type of neural network training.

## 4.6 Publications and Joint Work

This chapter is based on the analysis derived in [Wiesler & Ney 11], for which the author contributed the idea, derived the theory, and performed the experiments. The baseline systems for the handwriting recognition experiments were provided by P. Dreuw.

[1] in the context of neural networks known as softmax layer

# Chapter 5

# Hessian-Free Optimization for Cross-Entropy Training

Deep neural network-based acoustic models have been shown to be superior to Gaussian mixture models in a number of recent works. The de-facto standard algorithm for the optimization of neural networks is stochastic gradient descent. Its main advantage is that it scales well to large datasets. However, its asymptotic convergence behavior is very slow. In regions with pathological curvature, stochastic gradient descent may almost stagnate and thereby falsely indicate convergence. Another drawback of stochastic gradient descent is that it is hard to parallelize across mini-batches.

In this chapter, we study a second-order optimization algorithm known as Hessian-free (HF). This algorithm has been proposed originally in [Martens 10], where it has been evaluated on small-scale tasks such as MNIST. The focus of Martens' work is on the role of pre-training for deep neural networks. In a recent work [Kingsbury & Sainath$^+$ 12], Hessian-free optimization has been applied to large-scale sequence-discriminative training of neural networks for speech recognition. Kingsbury et al.'s main motivation is the observation that Hessian-free can be parallelized more easily than stochastic gradient descent.

The goal of this chapter is to better analyze the Hessian-free algorithm and to investigate whether it is useful for large-scale cross-entropy training of deep neural networks.

## 5.1 Introduction

Hessian-free is a second-order batch optimization algorithm, which has been designed specifically for optimizing neural networks. It is based on a full second-order model of the objective function, which is computed in every iteration. This distinguishes it from other second-order algorithms. For example, L-BFGS uses only a low-rank model, and Rprop a diagonal model, cf. Section 3.5.

As a batch algorithm, Hessian-free can be parallelized well. In the work by [Kingsbury & Sainath$^+$ 12], a speedup over SGD in terms of wall-clock time by a factor of more than five has been obtained by parallelizing Hessian-free. Furthermore, Hessian-free allows for a more accurate optimization than SGD. In the work by Kingsbury et al., this reduced word error rate by four percent relative.

From an optimization point of view, the characteristics of frame- and sequence-discriminative training are very different. In particular, sequence-discriminative training is initialized with a good model, while frame-discriminative training is typically started from scratch. Therefore, it is not clear whether Kingsbury et al.'s results generalize to cross-entropy training.

Our setting is the same as in the previous chapter. The training sample is a sequence of observations and classes $(x_n, c_n)_{n=1,\ldots,N}$. In training, the cross-entropy objective function

$$\mathcal{F}(\theta) = -\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(c_n|x_n) \tag{5.1}$$

is minimized. In contrast to Chapter 3 and 4, the posterior probability is given by a neural network with parameters $\theta$. The output layer of the network is assumed to be a log-linear model, also known as a softmax layer.

## 5.2 Martens' Hessian-Free Algorithm

This section describes Martens' Hessian-free algorithm and its relation to other algorithms in optimization literature.

The algorithm as it has been implemented in this work is outlined in Algorithm 1. It is a specific form of the Newton-CG method. Recall from Section 3.5.1 that the Newton step is defined as the minimum of the quadratic model of the objective function

$$m(P) = \frac{1}{2}P^\top M P + \nabla\mathcal{F}(\theta)^\top P + \mathcal{F}(\theta)\ . \tag{5.2}$$

The matrix $M$ is chosen as the Hessian matrix $\nabla^2\mathcal{F}(\theta)$. If $M$ is positive-definite, the Newton step is well-defined and can be computed by solving the linear system

$$M\,P = -\nabla\mathcal{F}(\theta)\ . \tag{5.3}$$

In every iteration of Newton's method, the model is updated by the solution of this equation. Newton's method converges rapidly in terms of iterations. For high-dimensional problems like the training of neural networks, Newton's method is prohibitive, because the size of the Hessian matrix is in the order of the square of the model size. Accumulating, storing, as well as inverting the Hessian matrix is not feasible.

The Newton-CG method is a well-known modification of Newton's method for high-dimensional problems, see e.g. [Nocedal & Wright 06, Chapter 6]. In the Newton-CG method, the linear system (5.3) is solved approximately using the iterative conjugate gradient (CG) algorithm [Hestenes & Stiefel 52]. Conjugate gradient only requires products of the Hessian matrix and an arbitrary vector, but does not require the matrix itself, cf. Algorithm 2. For problems, where an algorithm exists which computes these matrix-vector products efficiently, Newton-CG can be applied.

**Algorithm 1** Outline of the Hessian-free algorithm

**function** HESSIANFREE($\theta_0$, $\lambda_0$)
  $\theta \leftarrow \theta_0$ ▷ network parameters
  $\lambda \leftarrow \lambda_0$ ▷ damping factor
  $\mathcal{F}_{\text{prev}} \leftarrow \mathcal{F}(\theta_0)$ ▷ objective function
  $P_0 \leftarrow 0$ ▷ search direction
  **while** not converged **do**
    $A = G(\theta) + \lambda I$ ▷ damped Gauss-Newton matrix
    $b = -\nabla\mathcal{F}(\theta)^\top b$
    ▷ apply CG algorithm to $q_\theta(P) = \frac{1}{2}P^\top(G(\theta) + \lambda I)P + \nabla\mathcal{F}(\theta)^\top P$
    $\{P_1, \ldots, P_N\} = \text{CG}(A, b, P_0)$
    determine first $i$ with $\mathcal{F}(\theta + P_{i-1}) \geq \mathcal{F}(\theta + P_i)$ ▷ CG backtracking
    $\gamma \leftarrow 1$
    **if** $\mathcal{F}_{\text{prev}} < \mathcal{F}(\theta + P_i)$ **then**
      find $\gamma > 0$ with $\mathcal{F}_{\text{prev}} > \mathcal{F}(\theta + \gamma P_i)$ ▷ linesearch
    **end if**
    $\rho \leftarrow (\mathcal{F}_{\text{prev}} - \mathcal{F}(\theta + P_N))/q_\theta(P_N)$ ▷ damping factor update: our version
    **if** $\rho < 0.25$ **then**
      $\lambda \leftarrow 3/2\lambda$
    **else if** $\rho > 0.75$ **then**
      $\lambda \leftarrow 2/3\lambda$
    **end if**
    $\theta \leftarrow \theta + \gamma P_i$, $P_0 \leftarrow \beta P_N$, $\mathcal{F}_{\text{prev}} \leftarrow \mathcal{F}(\theta + \gamma P_i)$ ▷ update
  **end while**
  **return** $\theta$
**end function**

The most important design choice for the application of the Newton-CG algorithm is how to compute the required matrix-vector products. For neural networks, Hessian-vector products can be computed by a modified forward-backward algorithm similar to the one used for computing the gradient. In neural network literature, this is known as the *Pearlmutter trick* [Pearlmutter 94].

Since the Hessian matrix of the cross-entropy criterion is not positive definite, Martens decided to use the positive semi-definite Gauss-Newton matrix instead. The Gauss-Newton matrix has originally been defined for the squared error criterion, but can be generalized to the cross-entropy criterion [Schraudolph 02]. Algorithm 3 computes the Gauss-Newton matrix-vector products by performing a modified forward pass, which sets the error signal of the output layer. The matrix-vector product is then obtained by applying the conventional backward pass.

The cost of a Gauss-Newton matrix-vector product is roughly twice the cost of a gradient calculation. Calculating these products on the complete training data is highly expensive, since every run of conjugate gradient requires several of such products. A

**Algorithm 2** Conjugate gradient algorithm for solving the system of linear equations $Ax = b$ with initialization $x_0$

**function** CG($A$, $b$, $x_0$)
  $r_0 \leftarrow Ax_0 - b$ ▷ residual
  $p_0 \leftarrow -r_0$ ▷ search direction
  $k \leftarrow 0$
  **while** $r_k \neq 0$ **do** ▷ iterate
    $\alpha_k \leftarrow \frac{r_k^\top r_k}{p_k^\top A p_k}$
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $r_{k+1} \leftarrow r_k + \alpha_k A p_k$ ▷ matrix-vector product
    $\beta_{k+1} \leftarrow \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}$
    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$
    $k \leftarrow k + 1$
  **end while**
  **return** $x_k$
**end function**

central idea of Martens is to use a stochastic approximation of the Gauss-Newton matrix, i.e. to compute matrix-vector products on mini-batches. In contrast, gradients and objective function values are computed exactly.

The quadratic approximation (5.2) is only reliable in a neighborhood around $\theta$. Therefore, the Newton-CG method must be prevented from taking too large steps. This is in particular important when using a stochastic approximation of $M$. In optimization literature, it is suggested either to perform a line-search in the direction $P$, or, to assume a trust region around $\theta$. This means that instead of minimizing (5.2), a constrained minimization problem is solved:

$$P_{\text{TR}} = \min_P m(P) \quad \text{such that} \quad \|P\|_2 \leq \Delta \,. \tag{5.4}$$

The trust region radius $\Delta$ is dynamically adapted based on the progress of the algorithm. Problem (5.4) can be solved with matrix factorizations [Moré & Sorensen 83], but these are prohibitive for high-dimensional problems. A widely used cheap approximation of (5.4) is given by Steihaug's method [Steihaug 83], which simply terminates CG when the trust region is left. Martens found the well-understood Steihaug method not to be effective for neural network training and therefore uses a *damping heuristic* instead. This means that the matrix $M$ in Eq. (5.2) is replaced by $M + \lambda I$ with a damping parameter $\lambda > 0$. Here, $I$ denotes the identity matrix. Similar to trust region algorithms, the damping parameter $\lambda$ is adapted based on the ratio of the actual and the expected improvement

$$\rho = \frac{\mathcal{F}(\theta) - \mathcal{F}(\theta + P)}{m(0) - m(P)} \,. \tag{5.5}$$

For large values of $\rho$, the damping parameter is decreased, which results in larger

**Algorithm 3** Algorithm for computing the product of the Gauss-Newton matrix of a network with parameters $\theta = (W^{(1)}, \ldots, W^{(L)}, b^{(1)}, \ldots, b^{(L)})$ with a parameter vector $\vartheta = (P^{(1)}, \ldots, P^{(L)}, a^{(1)}, \ldots, a^{(L)})$ for one observation $x$ [Schraudolph 02].

**function** GNPRODUCT($\theta, \vartheta, x$)
  $x^{(0)} \leftarrow x$ ▷ standard forward-pass
  **for** $l = 1, \ldots, L-1$ **do**
    $x^{(l)} \leftarrow {W^{(l)}}^\top x^{(l-1)} + b^{(l)}$
    $x^{(l)} \leftarrow \sigma(x^{(l)})$ ▷ sigmoid layer
  **end for**
  $x^{(L)} \leftarrow {W^{(L)}}^\top x^{(L-1)} + b^{(L)}$
  $x^{(L)} \leftarrow \text{softmax}(x^{(L)})$ ▷ softmax layer
  $y^{(0)} \leftarrow 0$ ▷ modified forward-pass
  **for** $l = 1, \ldots, L-1$ **do**
    $y^{(l)} \leftarrow {W^{(l)}}^\top y^{(l-1)} + {P^{(l)}}^\top x^{(l-1)} + a^{(l)}$
    $y^{(l)} \leftarrow \sigma(x^{(l)})(1 - \sigma(x^{(l)})) \odot y^{(l)}$ ▷ $\odot$ denotes element-wise multiplication
  **end for**
  $y^{(L)} \leftarrow {W^{(L)}}^\top y^{(L-1)} + {P^{(L)}}^\top x^{(L-1)} + a^{(L)}$
  $\epsilon \leftarrow \left(\text{diag}(x^{(L)}) - x^{(L)} {x^{(L)}}^\top\right) y^{(L)}$ ▷ modified error signal for CE criterion
  **return** backwardPass($\theta, (x^{(1)}, \ldots, x^{(L)}), \epsilon$)
**end function**

steps. For small $\rho$, the damping parameter is increased. In principle, a damped Gauss-Newton step is equivalent to a trust region Gauss-Newton step. The heuristic lies in the update of the damping parameter, because it is not directly related to a trust region radius. Furthermore, the update of the damping parameter is only consistent if conjugate gradient is run until convergence and $\rho$ is evaluated with its final iterate. The advantage of the damping approach is that it is cheaper to solve for the damped update than solving Eq. (5.4), and it avoids the approximation of Steihaug's algorithm.

An important property of Hessian-free is that conjugate gradient is initialized with the solution of the previous conjugate gradient run multiplied with a constant scalar. This is in contrast to most other Newton-CG implementations, where conjugate gradient is initialized with zero. Martens reported that this initialization speeds up Hessian-free by "an order of magnitude" [Martens 10]. Indeed, a suitable initialization may save some iterations. On the other hand, a non-zero initialization may have a strongly negative effect when a hard-limit on the number of iterations is set. In addition, Steihaug's algorithm and related approaches require a zero initialization.

Another possibility for speeding up conjugate gradient is the use of a preconditioner. Martens suggests the use of a diagonal preconditioner based on the accumulated squared gradient.

Finally, Martens introduced a backtracking along intermediate results of the conjugate gradient evaluation. Backtracking is helpful if there is a strong mismatch between

**Table 5.1.** Results with the HF algorithm on MNIST

| Algorithm | CG init. | Prec. | Damping upd. | #Epochs | #CG iters |
|---|---|---|---|---|---|
| Gradient descent | - | - | - | - (59.8%) | - |
| Steihaug | Zero | No | | - (0.1%) | 2 810 |
| Hessian-free | Zero | No | Backtracking | 237 | 55 532 |
| | | | Final | 79 | 2 308 |
| | Prev. sol. | No | Backtracking | 56 | 1 443 |
| | | | Final | 60 | 1 377 |
| | | Yes | Backtracking | 63 | 1 505 |
| | | | Final | 56 | 1 458 |

the quadratic model and the objective function. In this case, the intermediate conjugate gradient results can lead to much better updates than the final result. On the other hand, the backtracking procedure causes an objective function evaluation for every tested model and is therefore highly expensive.

When using backtracking, it is not clear which model should be used for calculating the damping factor in Eq. (5.5). Martens uses the best model obtained from the backtracking [Martens 10]. However, this counteracts the damping approach. It means that the damping factor may be decreased although the solution of the conjugate gradient algorithm corresponds to a much too large step. This is undesirable, because it causes many backtrackings in the subsequent steps. Furthermore, the linear system Eq. (5.3) gets ill-conditioned with small damping parameters and therefore conjugate gradient converges very slowly. In the next section, we compare Martens' variant with using the final iterate for updating the damping parameter.

## 5.3 Empirical Analysis on Handwritten Digit Recognition

In this section, we analyze the consequences of Martens' design choices empirically on the MNIST database, a small-scale handwritten digit recognition task, see Appendix A.3. The task of the MNIST dataset is the classification of images of handwritten digits. The dataset consists of 60 000 training samples and 10 000 test samples.

The implementation of the algorithm has been realized in the Microsoft DNN training tool and makes use of GPU acceleration. As in the conventional SGD training [Seide & Li$^+$ 11b], all computations are performed on the GPU without synchronization to the main memory. The computation of the gradients and likelihoods can be parallelized to multiple GPUs or a large number of CPUs straight-forwardly, but we simply used one GPU in this work.

For the experiments, we used the same setup as in [Martens 10], i.e. the features fed into the neural networks were the gray values of the images with a resolution of $28 \times 28$

pixels. The DNNs had four layers with dimensions 1 000, 500, 250, and 30.

We trained a baseline system with SGD with a momentum term. It turns out that with this network architecture, zero training error can be achieved after 420 epochs. By using early stopping, we obtained similar results on the test set with all algorithms that can achieve zero classification error on the training data. Since the goal here is to improve the optimizer, we focus on the performance on the training data. As a measure of convergence speed, we compare the number of epochs that are required for separating the training data. For all experiments, we set the maximal number of epochs to 500. For Hessian-free, we used the settings recommended by Martens, in particular we used a maximum of 250 conjugate gradient iterations and ten percent of the training data for computing the matrix-vector products.

In a first test, we compared the batch algorithm gradient descent with a basic Hessian-free implementation, in order to verify that the second-order information accelerates the optimization. The basic Hessian-free implementation does not use preconditioning and always initializes conjugate gradient with zero. The results in Table 5.1 show that batch gradient descent is not useful for optimizing deep networks. Even after 500 epochs, the training error is still 59.8 percent. In contrast, the basic Hessian-free implementation separates the training data after 237 epochs.

In addition, we tested Steihaug's method. As Martens, we observed that it is not effective for optimizing deep networks. The reason is that the conjugate gradient iterates move out of the trust region after only very few iterations. Therefore, Steihaug's method does not fully exploit the second-order information.

The computation time per epoch of the basic Hessian-free implementation is very high, because a large number of conjugate gradient iterations and backtrackings is performed. By using the final iterate instead of the backtracking iterate for computing the damping update, Hessian-free's performance is greatly improved. The number of epochs is reduced to 79 and the number of conjugate gradient iterations from 55 532 to 2 308. This clearly shows that the correct update of the damping parameter is crucial and it is preferable to use the final iterate for calculating the damping parameter.

Initializing conjugate gradient with the previous solution instead of zero has two effects. First, the non-zero initialization reduces the average number of conjugate gradient iterations per epoch. For the variant where the backtracking iterate is used for calculating $\rho$, the damping parameters are very small and therefore the resulting linear problems are poorly conditioned. In this case, the average number of conjugate gradient iterations per epoch is strongly decreased by a factor of 10.2. When the final iterate is chosen for updating $\rho$, the number of conjugate gradient iterations is only reduced by a factor of 1.3.

Second, the number of epochs is reduced. The intermediate results which are used for backtracking are between the initialization and the exact solution of the linear system. Including the previous solution in the search direction weakens the effect of unrepresentative mini-batches, similar to the momentum term in stochastic algorithms.

Interestingly, we could not observe an acceleration by using preconditioning. From our results, we can not recommend the use of the squared-gradient preconditioner pro-

posed in [Martens 10].

An obvious way for accelerating Hessian-free is to compute the likelihoods for the backtracking and the damping parameter update on a small subset of the training data or as in [Kingsbury & Sainath$^+$ 12] on the validation set. However, we found that this approach does not work at all when training models from scratch. The reason is that small improvements on the training data do not necessarily carry over to the smaller dataset. This causes the algorithm to increase the damping parameter repeatedly and thereby testing smaller and smaller steps. Thus, the algorithm stagnates although it has not reached an optimum. This problem did not occur in [Kingsbury & Sainath$^+$ 12], because they used a very good initialization.

In comparison to SGD, Hessian-free requires less epochs until convergence. Since the computation time of Hessian-free per epoch is much higher, the total computation time of SGD and Hessian-free is comparable. On large-scale tasks, Hessian-free can be accelerated by parallelization. On the other hand, the stochastic approximation of SGD is much more beneficial on large-scale task. In terms of test set accuracy, Hessian-free does not perform better than SGD, because underfitting is not an issue on this task. This situation is different on large-scale tasks.

## 5.4 Experimental Results on Speech Recognition

In addition to the small-scale experiments on MNIST, we performed experiments on SMDR3, a Microsoft-internal short message dictation task. The training set consists of 63.4 hours of audio data. The dev and test set have 16 028 and 22 809 running words respectively.

The baseline acoustic model is a neural network with five hidden layers, each with 1 024 nodes with sigmoid activation. The input to the network is a 572-dimensional vector obtained by concatenating MFCC features with up to third-order derivatives in a window of 11 frames. The neural network is initialized with layerwise RBM pretraining. The fine-tuning is performed with SGD with a momentum term. The initial SGD learning rate is set to 0.08 and later reduced to 0.002. After 25 epochs, SGD seems to stagnate with a training frame error rate of 33.2 percent. The word error rate of the corresponding model is 23.8 percent on the development set and 22.9 percent on the test set.

We used the best Hessian-free configuration from the MNIST task, i.e. we updated the damping parameter using the final conjugate gradient iterate, initialized conjugate gradient with the previous solution, and did not use preconditioning. Because of the large size of the training set, we used only one percent of the training data for computing the matrix-vector products.

In an initial experiment, we compared Hessian-free with SGD and found that Hessian-free converges orders of magnitude slower than SGD on this large-scale task. Even considering the use of parallelization, it was not reasonable to train the DNNs from scratch with Hessian-free. Therefore, we did not investigate this possibility further.

**Table 5.2.** Results on the short message dictation task. The third column denotes the number of initial SGD epochs. The fourth column denotes the number of additional epochs with either SGD or HF. The fifth column lists the frame error rate on the training data, and the sixth column the word error rate on the development set.

| **Model** | **Algorithm** | **Epoch** Init | Add. | **FER train.** [%] | **WER dev** [%] |
|---|---|---|---|---|---|
| 5×1024 | SGD | 25 | | 33.2 | 23.8 |
| | | | +25 | 32.3 | **23.4** |
| | | | +75 | 30.7 | 23.7 |
| | HF | | +25 | 27.1 | 23.4 |
| | | | +45 | 24.8 | **23.3** |
| | | | +75 | 23.0 | 23.8 |
| 5×512 | SGD | 10 | | 37.9 | 26.1 |
| | | | +40 | 36.6 | **25.5** |
| | HF | | +10 | 35.3 | 25.2 |
| | | | +40 | 33.8 | **24.3** |
| 4×1024 | SGD | 10 | | 34.2 | 24.5 |
| | | | +40 | 32.7 | **24.2** |
| | HF | | +10 | 30.5 | 23.7 |
| | | | +30 | 28.2 | **23.4** |
| | | | +40 | 27.5 | 23.8 |

In the following experiments, we aimed at improving the SGD result using Hessian-free. This may be successful, because SGD does not evaluate the gradient on the complete training data. Therefore, it does not allow for detecting whether a local optimum of the objective function has been found. Unexpectedly, it turned out that the SGD parameters were not close to an optimum at all. After 75 additional Hessian-free iterations, the training frame error rate decreased from 33.2 to 23.0 percent, see the results in Table 5.2. Knowing the Hessian-free result, a natural question is whether the training error rate can be decreased with SGD as well. Indeed, after 75 additional SGD epochs, the frame error rate slowly decreased to 30.7 percent, but it did not reach the frame error rate of Hessian-free.

However, these large gains on the training set did not carry over to the development set. The best Hessian-free recognition result which we achieved was 23.3 percent word error rate, corresponding to a model with 24.8 percent frame error rate. This is only a tiny improvement of 0.1 percent word error rate over the SGD result. Decreasing the error rate on the training set further leads to overfitting. The behavior on the test set

is the same as on the development set. The SGD result is improved by 0.1 percent absolute as well.

In the following experiments, we investigated whether smaller models can be improved using Hessian-free, where overfitting is less an issue. This would be of great practical value, because reducing the decoding costs of hybrid DNN-HMM speech recognizers is a challenge for their large-scale application. In these experiments, we initialized Hessian-free with the result of SGD after ten epochs in order to reduce computation time. Using only half of the nodes per layer, the best SGD result is 25.5 percent word error rate, 2.1 percent worse than with the large model. This gap is reduced to only 0.9 percent with Hessian-free. This deep and narrow model does not overfit, instead some accuracy is lost due to underfitting. Another experiment is on a neural network with only four layers of 1 024 nodes. Here, the best SGD result is 24.2 percent word error rate. Interestingly, the best error rate with Hessian-free is again 23.4 percent, i.e. we achieved the same error rate with a four-layer network as with the five-layer network baseline.

## 5.5 Discussion

In this chapter, we empirically analyzed the properties of the Hessian-free algorithm, a second-order batch optimization algorithm, which has been proposed in [Martens 10]. Some techniques which are used in the algorithm are rather heuristic and their effect is not well understood. In particular, we observed in experiments on MNIST that the damping heuristic has a critical impact on the performance of the algorithm. Furthermore, the experiments show that the preconditioner proposed in the original Hessian-free paper [Martens 10] is not effective.

We observed that Hessian-free is not efficient for training deep neural networks from scratch in large-scale applications. But when starting from a reasonable initialization, making use of second-order information is highly beneficial. Since the Hessian-free algorithm can be parallelized well, one could perform only few epochs with SGD, and then continue with Hessian-free. We obtained strong improvements in terms of training error using Hessian-free. However, these improvements did not carry over to unseen data due to overfitting. These results clearly show the need for good regularization methods when improved optimizers are used. This topic is addressed in the next chapter.

Our experiments allow for drawing some general conclusions about neural network training. The first is about the role of local optima in neural network training. When SGD stagnates, it is often argued that a local optimum has been reached. However, our experiments show that the slow asymptotic convergence behavior of SGD can falsely indicate convergence. In our experiments, the objective function could still be improved strongly by continuing the optimization with a second-order algorithm.

Second, we could reduce size of the model without losing accuracy with Hessian-free optimization. Or conversely, the use of larger models is a possibility to overcome the limited optimization performance of SGD. From a practical point of view, the use of

more accurate optimization algorithms is interesting, because model size needs to be restricted in production systems.

## 5.6 Publications and Joint Work

The work described in this chapter has been performed during an internship at Microsoft Research, which resulted in the publication [Wiesler & Li$^+$ 13]. The idea for applying the Hessian-free algorithm to cross-entropy training of acoustic models, the implementation in the Microsoft DNN training tool, the experiments on MNIST, and the analysis of the results were contributed by the author. The speech recognition experiments were performed jointly with J. Xue. The work was supervised by J. Li and J. Xue.

# Chapter 6

# Mean-Normalized Stochastic Gradient Descent

In the previous chapter, we found that neural networks can be optimized more accurately using the Hessian-free algorithm than with stochastic gradient descent. Although this is of theoretical interest, we encountered two problems with Hessian-free. First, its updates are too costly for training deep neural networks from scratch. Second, with large networks, the improvements in terms of training accuracy did not generalize to unseen data.

In this chapter, we develop a simple and effective stochastic second-order algorithm [Wiesler & Richard$^+$ 14b], which is motivated by our convergence analysis of log-linear models in Chapter 4. It comes with only negligible additional costs over stochastic gradient descent and does not require further tuning of hyperparameters. Further, we study a linear bottleneck network structure, which we find to be an excellent means against overfitting. Experimental results are presented for a continuous handwriting recognition task and a conversational speech recognition task.

## 6.1 Introduction

An alternative to SGD is the use of batch algorithms, i.e. algorithms where the gradient is computed on the full dataset, for example L-BFGS [Liu & Nocedal 89, Dean & Corrado$^+$ 12], Rprop [Riedmiller & Braun 93], or the Hessian-free algorithm discussed in the previous chapter. These algorithms have in common that they make efficient use of second-order information. Further, they can be parallelized well by distributing the gradient computation across a large number of machines. However, the computational costs of batch algorithms may become prohibitive on large datasets. Therefore it is interesting to incorporate second-order techniques into stochastic algorithms.

Recall from Chapter 3 that the general form of a stochastic second-order update rule is

$$\theta_{i+1} = \theta_i - \eta_i B_i \nabla \mathcal{F}(\theta_i, \mathcal{B}) \,, \tag{6.1}$$

where $\eta_i > 0$ is the learning rate, $\nabla \mathcal{F}(\theta_i, \mathcal{B})$ is the gradient of the objective function $\mathcal{F}$ on a mini-batch $\mathcal{B}$, and the positive-definite matrix $B_i$ is chosen to approximate the inverse Hessian of $\mathcal{F}$ at $\theta_i$.

Most second-order optimization algorithms employ strong approximations to the Hessian matrix on the mini-batch, e.g. a diagonal approximation [LeCun & Bottou$^+$ 98] or a quasi-Newton approximation [Schraudolph & Yu$^+$ 07, Bordes & Bottou$^+$ 09], cf. Chapter 3. A difficulty of these approaches is that the stochastic second-order information is very noisy and the algorithms have additional computational costs. The idea of our proposed algorithm is to make use of the analytic results from Chapter 4 about the structure of the objective function. This allows for estimating a cheap and reliable second-order model from the limited data on mini-batch level.

It has often been observed that improved optimization algorithms may cause overfitting on machine learning tasks, cf. Chapter 5. Here, we investigate a recently proposed bottleneck network structure [Sainath & Kingsbury$^+$ 13, Xue & Li$^+$ 13] that allows for reducing model size and thus improves generalization. Furthermore, these smaller models reduce computational costs in both training and recognition.

## 6.2 Derivation of the Algorithm

Our proposed algorithm is based on the conclusion from Chapter 4: the convergence speed of training a log-linear model – the convex case of a neural network with softmax output layer – is improved by normalizing mean and variance of the input features. For neural networks, only the input to the lowest layer can be normalized directly, because the input to the other layers changes dynamically during training. The idea of our proposed algorithm is to shift the outputs of all hidden nodes based on their running averages. The shift is compensated by transforming the parameters of the subsequent layer. In this way, the function modeled by the network remains unchanged, but its parametrization is more amenable for numerical optimization. This whole operation can be performed implicitly as a second-order optimization algorithm.

In the following, we formalize this idea. For terms of simplicity, we consider only the parameters of one layer of the network. Let $W \in \mathbf{R}^{D_1 \times D_2}$ denote the weight matrix and $b \in \mathbf{R}^{D_2}$ the bias vector of this layer. The objective function for a single training example $(x, c)$ can be written as

$$\mathcal{F}(W, b) = \mathcal{G}(W^\top z + b) \; . \tag{6.2}$$

Here, $z$ is the input to the layer and therefore a function of $x$. The composition of the non-linearity of the layer, the remaining higher layers of the network, and the loss-term are denoted by $\mathcal{G}$. The loss-term depends on the class-identity $c$ of the training example.

By shifting the feature with a vector $a \in \mathbf{R}^{D_1}$, we obtain a new objective function

$$\widetilde{\mathcal{F}}(W, b) = \mathcal{G}(W^\top (z + a) + b) \; . \tag{6.3}$$

It is possible to map the parameters corresponding to the original feature space to those corresponding to the transformed features and vice versa. With

$$\phi(W, b) = (W, b - W^\top a) \; , \tag{6.4}$$

we have

$$\mathcal{F}(W, b) = \widetilde{\mathcal{F}}(\phi(W, b)) \tag{6.5}$$

and

$$\widetilde{\mathcal{F}}(W, b) = \mathcal{F}(\phi^{-1}(W, b)) \; . \tag{6.6}$$

Instead of explicitly normalizing the features and optimizing $\widetilde{\mathcal{F}}$, the parameters can be mapped to the normalized parameter space with $\phi$, updated with the SGD rule Eq. (1.34), and mapped back to the original parameter space with $\phi^{-1}$:

$$(\hat{W}, \hat{b}) := \phi^{-1}\Big(\phi(W, b) - \eta \cdot \nabla\widetilde{\mathcal{F}}\big(\phi(W, b)\big)\Big) \; . \tag{6.7}$$

This modified update rule requires the gradient of $\widetilde{\mathcal{F}}$, which can be calculated using chain rule:[1]

$$\begin{aligned} \nabla_W \widetilde{\mathcal{F}}(W, b) &= \nabla_W \mathcal{F}\big(\phi^{-1}(W, b)\big) + a \cdot \nabla_b \mathcal{F}\big(\phi^{-1}(W, b)\big)^\top \; , & (6.8) \\ \nabla_b \widetilde{\mathcal{F}}(W, b) &= \nabla_b \mathcal{F}\big(\phi^{-1}(W, b)\big) \; . & (6.9) \end{aligned}$$

Inserting into Eq. (6.7) leads to the update rule of our proposed mean-normalized stochastic gradient descent (MN-SGD) algorithm:

$$\begin{aligned} \hat{W} &= W - \eta \cdot \Big(\nabla_W \mathcal{F}(W, b) + a \cdot \nabla_b \mathcal{F}(W, b)^\top\Big) \; , & (6.10) \\ \hat{b} &= b - \eta \cdot \Big(\nabla_W \mathcal{F}(W, b)^\top \cdot a + (1 + a^\top a)\nabla_b \mathcal{F}(W, b)\Big) \; . & (6.11) \end{aligned}$$

In order to obtain zero-mean features, we set the shift $a$ to the negative of the activation mean, which can be calculated by running averages:

$$a_i = -\gamma \, \mathbf{E}\left(z|\mathcal{B}_i\right) + (1 - \gamma) a_{i-1} \; . \tag{6.12}$$

Here, $z$ is the activation of the layer, $\mathbf{E}(z|\mathcal{B}_i)$ is the activation mean on mini-batch $\mathcal{B}_i$, and $0 < \gamma < 1$ is a smoothing factor. Since the activations typically have a high variance, strong smoothing is required. We observed that the proposed algorithm is not sensitive to the choice of the smoothing factor and set it to $\gamma = 0.005$ in all experiments.

Note that this update rule can be written in the form of a general stochastic second-order update in the sense of Eq. (6.1). Writing the parameters in vectorized form as $\theta = (\mathrm{vec}(W), \mathrm{vec}(b))$ and omitting the iteration indices, the matrix $B$ in Eq. (6.1) is of the form

$$B = \begin{pmatrix} I_{D_1 D_2} & A^\top \\ A & (1 + a^\top a) I_{D_2} \end{pmatrix} \; , \tag{6.13}$$

[1] See Lemma C.1 in Appendix C for a detailed derivation.

where $A$ is defined as

$$A = \begin{pmatrix} a_1 \dots a_{D_1} & & \\ & \ddots & \\ & & a_1 \dots a_{D_1} \end{pmatrix} \in \mathbf{R}^{D_2 \times D_1 \cdot D_2} \,. \tag{6.14}$$

Analogous formulas can be derived for an implicit variance normalization. However, in initial experiments, we did not observe improvements by normalizing the variance. Therefore, we have not considered this approach further.

## 6.3 Convergence Proof

To prove convergence of the algorithm, we need to restrict ourselves to the strictly convex case, e.g. a single layer network trained according to the $\ell_2$-regularized cross-entropy criterion. Note that this assumption is required for almost all results on convergence guarantees. Our convergence proof is an application of Theorem 3.2 in [Sunehag & Trumpf$^+$ 09], which follows from a very general result by [Robbins & Siegmund 71], which makes use of supermartingale theory.

Sunehag et al.'s theorem states that a stochastic second-order algorithm converges almost surely if the matrix $B$ in Eq. (6.1) is symmetric and its eigenvalues are *uniformly bounded* below and above, i.e. bounded by positive numbers $m$ and $M$, which are independent of $a$. Further, commonly used assumptions on the learning rates and mild technical assumptions on the objective function are required.

The matrix $B$ in Eq. (6.13) is symmetric. Its block-structure allows for the computation of the extremal eigenvalues:[2]

$$\lambda_{\max,\,\min} = 1 + \frac{1}{2}\|a\|^2 \pm \sqrt{\|a\|^2 + \frac{1}{4}\|a\|^4} \,. \tag{6.15}$$

Assuming that $a$ is bounded, one can deduce that the extremal eigenvalues are uniformly bounded. It is easy to see that the maximal eigenvalue is bounded above, since it is monotonically increasing in $\|a\|$. To show the uniform lower bound, consider $\lambda_{\min}$ as a function of $\|a\|$ on the compact interval $[0, K]$, where $K > 0$ is the bound of $\|a\|$. With $\|a\|$ equal to zero, $\lambda_{\min}$ is 1. Further, $\lambda_{\min}$ is strictly positive, because for all $a$

$$1 + \frac{1}{2}\|a\|^2 = \sqrt{1 + \|a\|^2 + \frac{1}{4}\|a\|^4} > \sqrt{\|a\|^2 + \frac{1}{4}\|a\|^4} \,. \tag{6.16}$$

Hence, $\lambda_{\min}$ must attain a positive minimum on the compact interval $[0, K]$. This proves almost sure convergence of MN-SGD.

[2] Use the identity $\det \begin{pmatrix} A & C^\top \\ C & B \end{pmatrix} = \det(A) \det(B - CA^{-1}C^\top)$ .

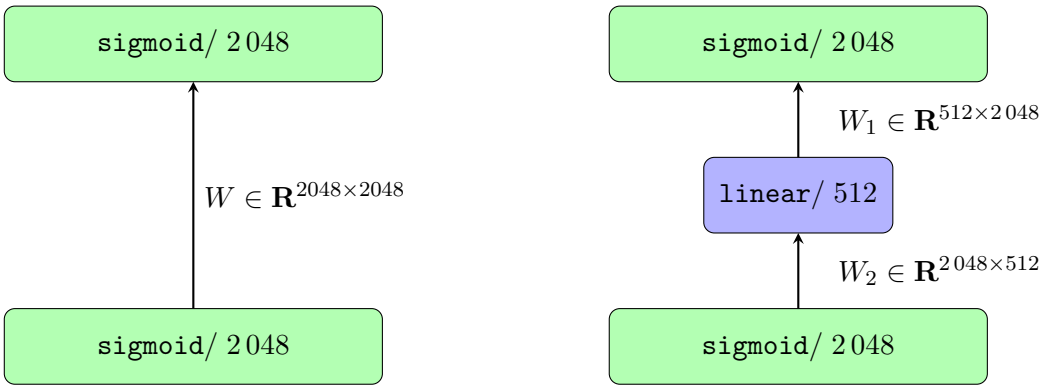


**Figure 6.1.** Illustration of a conventional neural network layer (left) and its counterpart with a linear bottleneck (right)

## 6.4 Improving Generalization Ability by Low-Rank Factorization

A general problem of more efficient optimization algorithms on machine learning tasks is that higher training accuracy can cause overfitting. A standard means against overfitting is the use of regularization, e.g. $\ell_2$-regularization. However, in preliminary experiments, we could not gain improvements in recognition word error rate by using $\ell_2$-regularization and therefore were searching for an alternative.

Overfitting can also be avoided by restricting the model size. This has the additional benefit of accelerating both training and recognition. The size of a deep neural network can be reduced by decreasing the number of layers or the number of hidden nodes per layer, as in our experiments with the Hessian-free algorithm in the previous chapter. But in these experiments, we could not gain improvements in comparison to the largest model.

Recently, a more sophisticated approach for reducing the size of a deep neural network has been proposed. The *linear bottleneck topology* introduced in [Sainath & Kingsbury$^+$ 13] is motivated by the observation that weight matrices are often effectively of low rank, i.e. most of their singular values are close to zero. Such a weight matrix $W \in \mathbf{R}^{D_1\times D_2}$ can be approximated well by a low-rank product

$$W \approx W_1 W_2 \tag{6.17}$$

with $W_1 \in \mathbf{R}^{D_1\times r}$ and $W_2 \in \mathbf{R}^{r\times D_2}$ and small $r > 0$. Figure 6.1 illustrates that this factorization is equivalent to inserting a linear bottleneck between two layers of the network. Instead of factorizing the weight matrices of an already-trained network, one can directly train a model with a linear bottleneck structure from scratch. Because of the smaller size of the model, training and recognition are accelerated. Sainath et al. found this technique to be effective for the output layer, but inserting linear bottlenecks

in the hidden layers degraded the result. They conjectured that the hidden layers do not have an underlying structure that allows for a low-rank factorization.

In another recent work [Xue & Li$^+$ 13], it has been confirmed that linear bottlenecks in hidden layers degrade performance when the deep neural network is trained from scratch using SGD. But Xue et al. could achieve a factorization of the hidden layers by first training a full-sized model, then factorizing the weight matrices by means of a singular value decomposition (SVD), and finally retraining the model. Using this approach, they could reduce model size by 80 percent without loss in accuracy. Although this approach does not accelerate training, it reduces the computational costs of evaluating the network in recognition.

Xue et al.'s result clearly shows that the degradation observed with linear bottlenecks in hidden layers is caused by optimization difficulties. The approach based on a singular value decomposition yields an accurate model with linear bottlenecks, but it cannot be found using SGD from scratch. In our experiments, we study whether our proposed MN-SGD algorithm is capable of optimizing neural networks with linear bottlenecks between all layers from scratch. While Sainath et al. and Xue et al. were using the linear bottleneck structure as a method for reducing computational costs only, we also study its use as a regularization method.

## 6.5 Learning Rate Strategies

A critical aspect for the performance of DNNs is the choice of a learning rate strategy, see e.g. [Senior & Heigold$^+$ 13]. For our neural network baseline systems, we use the popular *Newbob* learning rate strategy as it is implemented in the QuickNet software [Johnson 04], i.e. the learning rate is kept fixed as long as the frame error rate (FER) on the cross-validation set improves by at least 0.5 percent. In all subsequent epochs, the learning rate is halved. In contrast to the learning rate strategy in Eq. (3.40), which we used for log-linear models, Newbob does not require tuning of hyperparameters. Further, Newbob partly prevents overfitting by controlling the learning rate based on the cross-validation frame error. However, we noted that the standard implementation of Newbob decays the learning rate too quickly and it is preferable to reduce the learning rate only when there is an additional epoch without sufficient improvement on the cross-validation set.

For our purpose, the Newbob strategy is not well suited, because it prevents direct optimization on the training data. Thus, a direct comparison of SGD and MN-SGD in terms of an optimization performance is not possible. Instead, we use Newbob, but replace the validation set by a representative subset of the training data. Overfitting is avoided by early stopping, i.e. we use the model with the lowest word error rate on the development data. In the following we denote these learning rate strategies as Newbob/CV and Newbob/Train.

## 6.6 Experimental Results

We validated our proposed method on three tasks from small to large-scale complexity. Our focus is on the English Quaero 2011 task, a challenging broadcast conversations recognition task. We performed experiments on two variants of this task. In the first set of experiments, only a fifty hour subset of the acoustic training data has been used (Quaero/50h). This task is used as a benchmark task at RWTH. In the second set of experiments, the complete acoustic training data has been employed (Quaero/Full). Details on the tasks and our baseline systems are given in Appendix A.2.2.

In addition to these speech recognition tasks, we performed experiments on the IAM handwriting recognition task, building on the 2014 baseline system described in Appendix A.4.2.

### 6.6.1 Conversational speech recognition

All neural networks considered in this section were trained according to the cross-entropy criterion with the 4 501 emission model labels of the GHMM baseline system as outputs. After removing ten percent of the training data for cross-validation, 15.5 million training examples remained.

The input features to the neural networks were obtained by concatenating 16-dimensional MFCC vectors in a context window of size 17, the first derivatives of these vectors, and the second derivate of the first MFCC in a context window of size 13. This results in a feature dimension of 493. A global mean and variance normalization has been applied to the features. The baseline neural network has six 2 048-dimensional hidden layers with sigmoid activation function and a softmax output layer.

For the experiments with bottleneck models as described in Section 6.4, we placed a linear bottleneck between all hidden-to-hidden connections and the hidden-to-output connection. Statistics on the size of the considered models are given in Table 6.1. It is noteworthy that the size of the model without bottleneck structure is more than twice the number of training examples. However, in initial experiments, we determined that reducing the number of layers or the number of hidden nodes degrades performance.

All trainings were initialized with supervised layerwise pre-training as described in [Seide & Li$^+$ 11b], with either SGD or MN-SGD. The mini-batch size for SGD and MN-SGD has been set to 512. All experiments were performed with our open source neural network tool, which is described in Appendix B.

#### Experiments on the small-scale task

The experimental results on the Quaero/50h task are summarized in Table 6.2. The DNN baseline trained with Newbob/CV achieves 19.1 percent word error rate on the development data, which is a relative improvement of 19.1 percent in comparison to

**Table 6.1.** Number of parameters of bottleneck networks and reduction in comparison to the conventional DNN

| **Bottleneck dimension** | **#Params.** | **Reduction** [%] |
|---|---|---|
| - | 31.2M | 0.0 |
| 512 | 14.7M | 52.4 |
| 256 | 7.9M | 74.5 |
| 128 | 4.5M | 85.6 |
| 64 | 2.8M | 91.2 |

**Table 6.2.** Results on the Quaero/50h task. The first two columns specify the type of the learning rate schedule and the bottleneck dimension. Columns three to six show the results of SGD: the epoch used in recognition, the frame error rate on the training data at this point, and the word error rate on the development and test data. Analogously, columns seven to ten show the results of MN-SGD.

| | | **SGD** | | | | **MN-SGD** | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Newbob** | **BN** | **Ep.** | **FER** [%] | **WER** [%] | | **Ep.** | **FER** [%] | **WER** [%] | |
| | | | Train | Dev | Test | | Train | Dev | Test |
| CV | - | 17 | 50.1 | 19.1 | 25.2 | 13 | 47.8 | 19.0 | 25.0 |
| Train | - | 26 | 44.4 | 18.5 | **24.6** | 17 | 41.6 | 19.7 | 26.0 |
| Train | 512 | 25 | 49.7 | 18.8 | 25.0 | 25 | 31.0 | 18.3 | 23.9 |
| | 256 | 26 | 54.2 | 19.4 | 25.5 | 26 | 40.1 | 18.0 | **23.7** |
| | 128 | 20 | 58.1 | 21.1 | 27.7 | 23 | 48.4 | 18.3 | 24.2 |
| | 64 | 24 | 59.1 | 21.6 | 28.0 | 27 | 52.2 | 18.8 | 24.7 |

the discriminatively trained GHMM baseline.[3] For fair comparison, we selected the Newbob/CV model with the minimal recognition error rate instead of simply selecting the final model as it is usually done. The neural network trained with SGD and Newbob/Train outperforms the baseline with 18.5 percent word error rate. Comparing the training frame error obtained with the different learning rate schedules, one can presume that the conventional Newbob strategy terminates training too early.

In a first step, we compared the performance of SGD and MN-SGD for training models without linear bottlenecks. The results in the first two rows of Table 6.2 show

[3]Note that our results are slightly better than in [Wiesler & Richard$^+$ 14b] because of a technical detail. Our phoneme set contains a garbage model which is used for pronunciations of short word fragments, cf. Section 7.6.2. In [Wiesler & Richard$^+$ 14b], we excluded the garbage model states from neural network training. For the sequence training experiments in Chapter 7, we repeated the experiments with the garbage model states included in the optimization. Unexpectedly, this improved the performance of the system slightly.

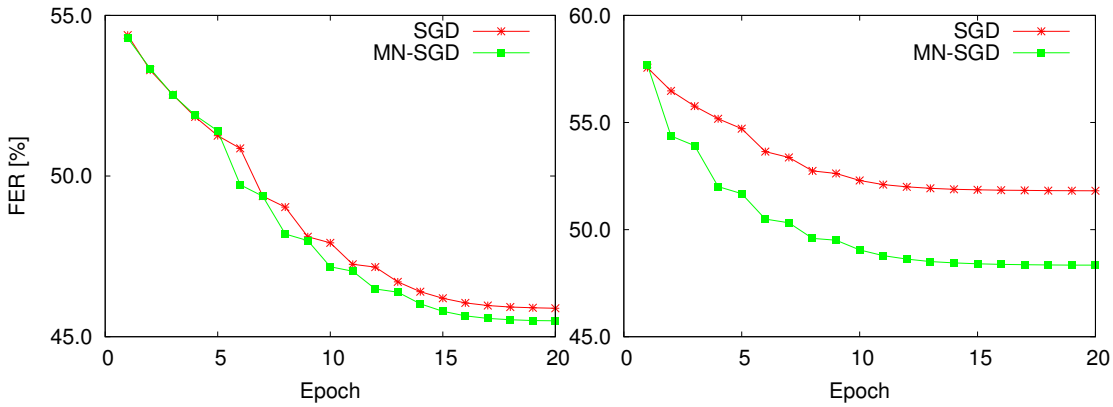


**Figure 6.2.** Evolution of training frame error rate of SGD and MN-SGD for conventional DNNs (left) and models with bottlenecks of size 256 (right) for the training on the complete Quaero 2011 data

that MN-SGD achieves a lower training frame error rate already in earlier epochs and is thus advantageous as an optimizer. However, in these experiments, the improvements on the training data do not transfer to unseen data. The result with MN-SGD and Newbob/Train is worse than the baseline. Since we use very large models in comparison to the amount of training data, it is crucial to avoid overfitting and there is nothing to gain in terms of word error rate by improving the optimization.

Next, we considered models of smaller size with linear bottlenecks. The performance of such models degrades when they are trained with SGD, see the remaining rows in Table 6.2. The error rate already slightly degrades with a reduction of the parameters by a factor of two. Reducing the number of parameters further, increases the error rate strongly. These results are in line with those of [Sainath & Kingsbury$^+$ 13, Xue & Li$^+$ 13].

Using MN-SGD, training behaves completely differently. Up to a certain point, the error rate decreases with the model size due to improved generalization. The best result of 18.0 and 23.7 percent word error rate on the development respectively test data is obtained with a model that is roughly four times smaller than the full model. We still achieve improvements on the development and test data with a parameter reduction of 85.6 percent. Even the model with a 64-dimensional bottleneck performs only marginally worse than the baseline model. This model has only 2.8 million parameters. For comparison, a fully connected neural network with only a single hidden layer has 10.2 million parameters.

### Experiments on the large-scale task

In a second series of experiments, we investigated whether the improvements obtained on the rather small training set scale to the large training corpus. The gains from using

**Table 6.3.** Results on the complete Quaero 2011 training data

| | | SGD | | | | MN-SGD | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Newbob** | **BN** | **Ep.** | **FER [%]** | **WER [%]** | | **Ep.** | **FER [%]** | **WER [%]** | |
| | | | Train | Dev | Test | | Train | Dev | Test |
| CV | – | 17 | 47.9 | 15.7 | 21.1 | 19 | 48.0 | 15.7 | 20.9 |
| Train | – | 17 | 45.9 | 15.2 | **20.9** | 17 | 45.6 | 15.5 | 20.6 |
| | 512 | 17 | 49.2 | 15.5 | 20.9 | 17 | 45.5 | 14.9 | **19.9** |
| Train | 256 | 12 | 51.8 | 16.2 | 21.5 | 15 | 48.4 | 15.2 | 20.5 |
| | 128 | 13 | 53.5 | 17.1 | 22.5 | 14 | 50.5 | 15.7 | 20.9 |

linear bottleneck models might be smaller, because overfitting is less severe on the larger task. The experimental setup was the same as on the small-scale task. Ten percent of the acoustic training data have been used for cross-validation. The remaining training set consists of 75.1 million training examples.

Figure 6.2 shows the evolution of the training frame error rate of SGD and MN-SGD for models with and without linear bottlenecks. The training on the large dataset shows the same characteristic behavior as observed on the small dataset. The difference between the two optimization algorithms for conventional DNNs is only minor. Both algorithms reach almost the same training accuracy. In contrast to SGD however, MN-SGD is capable of optimizing models with linear bottlenecks from scratch.

The recognition results are shown in Table 6.3. The general behavior is the same as on the smaller training set. With SGD, the word error rate increases with smaller model size. In contrast, reducing the model size improves the MN-SGD results. As expected, a larger bottleneck is required for optimal performance on the larger training set. With less than half of the parameters, the word error rate is improved from 15.2 to 14.9 percent on the development data and from 20.9 to 19.9 on the test data. The model size can be reduced up to a factor of about eight without degradation in comparison to the baseline system.

### 6.6.2 Offline handwriting recognition

Finally, we verified the efficacy of our approach on the IAM task for offline handwriting recognition. We applied the same algorithms and the same software as in our speech recognition system to this task, only the feature extraction part differs. Our baseline is the IAM 2014 system described in Appendix A.4.2. Among the three tasks considered in this chapter, IAM is the smallest. The training set contains 3.6 million observations.

The experimental setup was very similar to that on the speech recognition tasks. The neural networks were trained according to the cross-entropy criterion with the 563 states of the character models as outputs. The input features were obtained by concatenating the 24-dimensional features of the GHMM system in a symmetric context window of size

**Table 6.4.** Results on the IAM handwriting recognition task

| Alg. | Newbob | BN | #Params. (Red.) | Ep. | FER [%] | WER [%] | |
|---|---|---|---|---|---|---|---|
| | | | | | Train | Dev | Test |
| SGD | CV | - | 14.5M | 12 | 9.0 | **11.1** | **15.0** |
| | Train | | | 8 | 14.6 | 11.2 | 15.3 |
| MN-SGD | Train | 256 | 4.6M (68.5%) | 10 | 15.2 | 11.1 | 14.6 |
| | | 128 | 2.7M (81.7%) | 11 | 18.3 | **11.1** | **14.6** |
| | | 64 | 1.7M (88.3%) | 15 | 16.0 | 11.2 | 14.8 |
| | | 32 | 1.2M (91.6%) | 12 | 20.8 | 11.5 | 16.1 |

15. The global mean and variance of the input features were normalized before training. The baseline model had four 2 048-dimensional hidden layers with sigmoid activation function and a softmax output layer. In contrast to the speech recognition experiments, we did not use discriminative pre-training, because we observed that it causes overfitting on this small database. We used a mini-batch size of 128 for all experiments. We only performed the most promising experiments on IAM, i.e. we compared the baseline system trained with SGD and a bottleneck network system trained with MN-SGD.

The experimental results are summarized in Table 6.4. In general, all models achieve a very similar error rate on the development set, since the error rate of the baseline model is already very low. The baseline model and the model with a 128-dimensional bottleneck achieve the same performance on the development set. On the test set, the bottleneck model improves over the baseline with 14.6 percent word error rate compared to 15.0 percent word error rate.

## 6.7 Discussion

In this chapter, we developed a novel stochastic second-order optimization algorithm and proved its convergence in a convex setting.

In experiments on speech and handwriting recognition tasks, we have shown that our proposed algorithm is capable of training deep neural networks with a linear bottleneck structure from scratch. Thus, training and recognition are accelerated due to the reduced size of the models. Further, we found that the linear bottleneck structure is a very effective means against overfitting. Across all tasks, we obtained consistent recognition error rate improvements with much smaller models.

Our approach extends the work by [Sainath & Kingsbury$^+$ 13] and [Xue & Li$^+$ 13]. Sainath et al. trained linear bottleneck models with SGD from scratch, but this approach is only effective for the output layer. Xue et al.'s method allows for using linear bottlenecks between all layers, but it requires training a full model first. Therefore, their method does not accelerate training. The value of the linear bottleneck structure as a regularization method has not been identified in [Sainath & Kingsbury$^+$ 13, Xue & Li$^+$

13].

An interesting work related to ours is [Raiko & Valpola$^+$ 12], where zero mean activations are achieved by adapting the activation functions dynamically. Their approach requires a network structure with shortcut connections, which bypass the hidden layer. This is in particular difficult to realize with deep networks. In contrast, we proposed a general optimization algorithm which can be applied to neural networks with arbitrary topology.

A limitation of the experimental study in this chapter is that the models are trained according to frame-discriminative criteria. Sequence-discriminative training of neural networks is the topic of the following chapter.

## 6.8 Publications and Joint Work

The work described in this chapter is based on the publication [Wiesler & Richard$^+$ 14b]. The author derived the proposed algorithm, contributed the idea to apply it to linear bottleneck networks, and the idea to proof its convergence using the theorem by [Sunehag & Trumpf$^+$ 09]. The experiments on the speech recognition task and the implementation as part of the RASR neural network module [Wiesler & Richard$^+$ 14a] were performed by A. Richard under the author's supervision. These experiments were redone for this thesis for consistency with other chapters and extended by experiments on the handwriting recognition task. The baseline system for the handwriting recognition task builds on the system developed by M. Kozielski and P. Doetsch [Kozielski & Doetsch$^+$ 13].

# Chapter 7

# Sequence-Discriminative Training of Neural Networks

Acoustic models in hybrid neural-network-HMM speech recognition systems are typically trained on frame level, usually according to the cross-entropy criterion. In a number of recent works [Kingsbury 09,Kingsbury & Sainath$^+$ 12,Su & Li$^+$ 13,Veselý & Ghoshal$^+$ 13,Heigold & McDermott$^+$ 14,McDermott & Heigold$^+$ 14], it has been shown that neural-network-HMMs can be improved substantially by training them according to the sequence-discriminative criteria, which we discussed in Chapter 3. Sequence-discriminative training – or short *sequence training* – is considered as an essential step in building state-of-the-art speech recognition systems.

However, sequence-discriminative training of neural networks has turned out to be a complex technique, which is not well understood yet. Several authors observed problems with its stability and proposed a variety of solutions. From a practical point of view, sequence-discriminative training is also challenging because of its high computational costs.

The goal of this chapter is to gain more empirical insight into this important technique. Further, we extend our work on training linear bottleneck networks from scratch presented in the previous chapter. Experiments are performed on a small-scale handwriting recognition task and medium to large-scale speech recognition tasks [Wiesler & Golik$^+$ 15].

## 7.1 Introduction

Conceptually, sequence training of neural networks is rather straightforward. In comparison to conventional frame-based training, only the error signal of the output layer needs to be changed. The error signal is collected on a word lattice in the same way as in discriminative training of GHMMs. The forward and the error backpropagation pass of the neural network remain unchanged.

In practice, sequence training of neural networks has turned out to be a complex technique and it is not yet possible to draw clear conclusions from previous work about the best training setup. In particular, some authors found MPE or the closely related sMBR criterion to perform better than MMI [Veselý & Ghoshal$^+$ 13], while others found the converse to be true [Su & Li$^+$ 13, McDermott & Heigold$^+$ 14]. In some

papers [Heigold & McDermott$^+$ 14, Heigold & McDermott$^+$ 14], only MMI respectively sMBR has been applied, but their performance has not been compared. In addition, several authors observed problems with the stability of sequence training and proposed different modifications of the training criteria [Su & Li$^+$ 13, Veselý & Ghoshal$^+$ 13].

Another debated issue is the choice of the optimization algorithm. Most groups use SGD on utterance-level [Su & Li$^+$ 13, Veselý & Ghoshal$^+$ 13, Heigold & McDermott$^+$ 14]. [Kingsbury & Sainath$^+$ 12] used a parallelized implementation of the HF algorithm, which we studied in Chapter 5. They found HF not only to be faster than SGD in terms of wall-clock time, but also to achieve slightly better recognition results. So far, there has been no comparison of SGD with other batch algorithms in the context of sequence training.

Thinking of the hidden layers of a neural network as the feature extractor and the output layer as the classifier, sequence training can be compared with discriminative training of NN-GHMM-Tandem systems [Hermansky & Ellis$^+$ 00]. While for the latter, only the classifier is trained sequence-discriminatively, direct sequence training of neural networks typically optimizes the complete network jointly. It is an interesting question, whether this additional complexity is beneficial.

In the following, our aim is to shed more light on these open questions. In the next sections, we recap the sequence-discriminative training criteria and discuss modifications of them proposed in literature. We briefly discuss the optimization algorithms, which we considered in this work, in the context of sequence training. Then, we present our experimental results on speech and handwriting recognition tasks with increasing complexity.

## 7.2 Training Criteria

Exactly the same training criteria as for Gaussian and log-linear HMMs can be applied to neural networks as well. For brevity, we write the neural network as a log-linear model with a parameterized feature extractor $f_\vartheta(x)$ which is not further specified, i.e. for an emission model label $a$ and a feature vector $x$, we have

$$p_\theta(a|x) = \frac{1}{Z(x)} \exp(\lambda_a^\top f_\vartheta(x) + \beta_a) \ . \tag{7.1}$$

Here, $\Lambda = (\lambda_1; \ldots; \lambda_A)$ and $\beta = (\beta_1, \ldots, \beta_A)$ are the weight matrix and the bias vector of the output layer, $\vartheta$ are the parameters of the feature extractor, and $\theta = (\Lambda, \beta, \vartheta)$ is the tuple with all parameters of the network. The factor $Z(x)$ is the normalization constant.

We use the notation from Chapter 3, i.e. $\mathbf{x} = (x_1, \ldots, x_T)$ is a feature sequence with reference word sequence $\mathbf{w} = (w_1, \ldots, w_N)$. We assume that a Viterbi alignment with corresponding emission model labels $\mathbf{a} = (a_1, \ldots, a_T)$ and a word-pronunciation lattice $\mathcal{L}$ are given.

Recall that the cross-entropy criterion is given by

$$\mathcal{F}^{(\mathrm{CE})}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(a_t|x_t) \ . \tag{7.2}$$

With the definition of the utterance-posterior probability as in Eq. (3.23), the sequence-discriminative MMI criterion is defined as

$$\mathcal{F}^{(\mathrm{MMI})}(\theta) = -\log p_\theta(\mathbf{w}|\mathbf{x}) \ . \tag{7.3}$$

The MPE criterion is an instance of MBR with the cost $\mathcal{C}$ defined as the approximate phone error [Povey & Woodland 02]:

$$\mathcal{F}^{(\mathrm{MPE})}(\theta) = \sum_{\boldsymbol{\pi}\in\mathcal{L}} p_\theta(\boldsymbol{\pi}|\mathbf{x})\ \mathcal{C}_{\mathbf{w}}[\boldsymbol{\pi}] \ . \tag{7.4}$$

## 7.3 Modifications for Robust Training

In several works, problems with the stability of sequence training has been reported. The following modifications have been proposed to improve the performance of sequence training.

### 7.3.1 Cross-entropy smoothing

[Su & Li$^+$ 13] emphasize that lattice sparsity is an inherent cause of instability of lattice-based sequence training. Even in very dense lattices, only a fraction of the classes is represented at every frame. Unfavorable scores of unrepresented classes do not affect the objective function at all. In other words, when the model changes too much from the one used for lattice generation, there is a strong mismatch between objective function and recognition error rate. This problem is especially severe with stochastic optimization algorithms because of their frequent model updates.

As a solution to this problem, Su et al. proposed smoothing the sequence-discriminative objective function with the frame-level objective function. This yields for example the smoothed MMI criterion:

$$\mathcal{F}^{(\mathrm{sm-MMI})} = (1-\gamma)\mathcal{F}^{(\mathrm{MMI})} + \gamma\mathcal{F}^{(\mathrm{CE})} \ , \tag{7.5}$$

with an interpolation factor $0 < \gamma < 1$.

The problem of lattice sparsity can be circumvented completely by computing the lattices for every utterance on-demand [Heigold & McDermott$^+$ 14]. This approach however is only feasible within a complex software framework with a heavily parallelized implementation of asynchronous SGD.

### 7.3.2 Frame-rejection heuristic

The MMI objective function is unbounded, which makes MMI training sensitive to outliers. [Veselý & Ghoshal$^+$ 13] proposed a *frame-rejection heuristic* to make MMI training more robust. According to the heuristic, all frames, where the probability of the reference state given the whole observation sequence is below a small threshold, are discarded. Given a lattice with Viterbi alignments $a(t, \boldsymbol{\pi})_{t=1,\dots,T}$ per path $\boldsymbol{\pi}$, this probability is computed as

$$\gamma_t(a) = \sum_{\boldsymbol{\pi} \in \mathcal{L}} p(\pi|\mathbf{x}) \delta(a, a(t, \boldsymbol{\pi})) \ . \tag{7.6}$$

This quantity is also known as *state occupancy* or *generalized forward-backward probability*, cf. [Schlüter 00, p38]. A frame is discarded, if

$$\gamma_t(a_t) < \epsilon \tag{7.7}$$

for a threshold $\epsilon > 0$. The state occupancies are computed dynamically for every utterance. Therefore, the decision whether a frame is discarded or not may differ between epochs.

## 7.4 Optimization

The gradient of the sequence-discriminative criteria can be computed by backpropagation as in cross-entropy training, only the error signal at the output layer is computed differently. The error signal is accumulated on a word lattice in the same way as in sequence training of log-linear or Gaussian mixture models.

In general, any gradient-based numerical optimizer can be used for sequence training. For frame-discriminative training, SGD on a single GPU or asynchronous SGD in a distributed fashion are the de-facto standard. Most groups also use SGD for sequence training [Kingsbury 09, Su & Li$^+$ 13, Veselý & Ghoshal$^+$ 13, Heigold & McDermott$^+$ 14]. The advantages of stochastic optimization are less compelling in the case of sequence training. First, the frequent model updates let the lattices deviate quickly from the potential search space of the current model, which is the root cause of training instability. Second, SGD on utterance level is less efficient, because the model updates are more heterogeneous due to poorer data shuffling and the varying utterance length.

In contrast to frame-discriminative training, sequence training is initialized with a good model. In this case, batch algorithms benefit strongly from second-order information. Finally, batch algorithms can be parallelized straightforwardly and do not require fiddling with learning rates.

Motivated by its good performance for log-linear models, we evaluate the batch algorithm Rprop [Riedmiller & Braun 93] as an alternative to SGD. Rprop's advantages over the HF algorithm used in [Kingsbury & Sainath$^+$ 12] are its small number of tuning parameters, its simplicity, and that it has no computational overhead beyond the gradient computation.

**Table 7.1.** Experimental results on the IAM handwriting recognition task

| Criterion | BN | Epoch | FER [%] | WER [%] | |
|---|---|---|---|---|---|
| | | | Train | Dev | Test |
| CE | - | 12 | 9.0 | 11.1 | 15.0 |
| MMI | - | 12 | 43.0 | 11.0 | 14.2 |
| CE | 128 | 11 | 18.3 | 11.1 | 14.6 |
| MMI | 128 | 4 | 46.3 | 10.6 | 12.7 |

## 7.5 Implementation

We implemented the sequence training in the neural network tool, which is part of the RASR speech recognition toolkit described in Appendix B. As [Veselý & Ghoshal$^+$ 13], we chose a hybrid CPU/GPU approach. The neural network forward and backward pass are performed on GPU. The lattice computations are performed on CPU. For the lattice computations, we can re-use the discriminative GHMM training code, which has already been implemented in RASR. Since the neural network forward and backward pass are the same as in frame-discriminative training, the neural network code can be re-used as well. The challenge in the implementation is mostly to improve the efficiency of the CPU code, since this is the the performance bottleneck in a hybrid implementation. The computation time for sequence training depends highly on the hardware and the ratio of the size of the model and the lattices. In our experiments, one epoch of sequence training was roughly twice as expensive as one cross-entropy epoch.

## 7.6 Experimental Results

In this section, we present our results using sequence training. The experiments were performed on the same tasks as in the previous chapter, i.e. the handwriting recognition task IAM, and the English broadcast conversations recognition task Quaero.

### 7.6.1 Offline handwriting recognition

In our experiments on the IAM handwriting recognition task, we applied the most straightforward approach with SGD as the optimization algorithm and standard MMI as the training criterion.

We applied sequence training to the cross-entropy trained neural networks from Section 6.6. For comparison, we also report experiments of sequence training with bidirectional LSTM-RNNs, which have been obtained in a joint work[1] with P. Voigtländer and P. Doetsch [Voigtlaender 14, Voigtlaender & Doetsch$^+$ 15]. The LSTM-RNN had three hidden layers with 500 memory cells for both forward and backward direction.

[1] See author's contribution in Appendix E.

The network has been trained using backpropagation through time without truncation and SGD with a batch size of 30 sequences. The conventional four-layer feed-forward neural network, the network with 128-dimensional bottlenecks, and the LSTM achieved 15.0, 14.6, respectively 14.5 percent word error rates on the test set.

The lattices for the sequence training have been generated by decoding the training data with a unigram language model, the training lexicon, and the cross-entropy trained neural networks. The size of the lattices has been reduced using forward-backward pruning, and afterwards the force-aligned reference has been merged into the lattice. The average number of lattice arcs per reference word was about fifty. The lattices have been kept fixed for the complete training. The language model for sequence training was the same as the one used for lattice generation. The sequence training has been initialized with the cross-entropy model. We used fixed SGD learning rates between 1e-3 and 1e-5. For each training, the best epoch has been determined on the development set.

Already with this simple setup, sequence training improves the feed-forward network by about five percent relative, see Table 7.1. We also observed that – as expected – the frame error rate of the sequence-discriminatively trained model is higher than that of the frame-discriminatively trained model. This sequence-trained neural network already outperforms the cross-entropy trained LSTM. As demonstrated in [Voigtlaender & Doetsch$^+$ 15], the LSTM can be improved by sequence training as well using exactly the same approach, and then achieves an error rate of 13.5 percent word error rate, cf. Table A.6.

Analogously, we also applied sequence training to the bottleneck network from Chapter 6. For optimization we used MN-SGD instead of SGD. Apart from that, the training setup was identical. As can be seen in Table 7.1, we obtain an even higher relative improvement on the already stronger cross-entropy baseline. The obtained result of 12.7 percent word error rate even outperforms the sequence-discriminatively trained LSTM network. To compare this model with the latest state-of-the-art, we further applied the open vocabulary decoding technique described in [Kozielski & Rybach$^+$ 13]. The result of 11.9 percent word error rate on the test set is to our knowledge the best published result on this database so far. The results by [Voigtlaender & Doetsch$^+$ 15, Doetsch & Kozielski$^+$ 14, Kozielski & Doetsch$^+$ 13] are directly comparable to our result, since they use exactly the same feature extraction, lexicon, language model, and decoder setup. The fact that our rather simple system outperforms those obtained with a more sophisticated LSTM-RNN architecture highlights the importance of regularization for neural network training, in particular in the case of limited training data as on this task.

### 7.6.2 Conversational speech recognition

This section presents our experiments on the two variants of the English Quaero 2011 task described in Appendix A.2.2. The setup of the speech recognition system is the same as in Chapter 6.

**Table 7.2.** Comparison of the proposed system to results reported by other groups on the IAM database.

| System | WER [%] | |
|---|---|---|
| | Dev | Test |
| Our system (BN-DNN) | 8.9 | 11.9 |
| [Voigtlaender & Doetsch$^+$ 15] (LSTM, MMI) | 8.7 | 12.7 |
| [Doetsch & Kozielski$^+$ 14] (modified LSTM) | 8.4 | 12.2 |
| [Kozielski & Doetsch$^+$ 13] (LSTM/GHMM Tandem) | 9.5 | 13.3 |
| [Pham & Bluche$^+$ 14] (LSTM, dropout) | 11.2 | 13.6 |
| [Graves & Liwicki$^+$ 09] (LSTM, CTC) | - | 25.9 |
| [Bertolami & Bunke 08] (GHMM ensemble) | 26.9 | 32.9 |

Although we use the same algorithms and the same implementation for our speech and handwriting recognition systems, the tasks have different characteristics, which make the application of sequence training to conversational speech recognition more challenging.

First of all, the amount of training data differs strongly for the tasks under consideration. With our feature extraction, the number of training examples for the acoustic respectively visual model is 3.5 million for the handwriting task and roughly 75 million for the speech recognition task. Therefore, computational efficiency plays an important role. The language model used for the speech recognition task has been trained on 3.7 billion running words, in comparison to just three million words for the handwriting task, see Appendix A. With a stronger language model, improvements of the acoustic model provide smaller gains in terms of the overall performance of the system. Second, the output layer of the speech models is much larger, because of the use of context-dependent models. Thus lattice sparsity is more severe. Third, conversational speech transcriptions often contain errors, in particular with frequent speaker changes. This is problematic, because sequence training is known to be sensitive to transcription errors, see e.g. [Wang & Gales$^+$ 07]. Finally, conversational speech contains different kinds of noises and speech disfluencies, which are difficult to model.

#### Experiments on the small-scale task

Our initial setup for the sequence training on Quaero was the same as the one for the handwriting recognition task, except that we used a weaker lattice pruning to avoid lattice sparsity. For all speech recognition experiments, the average lattice density was between 300 and 500, which is high in comparison to the lattices typically used for discriminative GHMM training at RWTH.

While on the handwriting recognition task, sequence training in a straightforward way already improved the system, we observed a strong degradation of the models on

**Table 7.3.** Number of parameters and word error rates of the three cross-entropy baseline systems on the Quaero/50h task

| Model | Number of parameters | | WER [%] | |
|---|---|---|---|---|
| | All | Output layer | Dev | Test |
| Shallow NN | 10.2M | 9.2M | 21.8 | 28.5 |
| 6-layer DNN | 31.2M | 9.2M | 18.5 | 24.6 |
| 256-bottleneck-DNN | 7.9M | 1.2M | 18.0 | 23.7 |

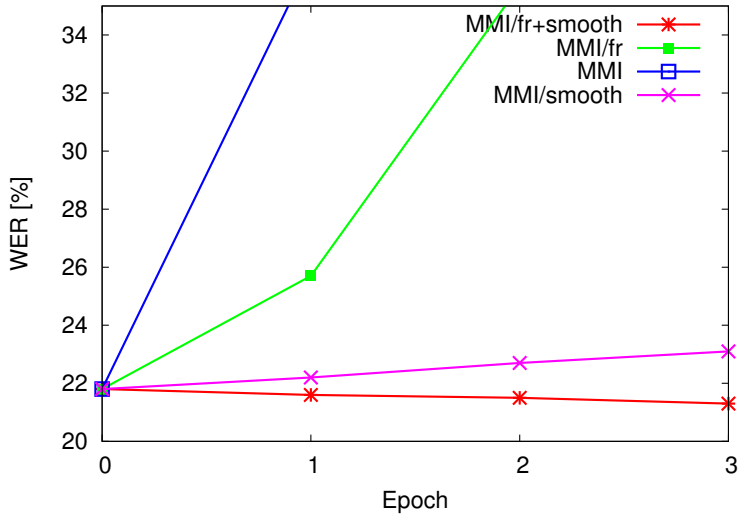


**Figure 7.1.** Evolution of the word error rate with different variants of MMI on Quaero/50h. `fr` stands for frame rejection and `smooth` for cross-entropy smoothing

the speech task. Our first results were therefore obtained in the most controlled setup – we only trained the output layer of a shallow network, which has just one hidden layer with 2 048 sigmoid units. With cross-entropy training, this model achieves an error rate of 28.5 percent on the test data, see Table 7.3.

**Training modifications** Figure 7.1 shows the evolution of the word error rate with different MMI variants. It can be seen that cross-entropy smoothing is essential to stabilize training. We observed the same behavior with MPE training (not shown in the figure). The only MMI configuration that provided improvements was the combination of cross-entropy smoothing and the frame rejection heuristic. The frame rejection heuristic threshold was set to 1e-6, which reduced the amount of training data by roughly five percent. All following results were obtained with cross-entropy smoothing with interpolation factor 0.1 and additionally the frame rejection heuristic with threshold

**Table 7.4.** Results with the shallow network with different training criteria and different language models on Quaero/50h.

| Criterion | Language model | WER [%] Dev | Test |
|---|---|---|---|
| CE | - | 21.8 | **28.5** |
| MPE | Unigram | 21.0 | 27.4 |
| | 4-gram | 20.3 | **26.8** |
| MMI | | 21.0 | 27.5 |

1e-6 in case of MMI.

**Lattice generation** Next, we inspected the lattices more closely. In our standard setup, the lattices are created by decoding the training data with the training lexicon and a unigram language model. We identified two issues with this setup. First, in our training lexicon, short word fragments are modeled with a garbage model phoneme. Such short word fragments appear frequently in conversational speech, for example in false starts like "nat- natural" or "wou- should". Since the word fragments are not contained in the recognition lexicon, the garbage model can be discarded in testing. We observed that a large number of the lattice-arcs have a garbage model pronunciation. Therefore, sequence training focuses on discriminating the garbage model from the reference state, although the garbage model is not used in the recognition lexicon at all. We fixed this by adding a large penalty on the garbage model in the network used for lattice generation. Second, we found that arcs with short words like "I" or "a" and disfluencies like "um" or "huh" are overrepresented in the lattices. This is a consequence of using a unigram language model and can be avoided with a higher-order language model, see Table 7.4. Note that this observation is in contrast to results reported in literature on discriminative training where a unigram language model is preferred, e.g. [Schlüter & Müller$^+$ 99, Povey 04, Heigold 10]. We conjecture that a higher-order language model is preferable on spontaneous speech tasks, where the word boundaries are not acoustically distinct.

**Training criterion** Finally, we found MPE to perform clearly better than MMI. With the best lattice configuration, MPE achieves a word error rate of 26.8 percent in comparison to 27.5 percent with MMI training, see Table 7.4.

In the next set of experiments, we applied sequence training to the deep neural networks from Chapter 6. Our primary interest is the benefit of the bottleneck network structure for sequence-trained models. Further, we compared the performance of Rprop with SGD and we evaluated the impact of including the hidden layers in the sequence training. According to the findings in the experiments with the shallow network, we

**Table 7.5.** Results on the Quaero/50h task

| Crit. | Alg. | Layers trained | DNN Ep. | DNN WER [%] Dev | DNN WER [%] Test | Bottleneck-DNN Ep. | Bottleneck-DNN WER [%] Dev | Bottleneck-DNN WER [%] Test |
|---|---|---|---|---|---|---|---|---|
| CE | (MN-)SGD | All | 26 | 18.5 | **24.6** | 26 | 18.0 | **23.7** |
| MPE | (MN-)SGD | All | 11 | 17.6 | 23.5 | 10 | 17.5 | 23.2 |
| | SGD | Output | 14 | 17.8 | 23.7 | 11 | 17.5 | 23.2 |
| MPE | Rprop | All | 29 | 17.5 | **23.3** | 31 | 17.4 | **23.0** |
| | | Output | 24 | 17.5 | 23.3 | 18 | 17.6 | 23.1 |

used four-gram lattices and cross-entropy smoothed MPE for these experiments. The results are shown in Table 7.5.

**Optimization algorithm** We applied Rprop with the standard hyperparameters from [Riedmiller & Braun 93]. The initial step size has been set to a small value which ensures a stable optimization start. Rprop is compared with either MN-SGD for the bottleneck network with all layers included in the optimization or SGD in all other cases. The results show that Rprop achieves the lowest error rate in all scenarios. As expected, Rprop requires more epochs than SGD. On the other hand, we ran SGD with multiple learning rates in parallel, which is not necessary for Rprop. Further, the gradient computation required for Rprop can be distributed easily to multiple machines. In our experiments, we used Rprop with ten GPUs. With this degree of parallelization, we have an almost linear speed-up. In comparison to SGD on a single GPU, Rprop is faster by a factor of about three in wall-clock time.

**Bottleneck structure** Next, we investigated whether the improvements obtained with bottleneck networks persist after sequence training. We applied sequence training only to the model with a bottleneck dimension of 256, because this model had the lowest error rate after cross-entropy training. The results are given in Table 7.5. The improvements from sequence training are smaller than with the conventional deep neural network. Still, a small but consistent improvement of the bottleneck topology remains.

**Output vs. all layers** One could expect it to be important to include the hidden layers in sequence training, because they amount to more than seventy percent of the parameters of the DNN and more than eighty percent of the bottleneck DNN. However, training the complete network only contributes to a rather small improvement of at most 0.2 percent word error rate. This justifies the typical approach taken in NN-GHMM Tandem systems, where only the GMM is trained sequence-discriminatively.

This observation would allow for reducing the computational complexity strongly by storing the activations of the last hidden layer and only training the log-linear output

**Table 7.6.** Results on the complete Quaero 2011 task

| Crit. | Alg. | Layers trained | DNN Ep. | DNN WER [%] Dev | DNN WER [%] Test | Bottleneck-DNN Ep. | Bottleneck-DNN WER [%] Dev | Bottleneck-DNN WER [%] Test |
|---|---|---|---|---|---|---|---|---|
| CE | (MN-)SGD | All | 17 | 15.2 | **20.9** | 17 | 14.9 | **19.9** |
| MPE | Rprop | All | 36 | 14.1 | **19.2** | 41 | 14.0 | **18.9** |
| | | Output | 31 | 14.2 | 19.1 | 27 | 14.1 | 19.1 |

layer. In particular when using a linear bottleneck, one could train only 1.2 million parameters sequence-discriminatively in comparison to the 31.2 million parameters of the complete DNN without linear bottlenecks. However, we did not exploit this possibility in this work.

### 7.6.3 Experiments on the large-scale task

Finally, we applied sequence training to the complete Quaero 2011 task. The baseline models are the conventional six-layer DNN and the model with 512-dimensional bottlenecks from Chapter 6. Again, we used four-gram lattices and cross-entropy smoothed MPE. For optimization, we used Rprop with ten graphics processing units (GPUs). The results are shown in Table 7.6. Sequence training improves the conventional DNN from 20.9 to 19.2 percent word error rate, which is a relative reduction of 8.1 percent. The bottleneck model is improved by 5.0 percent relative. Again, the improvement of the bottleneck model is smaller, but a slight gain of the bottleneck structure persists.

## 7.7 Discussion

Sequence training has been shown to give substantial improvements on state-of-the-art speech recognition systems. However, effective sequence training remains a complex engineering task and there is no common agreement on the best training configuration.

In this chapter, we presented experiments with our implementation of sequence training. Our experiments provide more empirical evidence on the best choice of the training criterion, training enhancements, and the optimization algorithm.

We observed that sequence training with simple MMI and SGD optimization already improves our handwriting recognition system on the IAM task. On the more difficult conversational speech recognition task, sequence training in a straightforward way degraded the performance of the system. In particular, we found the cross-entropy smoothing proposed by [Su & Li$^+$ 13] to be essential for obtaining improvements. Su et al. only evaluated this technique with stochastic optimization and only applied it to MMI. So far, their idea has not been taken up by other authors. It is not exactly clear under which conditions cross-entropy smoothing is required to avoid lattice sparsity.

One factor which impacts lattice sparsity is the size of the output layer. Still, [Kingsbury & Sainath$^+$ 12] and [Veselý & Ghoshal$^+$ 13] did not require smoothing for sequence training of DNNs with large output layers. One reason might be that their lattices are generated with a WFST decoder, while we use a dynamic decoder. In our opinion, this is an important issue, which requires further analysis.

The question which optimization algorithm should be used does not have a simple answer. [McDermott & Heigold$^+$ 14] applied sequence training on very large datasets and found that asynchronous SGD converges already before the whole data set is processed even once. On the other hand, Rprop, which has been used in this work, can be parallelized straightforwardly, and does not require tuning the learning rate. While asynchronous SGD is non-deterministic and the result depends on factors as network latency, Rprop is deterministic. This is a desirable property for reproducibility of scientific results. Furthermore, training with Rprop resulted in a better recognition performance than SGD. This observation is in line with the results by [Kingsbury & Sainath$^+$ 12,Saon & Soltau 14], where the batch algorithm Hessian-free has been found to be slightly more accurate than SGD. Overall, Rprop is a good choice in terms of efficiency for small to medium-scale tasks. Rprop is also useful if optimal performance or deterministic behavior is desired. On very large-scale tasks in the order of thousands of hours of acoustic training data, asynchronous SGD seems to be preferable.

Finally, we extended our work on neural networks with a linear bottleneck structure. The sequence-trained bottleneck model achieved the best published result on the IAM database we are aware of. This simple approach outperforms a much more complex sequence-discriminatively trained LSTM-RNN. Clearly, RNNs offer more opportunities for further improvements. Still, this competitive result shows the efficiency of the bottleneck network structure as a regularization method. The observation on the speech recognition tasks is similar. A consistent gain due to the bottleneck structure persists, although the gain is here smaller than with only cross-entropy training.

## 7.8 Publications and Joint Work

This chapter is based on the publication [Wiesler & Golik$^+$ 15]. The author realized the software implementation, performed the systematic series of experiments, and analyzed the results. P. Golik integrated the code into the official RASR release. Additional experiments on the large-scale speech recognition task and the handwriting recognition task were performed in the course of this thesis.

# Chapter 8

# Scientific Contributions

The aim of this work was to investigate discriminative models for sequence recognition with a focus on optimization methods. It resulted in the following contributions that cover different aspects of modeling and training:[1]

**Training an LVCSR system discriminatively from scratch using convex optimization**

Discriminative training of Gaussian mixture HMMs requires a maximum-likelihood trained initialization. The discriminative training itself involves many approximations and heuristics such as the use of word lattices. This leads to much engineering work in practice. As an alternative, we developed a hybrid log-linear HMM speech recognition system suitable for LVCSR. The training of the model is convex, which allowed us to train the model discriminatively from scratch. The log-linear system achieved comparable performance to a discriminatively trained Gaussian mixture system, but with a simpler lattice-free training. The main steps required to achieve this performance on an LVCSR task were the use of sparse features and the application of our convergence analysis of log-linear training.

In parallel to our work, hybrid neural-network HMMs have become the new state-of-the art in speech recognition. Neural networks jointly learn the classifier and the feature space. Thereby, they overcome the main disadvantage of log-linear models: the manual definition of the feature space. The success of neural networks shows that enforcing training convexity was too restricting, but the direction of hybrid discriminative HMMs with context-dependent states as classes was the right direction.

Publications: [Wiesler & Nußbaum$^+$ 09, Wiesler & Richard$^+$ 11, Wiesler & Schlüter$^+$ 12, Wiesler & Richard$^+$ 13, Heigold & Schlüter$^+$ 12]

**Convergence analysis of log-linear training**

We theoretically analyzed the condition number of the optimization problem in log-linear training. The analysis allows for drawing practical conclusions on how the conditioning of the optimization problem can be improved. We validated the analysis empirically on handwriting recognition (Chapter 4) and speech recognition tasks (Chapter 3).

[1] Compare also Appendix E for details on author's contributions.

Our study is more general than a previous result in literature, which has been derived for analyzing neural network training [LeCun & Kanter$^+$ 90]. The convergence analysis is also the motivation for the novel stochastic second-order optimization algorithm, which we developed in this work.

Publications: [Wiesler & Ney 11, Wiesler & Schlüter$^+$ 11]

**Evaluation of Hessian-Free optimization for large-scale cross-entropy training of DNNs**

The Hessian-free algorithm [Martens 10] has been tailored specifically for neural network training. In contrast to other second-order algorithms, it is based on a full second-order model of the objective function. In the original work [Martens 10], the algorithm has only been applied to small-scale tasks and has not been compared to stochastic gradient descent. In speech recognition, Hessian-free optimization has received attention, since [Kingsbury & Sainath$^+$ 12] achieved gains in wall-clock time and error rate by applying the algorithm to sequence-discriminative training. In this work, we evaluated the use of Hessian-free optimization for large-scale cross-entropy training. We found that the algorithm is not feasible for large-scale training of deep neural networks from scratch. With an appropriate initialization, Hessian-free can achieve lower training frame error rates as stochastic gradient descent, but these improvements did not carry over to unseen data. However, we could reduce the size of the model slightly with Hessian-free optimization without increase in error rate.

Publication: [Wiesler & Li$^+$ 13]

**Optimization of linear bottleneck networks from scratch using a novel stochastic second-order optimization algorithm**

We developed a novel stochastic second-order optimization algorithm and proved its convergence in a convex setting. The algorithm is motivated by our convergence analysis of log-linear training. The algorithm has negligible overhead in comparison to stochastic gradient descent and does not require additional tuning of hyperparameters.

Empirically, we showed that our proposed algorithm is capable of training deep neural networks with a linear bottleneck structure from scratch. Training and evaluation of the networks are accelerated due to the reduced size of the models. Further, we found that the linear bottleneck structure is a very effective regularization method. Across all considered tasks, we obtained consistent recognition error rate improvements with much smaller models.

Publication: [Wiesler & Richard$^+$ 14b]

**Investigations on sequence-discriminative training of neural networks**

Sequence-discriminative training of neural networks is a key technique in building state-of-the-art speech recognition systems. However, effective sequence training remains a complex engineering task and there is no common agreement on the best training configuration. Our experiments provide more empirical evidence on the best choice of the training criterion, training enhancements, and the optimization algorithm. Further, we extended our work on training linear bottleneck networks from scratch. A consistent gain due to the bottleneck structure persists after sequence-discriminative training.

In a joint work, sequence-discriminative training has also been show to be effective for recurrent neural networks with LSTM cells.

Publications: [Wiesler & Golik$^+$ 15, Voigtlaender & Doetsch$^+$ 15]

**Development of open source software for neural network training**

In the course of this thesis, a complex speech recognition software package has been extended by components for neural network training and hybrid NN-HMM recognition. The software package is freely available for academic use. The neural network implementation is flexible w.r.t. model structure, training criterion, and optimization algorithm. We demonstrated the efficiency of the software by an application to a real-world task and by comparing its performance with a well-proven neural network tool.

Publication: [Wiesler & Richard$^+$ 14a]

# Chapter 9

# Outlook

Our work could touch only a few issues in the emerging research area of neural networks in speech recognition. Possible future directions related to the findings in our work could include the following topics:

**Analysis of the objective function in neural network training**

Our work contributed to a better theoretical understanding of training convex log-linear models. With the resurgence of neural networks, a natural direction is the theoretical analysis of the objective function of multi-layer neural networks. Of course, this problem is much harder due to the non-convexity. First steps in this direction have been taken in [Dauphin & Pascanu$^+$ 14] and [Choromanska & Henaff$^+$ 15]. However, many questions remain open: How can (unrealistic) assumptions be weakened? Can the better understanding be used to actually improve training of neural networks?

**Model reduction**

With the training method introduced in our work, the size of deep neural networks can be reduced strongly without decrease in performance. For using deep neural networks in production, controlling model size is of great practical relevance. Another interesting approach to this problem proposed in literature is known as *model compression* or *knowledge distillation* [Bucila & Caruana$^+$ 06, Li & Zhao$^+$ 14, Hinton & Vinyals$^+$ 15], where a small model is learned with the outputs of a large model (or an ensemble of models) as targets. It would be interesting to compare this approach with ours or possibly to combine them.

**Training criteria**

In our experiments, we observed that sequence-discriminative training of neural networks provides substantial improvements over frame-discriminative training. Still, this technique requires a considerable engineering effort. An easier-to-use training criterion defined on sequence-level would be desirable. One interesting approach in this direction is the lattice-free *connectionist temporal classification* training [Graves & Fernández$^+$ 06, Sak & Senior$^+$ 15c].

Another interesting problem is how sequence-discriminative training can be made more robust to transcription errors. In practical applications, unsupervised or lightly

supervised in-domain data is typically abundant, but transcribing it carefully is expensive. In semi-supervised training, this data is transcribed automatically using a speech recognition system, cf. [Lamel & Gauvain$^+$ 02]. The data is then filtered based on confidences and heuristics, see e.g. [Kapralova & Alex$^+$ 14]. Can instead the training algorithm directly account for the unreliability of the transcriptions?

**Recurrent neural networks**

Recurrent neural networks are a wide topic, which has only been touched briefly in this work. In speech recognition, recurrent neural networks are of special interest, because they are sequential models. In principle, this enables integrating all components of a speech recognition system into the recurrent network. This direction has been explored for example in [Graves & Mohamed$^+$ 13, Hannun & Case$^+$ 14]. It is an open question, which classical speech recognition techniques are still required in such an integrated framework. For instance, context-dependent modeling of phonemes still gives a slight gain, but much less than in HMM-based speech recognition systems [Sak & Senior$^+$ 15b]. On the other hand, it has been shown that the improvements by sequence-discriminative training for recurrent networks are still in the same range as for feed-forward networks [Sak & Vinyals$^+$ 14, Voigtlaender & Doetsch$^+$ 15, Sak & Senior$^+$ 15a].

Since the recurrency makes optimization more difficult, it would also be interesting to study optimization in the context of recurrent networks, and in particular to adapt our proposed stochastic second-order optimization algorithm to recurrent networks.

# Appendix A

# Corpora and Systems

This appendix summarizes the information about the different tasks and baseline systems used in this work.

For the speech recognition experiments, we use a common acoustic front end consisting of MFCC features derived from a bank of 20 filters. The MFCC features are normalized by a vocal tract length normalization and augmented with a voicedness feature. Feature vectors from nine consecutive frames are concatenated and projected to a lower-dimensional feature space by means of an LDA.

The baseline acoustic models consist of Gaussian mixture hidden Markov models. A pronunciation is modeled by a sequence of triphones. Each triphone is modeled by a three-state HMM with left-to-right topology, with the exception of silence, which is modeled by a single-state HMM. The transition probabilities of the same type are tied, i.e. all loop, forward, and skip transitions have the same probability. Only the transition probabilities of the silence state are treated separately. Transitions leaving a word are penalized with an additional cost, the word penalty. The transition probabilities and the transition scale are not trained but manually tuned to minimize recognition error rate on the development data.

The Gaussian mixtures have a globally pooled diagonal covariance matrix. A phonetic decision tree is used to determine the tying of the emission model. Initial acoustic models are trained according to the maximum likelihood criterion using the EM algorithm with Viterbi approximation and a splitting procedure. The maximum likelihood model is then used as the initialization of an MPE training. The MPE objective is optimized using Rprop. The best MPE iteration is selected on the development set.

## A.1 Wall Street Journal

The WSJ0 corpus is a well-known corpus of American English read speech. It is available at the Linguistic Data Consortium (LDC93S6A). The corpus consists of business journal texts, read by American speakers. The training data consists of 15 hours of speech and the evaluation corpus of 0.7 hours of speech. Since the official WSJ0 corpus does not provide a development set, 410 sentences were extracted from ten new speakers of the North American Business (NAB) task and used as a development set. The task has a closed vocabulary of 5k words, i.e. all words in the development and evaluation corpus are known. The corpus statistics are summarized in Table A.1.

**Table A.1.** Details of the WSJ0 system

| | **Train** | **Dev** | **Eval** |
|---|---|---|---|
| Duration [h] | 15 | 0.7 | 0.7 |
| #Running words | 129k | 7k | 5k |
| OOV [%] | - | 0.0 | 0.0 |
| PPL | - | 58 | 53 |

For all experiments on the WSJ task, we used the default acoustic front end described above with an LDA target dimension of 33. The Gaussian mixture baseline system has phonetic decision tree with 1 501 leaves. The triphone context is limited by the word boundaries. The Gaussian mixture model has about 223k densities.

A trigram language model has been used for all recognitions. The maximum likelihood baseline system achieved a word error rate of 3.5 percent on the evaluation data. With a subsequent MPE training, the GHMM result has been improved to 3.2 percent word error rate.[1]

## A.2 Quaero English

The Quaero English corpus has been collected in the course of the Quaero project.[2] Quaero was a research project, funded by the French government, with the goal of automatic processing of general multimedia data in various languages. Speech recognition was one of the research topics of the project. The goal was to develop speech recognition systems for conversational data as it is typically found on the web, for example in podcasts. The baseline systems used in this work are simplified versions of the English evaluation systems, which have been built as part of RWTH Aachen's participation in the project [Sundermeyer & Nußbaum-Thom$^+$ 11]. RWTH's English evaluation system performed best among all submissions in the yearly evaluations from 2010 until the end of the project in 2013.

We distinguish between three different versions of the corpus, which are described below.

### A.2.1 Quaero 2010

The Quaero 2010 corpus contains the acoustic and text training data, which has been available in the 2010 evaluation campaign. The audio data consists of 103 hours of broadcast conversations. Details of the training and test sets are shown in Table A.2.

[1] Thanks to Markus Nußbaum-Thom for providing the baseline system for this task.

[2] `http://www.quaero.org`

**Table A.2.** Details of the Quaero English systems

| Corpus | | Duration [h] | #Words | OOV [%] | PPL |
|---|---|---|---|---|---|
| Quaero 2010 | Train | 103 | 855k | - | - |
| | Dev2010 | 3.3 | 39k | 0.3 | 151 |
| | Eval2010 | 3.7 | 41k | 0.3 | 149 |
| | Eval2011 | 3.3 | 35k | 0.4 | 157 |
| Quaero 2011 | Train-50h | 50 | 300k | - | - |
| | Train-Full | 234 | 1.7M | - | - |
| | Dev2011 | 3.7 | 41k | 0.3 | 132 |
| | Eval2011 | 3.3 | 35k | 0.4 | 144 |

The text training data contains newspaper articles, blog data, and the transcriptions of the acoustic training data. In total, it has a size of 3.1 billion running words.

We used the default feature extraction described above with an LDA target dimension of 45. The phonetic decision tree has 4 501 leaves. The Gaussian mixture model has roughly one million densities. The triphones include across-word context and word-boundary information.

The recognition lexicon consists of the 150k most frequent words in the text training data. The pronunciations are based on the BEEP pronunciation dictionary [Robinson 95]. Missing pronunciations were generated using automatic grapheme to phoneme conversion [Bisani & Ney 03]. The out of vocabulary (OOV) rate is below 0.4 percent on all test corpora.

A smoothed four-gram language model has been trained on the text training data. The language model has been obtained by pruning the large language model to 50.4 million $n$-grams.

The development corpus from the Quaero 2010 evaluation has been used for tuning the recognition system. The evaluation corpora from 2010 and 2011 have been used as test sets. The recognition results of the GHMM baseline system with maximum likelihood and MPE training are shown in Table A.3.

### A.2.2 Quaero 2011

The Quaero 2011 corpus builds on the Quaero 2010 corpus, but contains additional acoustic and text training data, which has been provided in the Quaero 2011 evaluation. After 2011, no additional training data has been provided within the project. Hence, the evaluation systems from 2011, 2012, and 2013 are all trained on the Quaero 2011 corpus. The total amount of acoustic data is 234 hours. The text data has a size of 3.7 billion running words.

In the Quaero 2011 systems, the feature extraction pipeline is the same as in the

**Table A.3.** Results of the GHMM baseline models with ML and MPE training on the Quaero English tasks

| Corpus | | WER [%] GHMM/ML | GHMM/MPE |
|---|---|---|---|
| Quaero 2010 | Dev2010 | 25.5 | 24.0 |
| | Eval2010 | 25.1 | 24.0 |
| | Eval2011 | 32.2 | 30.6 |
| Quaero 2011 / 50h | Dev2011 | 24.4 | 23.6 |
| | Eval2011 | 31.6 | 30.2 |
| Quaero 2011 / Full | Dev2011 | 22.1 | 20.4 |
| | Eval2011 | 28.6 | 26.2 |

default setup, except that the voicedness feature has not been used. The final feature dimension and the number of phonetic decision tree leaves is the same as in the Quaero 2010 system. The recognition lexicon is also the same as in the Quaero 2010 system. A new four-gram language model has been trained on the extended text training data. The pruned language model contains 47.7 million $n$-grams. Details on the amount of training data and the perplexity of the language model are given in Table A.2. The recognition systems were tuned on the Quaero 2011 development corpus, which coincides with the evaluation corpus from 2010. The evaluation corpus from 2011 has been used for testing.

At RWTH Aachen, a 50 hour subset of the training corpus has been selected as a rather small benchmark task for acoustic modeling. In this work, we performed experiments with acoustic models trained on both, the 50 hour subset and the complete training corpus. The recognition results of the GHMM baseline systems with maximum likelihood and MPE training are shown in Table A.3.

## A.3 Isolated Handwritten Digit Recognition

We consider two databases for handwritten digit recognition: the USPS and the MNIST database. Both are widely used reference datasets for handwritten digit recognition and allow for fast experiments due to their small size.

The USPS database consists of isolated images of handwritten digits taken from US mail envelopes. The images were normalized and scaled to 16×16 pixels. The task is to classify the images to the digits from 0 to 9, i.e. there are ten classes. The training set contains 7 291 images. The test set contains 2 007 images.

MNIST is another dataset for handwritten digit classification. It is one of the most widely used machine learning tasks. The dataset contains 70 000 images of handwritten

**Table A.4.** Details of the IAM dataset

| | Train | Development | Test |
|---|---|---|---|
| #Running words | 54k | 9k | 25k |
| #Lines | 6.1k | 0.9k | 2.7k |
| #Writers | 283 | 57 | 162 |

**Table A.5.** Details of the IAM handwriting recognition systems

| | IAM 2011 | | IAM 2014 | |
|---|---|---|---|---|
| | Validation | Test | Development | Test |
| OOV [%] | 3.9 | 3.4 | 3.9 | 3.4 |
| PPL | 394 | 440 | 378 | 231 |
| WER GHMM/ML [%] | 33.5 | 39.4 | 13.1 | 18.3 |
| WER GHMM/MPE [%] | 24.1 | 29.3 | 12.2 | 14.9 |

digits. 60 000 are intended for training and 10 000 for testing. The resolution of the images is 28×28 pixels.

## A.4 Offline Continuous Handwriting Recognition

The IAM database [Marti & Bunke 02] contains images of handwritten English text and provides four tasks. We apply our methods to the open-vocabulary line recognition task, i.e. the task is to recognize the text from line images only. Information from movement of the pen tip is not available. The vocabulary of the development and test set is unconstrained. The database is split into a training, a development, and a test set. The corpus statistics are summarized in Table A.4. All words are composed of 79 symbols, which consists of upper- and lowercase characters, punctuation, quotation marks, a special symbol for crossed out words, and a white-space model.

The Lancaster-Oslo-Bergen, Brown and Wellington corpus and the transcriptions of the training data are used for training the language model and selecting the lexicon, as proposed in [Bertolami & Bunke 08]. The total amount of text training data is 3.3 million running words. The recognition lexicon has 50k words. The language model is a Kneser-Ney smoothed trigram.

We consider two baseline systems on the IAM database. The first system, which we refer to as IAM 2011, is described in [Dreuw & Heigold$^+$ 11, Dreuw 12]. The second one includes recent improvements by our group and is based on the work described in [Kozielski & Doetsch$^+$ 13].

### A.4.1 The IAM 2011 system

In the IAM 2011 system, the feature preprocessing consists only of basic deslanting and size normalization, as it is commonly applied in handwriting recognition. An image slice is extracted at every position. Seven features in a sliding window are concatenated and projected to a 30-dimensional vector by a principal component analysis. The 78 characters are modeled by five-state left-to-right HMMs, resulting in 390 distinct states plus one state for the whitespace model. The emission probabilities of the generative baseline system are modeled by GMMs with 25k mixture components. The baseline results are shown in Table A.5. The maximum likelihood model achieves an error rate of 39.4 percent on the test data, which is comparable to the baseline results in [Bertolami & Bunke 08, España Boquera & Castro-Bleda$^+$ 11]. The results is improved to 29.3 percent word error rate with MPE training.

### A.4.2 The IAM 2014 system

The IAM 2014 system is strongly improved over the IAM 2011 system. The main reasons for the better performance are improvements in the feature preprocessing, a character length modeling, and a better language model trained on the same data [Kozielski & Doetsch$^+$ 13].

The features obtained from the improved preprocessing pipeline are 24-dimensional. With character length modeling, the number of distinct HMM states is 563. The statistics of the IAM 2014 system are shown in Table A.5. The word error rate of the IAM 2014 system is 14.9 percent on the test data.

Note that although the IAM 2014 system outperforms the IAM 2011 system by a large margin, the IAM 2011 system was competitive at the time of its development, compare Table 7.2.

## A.5 Overview of Experimental Results

Table A.6 gives an overview of our main experimental results on three tasks, which are used throughout this work: the IAM handwriting recognition task based on the 2014 system and the English broadcast conversations recognition task based on the 2011 system with the complete acoustic training data (Quaero/Full) and the fifty hour subset (Quaero/50h). The results are ordered by the type of the model and the training criterion. The Gaussian mixture model baselines were partly trained by other members of the group.

The results with log-linear models with polynomial features have been added for sake of completeness. The features were in this case third-order polynomials and the models have been trained using stochastic gradient descent by incrementally increasing the polynomial order. One can observe that the approach does not scale well to larger datasets. Details on the result with sparse features can be found in Chapter 3.

**Table A.6.** Overview of experimental results

| Method | | IAM | | Quaero/50h | | Quaero/Full | |
|---|---|---|---|---|---|---|---|
| Model | Training Criterion | Dev | Test | Dev | Test | Dev | Test |
| GMM | ML | 13.1 | 18.3 | 24.4 | 31.6 | 22.1 | 28.6 |
| | MMI | 12.5 | 15.4 | 24.1 | 31.2 | 21.7 | 28.1 |
| | MPE | 12.2 | 14.9 | 23.6 | 30.2 | 20.4 | 26.2 |
| LL/polynomial | CE | 12.5 | 16.5 | 24.2 | 31.8 | 22.2 | 29.1 |
| LL/sparse | CE | - | - | 23.6 | 30.6 | - | - |
| DNN | CE | 11.1 | 15.0 | 18.5 | 24.6 | 15.2 | 20.9 |
| | MMI/MPE | 11.0 | 14.2 | 17.5 | 23.3 | 14.1 | 19.2 |
| BN-DNN | CE | 11.1 | 14.6 | 18.0 | 23.7 | 14.9 | 19.9 |
| | MMI/MPE | 10.6 | 12.7 | 17.4 | 23.0 | 14.0 | 18.9 |
| LSTM-RNN | CE | 10.9 | 14.5 | - | - | - | - |
| | MMI | 10.4 | 13.5 | - | - | - | - |

The cross-entropy results with deep neural networks and bottleneck networks can be found in Chapter 6. The corresponding results using sequence-discriminative training (MMI for the handwriting task and MPE for the speech recognition tasks) are described in Chapter 7.

The results using LSTM-RNNs are given for comparison with the feed-forward network results on IAM, see Section 7.6.1. Details on these result can be found in [Voigtlaender 14] and the corresponding publication [Voigtlaender & Doetsch$^+$ 15].

# Appendix B

# Implementation of Neural Networks in the RASR Toolkit

At RWTH Aachen University, the speech recognition toolkit RASR has been developed and actively used for more than fourteen years. The toolkit has been made publicly available[1] in 2008 [Rybach & Gollan$^+$ 09] with the aim of facilitating research in this area and making scientific results reproducible. Within the scope of this work, RASR has been extended by a neural network module [Wiesler & Richard$^+$ 14a]. In particular, the MN-SGD optimization algorithm and the sequence training algorithms discussed in Chapter 6 and Chapter 7 are part of the toolkit.

In this chapter, we give an overview on our implementation of neural networks in the RASR toolkit and compare its performance with QuickNet, an efficient publicly available neural network software.

## B.1 Implementation

The support of neural networks in an open source toolkit is challenging for two reasons. First, neural networks are a highly active research topic. We therefore wrote our neural network code base as generic as possible in order to allow for rapid integration of new concepts. This is achieved by decoupling different software parts as much as possible while maintaining clean interfaces that allow to modify and extend single aspects of neural networks. Second, the computation time required for training neural networks and using them in recognition is critical. Since our goal is to apply our methods to real-life tasks, the implementation must be efficient. In our code, efficiency is achieved by supporting CPU multithreading and general purpose computing on GPUs. The GPU implementation makes use of the CUBLAS library and own CUDA kernels[2] . For CPU computations, highly optimized matrix libraries following the BLAS standard can be used, for example Intel MKL or ACML.[3]

In the following, the main features of the neural network implementation in RASR are described.

[1] see `http://www-i6.informatik.rwth-aachen.de/rwth-asr`

[2] see `http://www.nvidia.com/object/cuda_home_new.html`

[3] see `https://software.intel.com/en-us/intel-mkl` and `http://developer.amd.com/tools-and-sdks/archive/amd-core-math-library-acml/`

### B.1.1 Models

RASR supports feed-forward networks of a very general topology. The networks can have an arbitrary number of layers of different sizes and different activation functions. Each layer may have multiple inputs from other layers. In general, the network must be representable by a directed acyclic graph. In contrast, many other neural network implementations such as QuickNet only allow for a simple linear structure. The non-linear network structure enables for example the use of skip-connections or the joint training of hierarchical models [Veselý & Karafiát$^+$ 11]. The type of each layer can be chosen independently. Currently, RASR provides the commonly used activation functions sigmoid, tanh, linear and softmax as well as the more recent rectified linear units (ReLU) [Nair & Hinton 10].

### B.1.2 Frame-discriminative training

The structure of the neural network training code is similar to RASR's implementation of Gaussian mixture estimation. The `FeedForwardTrainer` has a `NeuralNetwork`, an `Estimator`, a `Statistics` object, and a `Regularizer`. Given a mini-batch of training examples, the trainer computes the sufficient statistics based on a forward and an error backpropagation pass, updates the statistics, and performs an estimation step. If available, all these operations are performed on a GPU without synchronization to the main memory, thus avoiding expensive communication.

The available frame-discriminative training criteria are *cross entropy*, *squared error*, and *binary divergence* [Solla & Levin$^+$ 88]. Optionally, regularization parameters w.r.t. $\ell_2$- or $\ell_1$-norm, momentum, and learning rates can be set individually for each layer.

While many neural network tools implement only stochastic gradient descent, RASR supports a variety of optimization algorithms. Currently, the supported estimators include basic SGD, SGD with momentum, and the MN-SGD algorithm developed in this work. In addition, batch estimation with gradient descent and Rprop [Riedmiller & Braun 93] is possible.

Almost all training components can be configured independently. In particular, we do not only allow "natural pairings" of training criteria and output-layer, e.g. softmax with cross-entropy or identity with squared-error as in QuickNet. More exotic combinations like softmax-outputs with squared-error criterion are possible as well [Golik & Doetsch$^+$ 13].

A difficulty in the implementation of stochastic optimization algorithms is that they require i.i.d. training examples, which is simulated by shuffling the training data. For relatively small datasets, simply all training examples can be loaded into main memory and then accessed in random order. For larger datasets, we use a twofold randomization. We load as many training utterances as possible in random order into a buffer, and then shuffle the buffer on frame-level.

**Table B.1.** Evaluation of QuickNet and RASR

| | WER [%] | | | |
|---|---|---|---|---|
| | QuickNet learning rates | | RASR learning rates | |
| **Training using** | Dev | Test | Dev | Test |
| QuickNet | 19.6 | 26.2 | 19.4 | 25.9 |
| RASR | 19.8 | 25.7 | 19.2 | 25.4 |

### B.1.3 Sequence-discriminative training

The available sequence-discriminative criteria include MMI and MPE, optionally with cross-entropy smoothing and frame rejection heuristic as discussed in Chapter 7. The lattice computations required for computing the error signal at the output layer are performed on CPU. All neural network computations are performed on a GPU, if available.

All optimization algorithms, which are available for frame-discriminative training, can be applied in sequence-discriminative training as well. Stochastic optimization algorithms are applied on utterance level.

### B.1.4 Recognition

RASR comes with a dynamic decoder that is based on the history-conditioned lexical tree approach [Ney & Ortmanns 00]. The generic code structure of RASR makes it easy to implement a hybrid NN-HMM model in the decoder. The decoder uses an abstract `FeatureScorer` for computing all required scores. The neural network feature scorer performs a forward pass of the network and converts the state posteriors to likelihood scores. In case of a softmax output layer, the expensive softmax activation function is not evaluated, because the normalization term is a constant in the search problem and can therefore be discarded. The feature scorer supports batching, i.e. multiple features are processed at once, which strongly speeds up matrix multiplications.

### B.1.5 Feature extraction

RASR provides a variety of signal analysis and preprocessing methods. Amongst others, RASR supports MFCC, PLP, Gammatone, and MRASTA [Hermansky & Fousek 05] features.

As described in Section B.1.2, stochastic optimization algorithms require that the features are loaded into a buffer and shuffled. Typical features for speech applications are obtained by stacking several consecutive frames in a sliding window. Buffering the windowed features directly thus increases the memory requirements proportional to the window size, which is typically in the order of 10 to 20. Instead, RASR can buffer only the central frames and construct the windowed features on-the-fly when generating the mini-batch.

**Table B.2.** Runtime analysis of RASR and QuickNet

| Model | Hardware | Implementation | Time/Epoch [m] | Speedup |
|---|---|---|---|---|
| 1x2048 | GPU | QuickNet | 36.0 | |
| | | RASR | 17.1 | 2.1 |
| | CPU | QuickNet | 616.1 | |
| | | RASR | 187.0 | 3.3 |
| 6x2048 | GPU | QuickNet | 58.2 | |
| | | RASR | 37.1 | 1.8 |
| | CPU | QuickNet | 1773.3 | |
| | | RASR | 549.3 | 3.2 |

## B.2 Experimental Comparison with QuickNet

In this section, we present experimental results which demonstrate the usefulness of RASR's neural network implementation on real-world tasks. We compare RASR with QuickNet in order to verify the implementation.

Experiments have been performed on the 50 hour subset of the English Quaero corpus, which is described in Section A.2.2. The experimental setup is the same as in Chapter 6, except that we did not include the three states of the garbage model in the neural network training. We evaluate two types of neural networks: a *shallow* neural network with one hidden layer of 2 048 sigmoid units and a *deep* network with six hidden layers of the same size. The number of network outputs is 4 498. All networks were trained according to the cross-entropy criterion using stochastic gradient descent with a mini-batch size of 512. The deep network has been initialized using layerwise pre-training.

As mentioned in Chapter 6, RASR uses a modification of the QuickNet learning rate schedule *Newbob*. For a fair comparison we ran RASR and QuickNet with both learning rate configurations. We aimed at keeping the QuickNet and RASR setups comparable at all levels. We used exactly the same training data and feature extraction for both implementations. All recognitions were performed with RASR using the same search parameters.

Table B.1 shows the word error rates of RASR and QuickNet in experiments with deep neural networks. Although both implementations use the same algorithms, it is important to compare their performance, because the numerical stability of the code and implementation details can impact the results. The results in Table B.1 confirm that the RASR learning rates are slightly better than the default Newbob learning rates in QuickNet. Further, in three out of four experiments, the model trained using RASR achieved better results. The differences may not be significant, but we can conclude that the results obtained with RASR are competitive.

In order to be able to train large models for real-world tasks and to tune systems, an efficient implementation is required. Table B.2 summarizes the wall-clock time per training epoch for the shallow and the deep network with RASR and QuickNet. The runtime analysis has been performed on an an Nvidia Tesla K20c and a twelve-core AMD Opteron (2.3GHz). It can be seen that RASR is faster than QuickNet by a factor of 2.3 for the shallow network. The difference between both implementations is less pronounced for the deep network, but RASR is still 1.8 times faster than QuickNet.

Comparing RASR's GPU and CPU implementations, the training on GPU is faster by a factor of 11 for the shallow network and 15 for the deep network. The gain of using a GPU may be smaller on a faster CPU. Nevertheless, there is a clear advantage of using a GPU for training deep neural networks.

## B.3 Summary

In this chapter, we gave a detailed description of the neural network implementation in our publicly available speech recognition toolkit RASR. The implementation served as the basis for our research on neural networks. By making it publicly available to the scientific community, we aim at allowing for a better reproducibility of our results and facilitating research in this area in general.

# Appendix C

# Detailed Calculations

## C.1 Chapter 6

**Lemma C.1.** *Assume the notation of Section 6.2. The gradient of*

$$\widetilde{\mathcal{F}} = \mathcal{F} \circ \phi^{-1} \,, \tag{C.1}$$

*is*

$$\begin{aligned} \nabla_W \widetilde{\mathcal{F}}(W,b) &= \nabla_W \mathcal{F}\big(\phi^{-1}(W,b)\big) + a \cdot \nabla_b \mathcal{F}\big(\phi^{-1}(W,b)\big)^\top \,, & \text{(C.2)} \\ \nabla_b \widetilde{\mathcal{F}}(W,b) &= \nabla_b \mathcal{F}\big(\phi^{-1}(W,b)\big) \,. & \text{(C.3)} \end{aligned}$$

*Proof.* Recall that $\phi^{-1}$ is defined as

$$\phi^{-1}(W,b) = (W, b + W^\top a) \,. \tag{C.4}$$

In the following, we use the shortcut

$$(\widetilde{W}, \widetilde{b}) = \phi^{-1}(W,b) \,. \tag{C.5}$$

We begin with the partial derivatives with respect to the weights:

$$\begin{aligned} \frac{\partial \widetilde{\mathcal{F}}}{\partial w_{i,j}}(W,b) =& \sum_{i',j'} \frac{\partial \mathcal{F}}{\partial \tilde{w}_{i',j'}}(\widetilde{W},\widetilde{b}) \cdot \frac{\partial \tilde{w}_{i',j'}}{\partial w_{i,j}}(W,b) & \text{(C.6)} \\ &+ \sum_{j'} \frac{\partial \mathcal{F}}{\partial \tilde{b}_{j'}}(\widetilde{W},\widetilde{b}) \cdot \frac{\partial \tilde{b}_{j'}}{\partial w_{i,j}}(W,b) & \text{(C.7)} \\ =& \sum_{i',j'} \frac{\partial \mathcal{F}}{\partial \tilde{w}_{i',j'}}(\widetilde{W},\widetilde{b}) \cdot \delta_{i',i} \cdot \delta_{j',j} & \text{(C.8)} \\ &+ \sum_{j'} \frac{\partial \mathcal{F}}{\partial \tilde{b}_{j'}}(\widetilde{W},\widetilde{b}) \cdot \delta_{j',j} \cdot a_i & \text{(C.9)} \\ =& \frac{\partial \mathcal{F}}{\partial \tilde{w}_{i,j}}(\widetilde{W},\widetilde{b}) + \frac{\partial \mathcal{F}}{\partial \tilde{b}_j}(\widetilde{W},\widetilde{b}) \cdot a_i \,. & \text{(C.10)} \end{aligned}$$

Similarly, the partial derivatives with respect to the biases are:

$$\frac{\partial \widetilde{\mathcal{F}}}{\partial b_j}(W,b) = \sum_{i',j'} \frac{\partial \mathcal{F}}{\partial \tilde{w}_{i',j'}}(\widetilde{W},\widetilde{b}) \cdot \frac{\partial \tilde{w}_{i',j'}}{\partial b_j}(W,b) \tag{C.11}$$

$$+ \sum_{j'} \frac{\partial \mathcal{F}}{\partial \tilde{b}_{j'}}(\widetilde{W},\widetilde{b}) \cdot \frac{\partial \tilde{b}_{j'}}{\partial b_j}(W,b) \tag{C.12}$$

$$= \sum_{j'} \frac{\partial \mathcal{F}}{\partial \tilde{b}_{j'}}(\widetilde{W},\widetilde{b}) \cdot \delta_{j',j} \tag{C.13}$$

$$= \frac{\partial \mathcal{F}}{\partial \tilde{b}_j}(\widetilde{W},\widetilde{b}) \ . \tag{C.14}$$

□

# Appendix D

# Symbols and Acronyms

In this appendix, all relevant acronyms and mathematical symbols which are used in this thesis are defined for convenience.

## D.1 Symbols

### General Mathematical Notation

| | |
|---|---|
| $a_1^n$ | short-hand notation for a finite sequence, $a_1^n = (a_1, \dots, a_n)$ |
| $A \otimes B$ | Kronecker product of two matrices $A$ and $B$ |
| $A^\top$ | transpose of a matrix $A$ |
| $\delta_{i,j}$ | Kronecker delta |
| $\mathbf{E}$ | expectation of a random variable |
| exp | natural exponential function |
| $\nabla f$ | gradient of a function $f$ |
| $\frac{\partial f}{\partial x}$ | partial derivative of a function $f$ w.r.t. a variable $x$ |
| $\nabla^2 f$ | Hessian matrix of a function $f$ |
| $I_n$ | identity matrix of size $n$ |
| inf | infimum |
| $\kappa(A)$ | condition number of a positive definite matrix $A$ |
| log | natural logarithm |
| $\mathbf{N}$ | set of natural numbers (including zero) |
| $\|x\|_p$ | $p$-norm of a vector $x$, $p \geq 1$ |
| $\|x\|$ | Euclidean norm of a vector $x$, i.e. $\|x\| = \|x\|_2$ |
| $\mathcal{N}(\cdot\|\mu, \Sigma)$ | normal distribution with mean $\mu \in \mathbf{R}^n$ and covariance matrix $\Sigma \in \mathbf{R}^{n\times n}$ |
| $\mathbf{R}$ | set of real numbers |
| $\mathbf{R}^n$ | $n$-dimensional vector space over $\mathbf{R}$ |
| $\mathbf{R}^{m\times n}$ | set of $m \times n$-dimensional real matrices |
| $\sigma(A)$ | spectrum of matrix $A$ |
| $\vartheta_j(A)$ | $j$-th eigenvalue of a real and symmetric matrix $A$ in ascending order |

### Special Symbols

| | |
|---|---|
| $a, a'$ | emission model label, typically given by the phonetic tree |
| $a_1^T, \mathbf{a}$ | sequence of emission model labels, e.g. $a_1^T = (a_1, \ldots, a_T)$ |
| $\mathcal{A}$ | state tying, maps HMM state to emission model label |
| $\mathcal{B}$ | mini-batch |
| $b^{(l)}$ | bias vector of layer $l$ in a neural network |
| $c, \bar{c}$ | class index |
| $C$ | number of classes |
| $c_{a,l}$ | mixture weight of a GMM, where $a$ is the emission model label and $l$ the index of the mixture component |
| $\mathcal{C}_{\mathbf{w}}[\boldsymbol{\pi}]$ | word-pronunciation cost function, depending on reference $\mathbf{w}$ |
| $D$ | dimension of observation space |
| $\eta, \eta_i$ | learning rate |
| $\mathcal{F}$ | objective function (training criterion), to be minimized |
| $\mathcal{F}^{(\mathrm{CE})}$ | cross-entropy (CE) objective function (with hard labels) |
| $\mathcal{F}^{(\mathrm{MMI})}$ | sequence-discriminative maximum mutual information (MMI) objective function |
| $\mathcal{F}^{(\mathrm{MBR})}$ | minimum Bayes risk (MBR) objective function |
| $\mathcal{F}^{(\mathrm{MPE})}$ | minimum phone error (MPE) objective function |
| $f$ | feature function of log-linear model |
| $J$ | number of features of log-linear model |
| $\lambda_c$ | parameters of log-linear model for class $c$ |
| $\Lambda$ | parameters of log-linear model |
| $\mathcal{L}$ | word lattice |
| $\mathcal{L}(w_1^N)$ | sub-lattice of $\mathcal{L}$, containing only paths with word sequence $w_1^N$ |
| $L_a$ | number of mixture components of the GMM for label $a$ |
| $\mathrm{Lev}(\cdot, \cdot)$ | Levenshtein distance |
| $\mu_{a,l}$ | mean of a GMM, where $a$ is the label and $l$ the index of the mixture component |
| $\pi_1^N, \boldsymbol{\pi}$ | word-pronunciation sequence, path in a lattice |
| $r$ | speech segment index |
| $R$ | number of speech segments |
| $\mathcal{R}$ | regularization term |
| $s$ | HMM state |
| $s_1^T, \mathbf{s}$ | HMM state sequence, e.g. $s_1^T = (s_1, \ldots, s_T)$ |
| $\Sigma$ | covariance matrix of a Gaussian distribution |
| $\sigma$ | non-linearity in a neural network |
| $\theta$ | parameters of a neural network |
| $W^{(l)}$ | weight matrix of layer $l$ in a neural network |
| $w, v$ | word |
| $w_1^N, \mathbf{w}$ | word sequence, e.g. $w_1^N = (w_1, \ldots, w_N)$ |

| | |
|---|---|
| $\omega(\boldsymbol{\pi})$ | function mapping from a word-pronunciation sequence to the corresponding word sequence |
| $x, x'$ | feature vector |
| $x_1^T, \mathbf{x}$ | sequence of feature vectors, e.g. $x_1^T = (x_1, \ldots, x_T)$ |

## D.2 Acronyms

| | |
|---|---|
| AM | acoustic model |
| BFGS | Broyden-Fletcher-Goldfarb-Shanno algorithm |
| BN | bottleneck |
| CE | cross-entropy |
| CG | conjugate gradient |
| CMLLR | constrained maximum likelihood linear regression |
| CPU | central processing unit |
| CRF | conditional random field |
| CTC | connectionist temporal classification |
| CV | cross-validation |
| DNN | deep neural network |
| EM | expectation maximization |
| FER | frame error rate |
| GHMM | Gaussian-mixture-HMM |
| GIS | generalized iterative scaling |
| GMM | Gaussian mixture model |
| GPU | graphics processing unit |
| HCRF | hidden CRF |
| HF | Hessian-free |
| HMM | hidden Markov model |
| i.i.d. | independent and identically distributed |
| IAM | handwriting database provided by the Institut für Informatik und angewandte Mathematik, University of Bern |
| IIS | improved iterative scaling |
| L-BFGS | limited-memory BFGS |
| LDA | linear discriminant analysis |
| LL | log-linear |

| | |
|---|---|
| LM | language model |
| LSTM | long short-term memory |
| LVCSR | large vocabulary continuous speech recognition |
| MBR | minimum Bayes risk |
| MEMM | maximum entropy Markov model |
| MFCC | Mel-frequency cepstral coefficients |
| ML | maximum likelihood |
| MMI | maximum mutual information |
| MN-SGD | mean-normalized stochastic gradient descent |
| MNIST | Mixed National Institute of Standards and Technology |
| MPE | minimum phone error |
| MPFE | minimum phone frame error |
| NN | neural network |
| oLBFGS | online L-BFGS |
| OOV | out of vocabulary |
| OW-Rprop | orthant-wise Rprop |
| PLP | perceptual linear prediction |
| PPL | perplexity |
| RASR | RWTH Aachen University Automatic Speech Recognition Toolkit |
| RBM | restricted Boltzmann machine |
| RNN | recurrent neural network |
| Rprop | resilient backpropagation |
| RWTH | Rheinish-Westfälische Technische Hochschule |
| SGD | stochastic gradient descent |
| sMBR | state-level minimum Bayes risk |
| SMDR3 | Microsoft Short Message Dictation Database |
| SVD | singular value decomposition |
| SVM | support vector machine |
| USPS | U.S. Postal Service |

| | |
|---|---|
| WER | word error rate |
| WFST | weighted finite state transducer |
| WSJ | Wall Street Journal |

# Appendix E

# Publications and Joint Work

This thesis is based on several publications, which have been co-authored by other members of the RWTH Aachen speech recognition group. All publications have been supervised and revised by Prof. Dr.-Ing. H. Ney and Dr. rer. nat. R. Schlüter (with the exception of [Wiesler & Li$^+$ 13]). In the following, the author's contribution is described as required by the university's dissertation rules.

**[Wiesler & Nußbaum$^+$ 09]** S. Wiesler and G. Heigold jointly developed the idea of scaling log-liner models to large-scale tasks using sparse clustering features. M. Nußbaum-Thom contributed the alignment for the from-scratch training. The software implementation of the hybrid log-linear approach is by G. Heigold. S. Wiesler performed the experiments, the analysis of the results, and wrote the manuscript.

**[Wiesler & Richard$^+$ 11], [Wiesler & Schlüter$^+$ 12]** S. Wiesler contributed the ideas for the publications, in particular the OW-Rprop optimization algorithm and the acceleration technique described in [Wiesler & Schlüter$^+$ 12], and implemented the methods. The experiments have been performed jointly by S. Wiesler and A. Richard. The manuscripts have been written by S. Wiesler.

**[Wiesler & Richard$^+$ 13]** S. Wiesler designed the publication, implemented the framework for stochastic optimization, and wrote the manuscript. A. Richard implemented specific optimization algorithms. The experiments were performed jointly by S. Wiesler and A. Richard. The manuscript has been written by S. Wiesler.

**[Heigold & Schlüter$^+$ 12]** S. Wiesler wrote the sections Optimization, Extended Baum-Welch, Rprop, and Hybrid Approach Using Log-Linear Models. The results on the WSJ task (Table 6) are by S. Wiesler.

**[Wiesler & Ney 11], [Wiesler & Schlüter$^+$ 11]** S. Wiesler contributed the idea for the convergence analysis, derived the theory, performed the experiments, and wrote the manuscripts.

**[Wiesler & Li$^+$ 13]** S. Wiesler contributed the idea for applying the HF algorithm and implemented the algorithm in the Microsoft DNN training tool. The experiments on MNIST were performed by S. Wiesler. The speech recognition experiments were per-

formed jointly by S. Wiesler and J. Xue. The analysis of the results and the manuscript are by S. Wiesler. The work was supervised by J. Li and J. Xue.

**[Wiesler & Richard$^+$ 14a]** S. Wiesler designed the implementation described in the publication and realized a major part of it. A. Richard, who supported the implementation, has been supervised by S. Wiesler. The experimental evaluation has been carried out by A. Richard and P. Golik. The sections Introduction, Implementation, and Conclusion have been written by S. Wiesler.

**[Wiesler & Richard$^+$ 14b]** S. Wiesler derived the proposed algorithm, contributed the idea to apply it to linear bottleneck networks, and the idea for the convergence proof. The experiments and the implementation, which builds on [Wiesler & Richard$^+$ 14a], were performed by A. Richard under S. Wiesler's supervision. S. Wiesler wrote the manuscript, which has been revised by all co-authors.

**[Wiesler & Golik$^+$ 15]** S. Wiesler realized the implementation described in the publication, performed the systematic series of experiments and their analysis, and wrote the manuscript. The integration of the code into the official RASR release package has been performed by P. Golik. S. Wiesler wrote the manuscript, which has been revised by all co-authors.

**[Voigtlaender & Doetsch$^+$ 15]** The publication has been designed jointly by P. Doetsch, P. Voigtländer, and S. Wiesler. The sequence training code has been written by S. Wiesler, the Theano-based LSTM software by P. Doetsch, and the interface by P. Voigtländer. The sequence training experiments with feed-forward networks have been performed by S. Wiesler and the experiments with LSTM's by P. Voigtländer. The manuscript has been written jointly by P. Voigtländer, P. Doetsch, and S. Wiesler.

# List of Figures

# List of Tables

# Bibliography

[Aizerman & Braverman$^+$ 64] A. Aizerman, E.M. Braverman, L. Rozoner: Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, Vol. 25, pp. 821–837, 1964.

[Allauzen & Mohri 03] C. Allauzen, M. Mohri: Efficient algorithms for testing the twins property. *J. of Automata, Languages, and Combinatorics*, Vol. 8, No. 2, 2003.

[Alleva & Huang$^+$ 96] P. Alleva, X.D. Huang, M.Y. Hwang: Improvements on the pronunciation prefix tree search organization. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 133–136, Atlanta, GA, USA, May 1996.

[Anastasiadis & Magoulas$^+$ 05] A.D. Anastasiadis, G.D. Magoulas, M.N. Vrahatis: New globally convergent training scheme based on the resilient propagation algorithm. *Neurocomputing*, Vol. 64, pp. 253–270, 2005.

[Andrew & Gao 07] G. Andrew, J. Gao: Scalable training of $L^1$-regularized log-linear models. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 33–40, Corvallis, OR, USA, June 2007.

[Aubert 02] X.L. Aubert: An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech and Language*, Vol. 16, No. 1, pp. 89–114, 2002.

[Bahl & Brown$^+$ 86] L. Bahl, P. Brown, P. de Souza, R. Mercer: Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 49–52, Tokyo, Japan, May 1986.

[Bahl & Jelinek$^+$ 83] L.R. Bahl, F. Jelinek, R.L. Mercer: A maximum likelihood approach to continuous speech recognition. *IEEE Trans. on Pattern Anal. and Mach. Intell. (PAMI)*, Vol. 5, pp. 179–190, March 1983.

[Baker 75] J.K. Baker: Stochastic modeling for automatic speech understanding. In D.R. Reddy, editor, *Speech Recognition*, pp. 512–542. Academic Press, New York, NY, USA, 1975.

[Bakis 76] R. Bakis: Continuous speech recognition via centisecond acoustic states. *The J. of the Acoustical Soc. of America*, Vol. 59, No. S1, pp. S97–S97, 1976.

[Baum 72] L.E. Baum: An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. In O. Shisha, editor, *Inequalities*, Vol. 3, pp. 1–8. Academic Press, New York, NY, 1972.

[Baum & Eagon 67] L.E. Baum, J. Eagon: An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the Amer. Math. Soc.*, Vol. 73, No. 3, pp. 360–363, 1967.

[Bayes 63] T. Bayes: An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFRS. *Philosophical Transactions of the Royal Society*, Vol. 53, pp. 370–418, 1763. reprinted in Biometrika, vol. 45, no. 3/4, pp. 293–315, December 1958.

[Bellman 57] R.E. Bellman: Dynamic Programming. Princeton University Press, Princeton, NJ, USA, 1957.

[Bengio & Ducharme$^+$ 01] Y. Bengio, R. Ducharme, P. Vincent: A neural probabilistic language model. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 932–938, Denver, CO, USA, Dec. 2001.

[Berger & Pietra$^+$ 96] A.L. Berger, V.J.D. Pietra, S.A.D. Pietra: A maximum entropy approach to natural language processing. *Computational Linguistics*, Vol. 22, No. 1, pp. 39–71, 1996.

[Bertolami & Bunke 08] R. Bertolami, H. Bunke: Hidden Markov model-based ensemble methods for offline handwritten text line recognition. *Pattern Recognition*, Vol. 41, No. 11, pp. 3452–3460, 2008.

[Bisani & Ney 03] M. Bisani, H. Ney: Multigram-based grapheme-to-phoneme conversion for LVCSR. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 933–936, Geneva, Switzerland, Sept. 2003.

[Bishop 06] C.M. Bishop: *Pattern Recognition and Machine Learning*, Vol. 1. Springer, 2006.

[Bordes & Bottou$^+$ 09] A. Bordes, L. Bottou, P. Gallinari: SGD-QN: careful quasi-Newton stochastic gradient descent. *J. of Mach. Learning Research*, Vol. 10, pp. 1737–1754, 2009.

[Boser & Guyon$^+$ 92] B.E. Boser, I.M. Guyon, V.N. Vapnik: A training algorithm for optimal margin classifiers. In *Proc. of the Ann. Workshop on Computational Learning Theory (COLT)*, pp. 144–152, Pittsburgh, PA, USA, July 1992.

[Bottou 91] L. Bottou: *Une approche théorique de l'apprentissage connexionniste: applications à la reconnaissance de la parole.* Ph.D. thesis, Université de Paris XI, Orsay, France, 1991.

[Bottou 98] L. Bottou: Online learning and stochastic approximations. In D. Saad, editor, *Online Learning in Neural Networks*, pp. 9–43. Cambridge University Press, Cambridge, UK, 1998.

[Bottou & Bousquet 08] L. Bottou, O. Bousquet: The tradeoffs of large scale learning. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 161–168, Vancouver, Canada, Dec. 2008.

[Bottou & Lin 07] L. Bottou, C.J. Lin: Support vector machine solvers. In L. Bottou, O. Chapelle, D. DeCoste, J. Weston, editors, *Large Scale Kernel Machines*, pp. 301–320. MIT Press, Cambridge, MA, USA, 2007.

[Bourlard & Morgan 94] H.A. Bourlard, N. Morgan: *Connectionist speech recognition: a hybrid approach*, Vol. 247. Kluwer Academic Publishers, 1994.

[Boyd & Vandenberghe 09] S. Boyd, L. Vandenberghe: *Convex Optimization*. Cambridge University Press, 2009.

[Boyd & Vandenberghe 14] S. Boyd, L. Vandenberghe: Subgradients. Stanford University Lecture slides. Retrieved September 10, 2014 from `http://stanford.edu/class/ee364b/lectures/subgradients_notes.pdf`, May 2014.

[Bridle 90] J.S. Bridle: Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pp. 227–236. Springer, 1990.

[Bryson & Denham$^+$ 63] A.E. Bryson, W.F. Denham, S.E. Dreyfus: Optimal programming problems with inequality constraints. *AIAA journal*, Vol. 1, No. 11, pp. 25–44, 1963.

[Bucila & Caruana$^+$ 06] C. Bucila, R. Caruana, A. Niculescu-Mizil: Model compression. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 535–541, Philadelphia, PA, USA, Aug. 2006.

[Burr 86] D.J. Burr: A neural network digit recognizer. In *Proc. of the IEEE Int. Conf. of Syst., Man and Cybernetics*, pp. 1621–1625, Atlanta, GA, USA, Oct. 1986.

[Chapelle 07] O. Chapelle: Training a support vector machine in the primal. *Neural Computation*, Vol. 19, No. 5, pp. 1155–1178, 2007.

[Choromanska & Henaff$^+$ 15] A. Choromanska, M. Henaff, M. Mathieu, G.B. Arous, Y. LeCun: The loss surfaces of multilayer networks. In *Proc. of the Int. Conf. on Artificial Intell. and Stat. (AISTATS)*, pp. 192–204, San Diego, CA, USA, May 2015.

[Cohn 07] T.A. Cohn: *Scaling conditional random fields for natural language processing*. Ph.D. thesis, University of Melbourne, Melbourne, Australia, Jan. 2007.

[Cortes & Vapnik 95] C. Cortes, V. Vapnik: Support-vector networks. *Mach. Learning*, Vol. 20, No. 3, pp. 273–297, 1995.

[Dahl & Ranzato$^+$ 10] G.E. Dahl, M. Ranzato, A. Mohamed, G.E. Hinton: Phone recognition with the mean-covariance restricted Boltzmann machine. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 469–477, Vancouver, Canada, Dec. 2010.

[Dahl & Yu$^+$ 12] G.E. Dahl, D. Yu, L. Deng, A. Acero: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. on Audio, Speech, and Language Process.*, Vol. 20, No. 1, pp. 30–42, 2012.

[Darroch & Ratcliff 72] J.N. Darroch, D. Ratcliff: Generalized iterative scaling for log-linear models. *Ann. of the Math. Stat.*, Vol. 43, No. 5, pp. 1470–1480, 1972.

[Dauphin & Pascanu$^+$ 14] Y.N. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, Y. Bengio: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 2933–2941, Montreal, Canada, Dec. 2014.

[Davis & Mermelstein 80] S. Davis, P. Mermelstein: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoust., Speech, and Signal Process.*, Vol. ASSP-28, No. 4, pp. 357 – 366, Aug. 1980.

[Dean & Corrado$^+$ 12] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q.V. Le, M.Z. Mao, M. Ranzato, A.W. Senior, P.A. Tucker, K. Yang, A.Y. Ng: Large scale distributed deep networks. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 1232–1240, Lake Tahoe, NV, USA, Dec. 2012.

[Dempster & Laird$^+$ 77] A. Dempster, N. Laird, D. Rubin: Maximum likelihood from incomplete data via the EM algorithm. *J. of the Roy. Statistical Soc. Series B (Methodological)*, Vol. 39, No. B, pp. 1 – 38, 1977.

[Do & Artières 10] T.M.T. Do, T. Artières: Neural conditional random fields. In *Proc. of the Int. Conf. on Artificial Intell. and Stat. (AISTATS)*, pp. 177–184, Sardinia, Italy, May 2010.

[Doetsch & Kozielski$^+$ 14] P. Doetsch, M. Kozielski, H. Ney: Fast and robust training of recurrent neural networks for offline handwriting recognition. In *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 279–284, Crete, Greece, Sept. 2014.

[Dreuw 12] P. Dreuw: *Probabilistic sequence models for image sequence processing and recognition.* Ph.D. thesis, RWTH Aachen University, Aachen, Germany, April 2012.

[Dreuw & Heigold$^+$ 11] P. Dreuw, G. Heigold, H. Ney: Confidence and margin-based MMI/MPE discriminative training for offline handwriting recognition. *Int. J. on Document Anal. and Recognition*, Vol. 14, No. 3, pp. 273–288, April 2011.

[Duchi & Hazan+ 11] J. Duchi, E. Hazan, Y. Singer: Adaptive subgradient methods for online learning and stochastic optimization. *J. of Mach. Learning Research*, Vol. 12, pp. 2121–2159, 2011.

[Eide & Gish 96] E. Eide, H. Gish: A parametric approach to vocal tract length normalization. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 346–348, Atlanta, GA, USA, May 1996.

[Esling 10] J.H. Esling: *Phonetic Notation*, pp. 678–702. Blackwell Publishing Ltd., 2010.

[España Boquera & Castro-Bleda+ 11] S. España Boquera, M. Castro-Bleda, J. Gorbe-Moya, F. Zamora-Martinez: Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Trans. on Pattern Anal. and Mach. Intell. (PAMI)*, Vol. 33, No. 4, pp. 767–779, April 2011.

[Evermann & Woodland 00] G. Evermann, P. Woodland: Posterior probability decoding, confidence estimation and system combination. In *Proc. of the NIST Speech Transcription Workshop*, College Park, MD, USA, 2000.

[Fisher 36] R.A. Fisher: The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, Vol. 7, No. 179-188, 1936.

[Fontaine & Ris+ 97] V. Fontaine, C. Ris, J.M. Boite: Nonlinear discriminant analysis for improved speech recognition. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, Vol. 2, pp. 2071–2074, Rhodos, Greece, Sept. 1997.

[Fosler-Lussier & Morris 08] E. Fosler-Lussier, J. Morris: Crandem systems: conditional random field acoustic models for hidden Markov models. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 4049–4052, Las Vegas, NV, USA, March 2008.

[Gales 98] M.J. Gales: Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, Vol. 12, No. 2, pp. 75–98, 1998.

[Geman & Bienenstock+ 92] S. Geman, E. Bienenstock, R. Doursat: Neural networks and the bias/variance dilemma. *Neural Computation*, Vol. 4, No. 1, pp. 1–58, 1992.

[Generet & Ney+ 95] M. Generet, H. Ney, F. Wessel: Extensions to absolute discounting for language modeling. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 1245–1248, Madrid, Spain, Sept. 1995.

[Gibson & Hain 06] M. Gibson, T. Hain: Hypothesis spaces for minimum Bayes risk training in large vocabulary speech recognition. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 2406–2409, Pittsburgh, PA, USA, Sept. 2006.

[Glorot & Bengio 10] X. Glorot, Y. Bengio: Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the Int. Conf. on Artificial Intell. and Stat. (AISTATS)*, pp. 249–256, Sardinia, Italy, May 2010.

[Gold & Lippmann$^+$ 87] B. Gold, R. Lippmann, M. Malpass: Some neural net recognition results on isolated words. In *Proc. of Int. Conf. on Neural Networks (ICNN)*, San Diego, CA, USA, June 1987.

[Golik & Doetsch$^+$ 13] P. Golik, P. Doetsch, H. Ney: Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1756–1760, Lyon, France, Aug. 2013.

[Graves & Fernández$^+$ 06] A. Graves, S. Fernández, F.J. Gomez, J. Schmidhuber: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 369–376, Pittsburgh, PA, USA, June 2006.

[Graves & Liwicki$^+$ 09] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber: A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. on Pattern Anal. and Mach. Intell. (PAMI)*, Vol. 31, No. 5, pp. 855–868, 2009.

[Graves & Mohamed$^+$ 13] A. Graves, A. Mohamed, G.E. Hinton: Speech recognition with deep recurrent neural networks. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 6645–6649, Vancouver, Canada, May 2013.

[Grezl & Karafiát$^+$ 07] F. Grezl, M. Karafiát, S. Kontár, J. Cernockỳ: Probabilistic and bottle-neck features for LVCSR of meetings. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 757–760, Honolulu, HI, USA, April 2007.

[Griewank 12] A. Griewank: Who invented the reverse mode of differentiation? *Documenta Matematica*, Vol. Optimization Stories, pp. 389–400, 2012.

[Gunawardana & Mahajan$^+$ 05] A. Gunawardana, M. Mahajan, A. Acero, J.C. Platt: Hidden conditional random fields for phone classification. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 1117–1120, Lisbon, Portugal, Sept. 2005.

[Häb-Umbach & Ney 92] R. Häb-Umbach, H. Ney: Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 13–16, San Francisco, CA, USA, March 1992.

[Hannun & Case$^+$ 14] A.Y. Hannun, C. Case, J. Casper, B.C. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A.Y. Ng: Deep speech: scaling up end-to-end speech recognition, 2014. arXiv preprint `http://arxiv.org/abs/1412.5567`.

[Hart & Nilsson$^+$ 68] P.E. Hart, N.J. Nilsson, B. Raphael: A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Syst. Sci. and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968.

[Heigold 10] G. Heigold: *A log-linear discriminative modeling framework for speech recognition*. Ph.D. thesis, RWTH Aachen University, Aachen, Germany, June 2010.

[Heigold & Deselaers$^+$ 08] G. Heigold, T. Deselaers, R. Schlüter, H. Ney: Modified MMI/MPE: a direct evaluation of the margin in speech recognition. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 384–391, Helsinki, Finland, June 2008.

[Heigold & McDermott$^+$ 14] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior, M. Bacchiani: Asynchronous stochastic optimization for sequence training of deep neural networks. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 5624–5628, Florence, Italy, May 2014.

[Heigold & Ney$^+$ 11] G. Heigold, H. Ney, P. Lehnen, T. Gass, R. Schlüter: Equivalence of generative and log-linear models. *IEEE Trans. on Audio, Speech, and Language Process.*, Vol. 19, No. 5, pp. 1138–1148, July 2011.

[Heigold & Rybach$^+$ 09] G. Heigold, D. Rybach, R. Schlüter, H. Ney: Investigations on convex optimization using log-linear HMMs for digit string recognition. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 216–219, Brighton, UK, Sept. 2009.

[Heigold & Schlüter$^+$ 12] G. Heigold, R. Schlüter, H. Ney, S. Wiesler: Discriminative training for automatic speech recognition: modeling, criteria, optimization, implementation, and performance. *IEEE Signal Process. Mag.*, Vol. 29, No. 6, pp. 58–69, Nov. 2012.

[Heigold & Wiesler$^+$ 10] G. Heigold, S. Wiesler, M. Nußbaum-Thom, P. Lehnen, R. Schlüter, H. Ney: Discriminative HMMS, log-linear models, and CRFS: What is the difference? In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 5546–5549, Dallas, TX, USA, March 2010.

[Heigold & Zweig$^+$ 09] G. Heigold, G. Zweig, X. Li, P. Nguyen: A flat direct model for speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 3861–3864, Taipei, Taiwan, April 2009.

[Hermansky 90] H. Hermansky: Perceptual linear predictive (PLP) analysis of speech. *J. of the Acoustical Soc. of America*, Vol. 87, No. 4, pp. 1738 – 1752, June 1990.

[Hermansky & Ellis$^+$ 00] H. Hermansky, D.P.W. Ellis, S. Sharma: Tandem connectionist feature extraction for conventional HMM systems. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 1635–1638, Istanbul, Turkey, June 2000.

[Hermansky & Fousek 05] H. Hermansky, P. Fousek: Multi-resolution RASTA filtering for TANDEM-based ASR. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 361–364, Lisbon, Portugal, Sept. 2005.

[Hestenes & Stiefel 52] M.R. Hestenes, E. Stiefel: Methods of conjugate gradients for solving linear systems. *J. of Research of the Nat. Inst. of Standards*, Vol. 49, No. 6, pp. 409–436, 1952.

[Hifny & Renals$^+$ 05] Y. Hifny, S. Renals, N.D. Lawrence: A hybrid Maxent/HMM based ASR system. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 3017–3020, Lisbon, Portugal, Sept. 2005.

[Hifny & Renals 09] Y. Hifny, S. Renals: Speech recognition using augmented conditional random fields. *IEEE Trans. on Audio, Speech, and Language Process.*, Vol. 17, No. 2, pp. 354–365, 2009.

[Hinton & Osindero$^+$ 06] G. Hinton, S. Osindero, Y.W. Teh: A fast learning algorithm for deep belief nets. *Neural Computation*, Vol. 18, No. 7, pp. 1527–1554, 2006.

[Hinton & Salakhutdinov 06] G.E. Hinton, R.R. Salakhutdinov: Reducing the dimensionality of data with neural networks. *Science*, Vol. 313, No. 5786, pp. 504–507, 2006.

[Hinton & Vinyals$^+$ 15] G. Hinton, O. Vinyals, J. Dean: Distilling the knowledge in a neural network, 2015. arXiv preprint `http://arxiv.org/abs/1503.02531v1`.

[Hon & Lee 91] H.W. Hon, K.F. Lee: Recent progress in robust vocabulary-independent speech recognition. In *Proc. of the DARPA Speech and Natural Language Process. Workshop*, pp. 258–263, Pacific Grove, CA, USA, Feb. 1991.

[Horn & Johnson 94] R. Horn, C. Johnson: *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[Horn & Johnson 05] R. Horn, C. Johnson: *Matrix Analysis*. Cambridge University Press, 2005.

[Huang & Zweig$^+$ 14] Z. Huang, G. Zweig, B. Dumoulin: Cache based recurrent neural network language model inference for first pass speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 6404 – 6408, Florence, Italy, May 2014.

[Igel & Hüsken 03] C. Igel, M. Hüsken: Empirical evaluation of the improved Rprop learning algorithms. *Neurocomputing*, Vol. 50, pp. 105–123, 2003.

[Jaakkola & Haussler 99] T. Jaakkola, D. Haussler: Exploiting generative models in discriminative classifiers. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 487–493, Denver, CO, USA, Dec. 1999.

[Jaitly & Hinton 13] N. Jaitly, G.E. Hinton: Vocal tract length perturbation (VTLP) improves speech recognition. In *Proc. of the ICML Workshop on Deep Learning for Audio, Speech and Language Process.*, Atlanta, GA, USA, June 2013.

[Jaitly & Nguyen$^+$ 12] N. Jaitly, P. Nguyen, A.W. Senior, V. Vanhoucke: Application of pretrained deep neural networks to large vocabulary speech recognition. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, Portland, OR, USA, Sept. 2012.

[Jaynes 57] E.T. Jaynes: Information theory and statistical mechanics. *Physical Review*, Vol. 106, pp. 620–630, May 1957.

[Jelinek 69] F. Jelinek: A fast sequential decoding algorithm using a stack. *IBM J. of Research and Develop.*, Vol. 13, pp. 675–685, Nov. 1969.

[Johnson 04] D. Johnson: QuickNet, Speech Group at ICSI, Berkeley. `http://www.icsi.berkeley.edu/Speech/qn.html`.

[Kaiser & Horvat$^+$ 00] J. Kaiser, B. Horvat, Z. Kacic: A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Proc. of the Int. Conf. on Spoken Language Process. (ICSLP)*, pp. 887–890, Beijing, China, Oct. 2000.

[Kaiser & Horvat$^+$ 02] J. Kaiser, B. Horvat, Z. Kacic: Overall risk criterion estimation of hidden Markov model parameters. *Speech Commun.*, Vol. 38, No. 3–4, pp. 383–398, 2002.

[Kanthak & Ney$^+$ 02] S. Kanthak, H. Ney, M. Riley, M. Mohri: A comparison of two LVR search optimization techniques. In *Proc. of the Int. Conf. on Spoken Language Process. (ICSLP)*, pp. 1309–1312, Denver, CO, USA, Sept. 2002.

[Kanthak & Schütz$^+$ 00] S. Kanthak, K. Schütz, H. Ney: Using SIMD instructions for fast likelihood calculation in LVCSR. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 1531–1534, Istanbul, Turkey, June 2000.

[Kapralova & Alex$^+$ 14] O. Kapralova, J. Alex, E. Weinstein, P.J. Moreno, O. Siohan: A big data approach to acoustic model training corpus selection. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 2083–2087, Singapore, Sept. 2014.

[Katz 87] S.M. Katz: Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. on Speech and Audio Process.*, Vol. 35, pp. 400–401, March 1987.

[Kingsbury 09] B. Kingsbury: Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 3761–3764, Taipei, Taiwan, April 2009.

[Kingsbury & Sainath$^+$ 12] B. Kingsbury, T.N. Sainath, H. Soltau: Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free Optimization. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 10–13, Portland, OR, USA, Sept. 2012.

[Kira & Rendell 92] K. Kira, L.A. Rendell: A practical approach to feature selection. In *Proc. of the Int. Workshop on Mach. Learning*, pp. 249–256, Aberdeen, Great Britain, 1992.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved backing-off for m-gram language modeling. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 181–184, Detroit, MI, USA, May 1995.

[Kononenko 94] I. Kononenko: Estimating attributes: analysis and extensions of RELIEF. In *Proc. of the European Conf. on Mach. Learning (ECML)*, pp. 171–182, Catania, Italy, April 1994.

[Kozielski & Doetsch$^+$ 13] M. Kozielski, P. Doetsch, H. Ney: Improvements in RWTH's system for off-line handwriting recognition. In *Proc. of the Int. Conf. on Document Anal. and Recognition (ICDAR)*, pp. 935–939, Washington DC, USA, Aug. 2013.

[Kozielski & Rybach$^+$ 13] M. Kozielski, D. Rybach, S. Hahn, R. Schlüter, H. Ney: Open vocabulary handwriting recognition using combined word-level and character-level language models. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 8257–8261, Vancouver, Canada, May 2013.

[Kubo & Wiesler$^+$ 11] Y. Kubo, S. Wiesler, R. Schlüter, H. Ney, S. Watanabe, A. Nakamura, T. Kobayashi: Subspace pursuit method for kernel-log-linear models. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 4500–4503, Prague, Czech Republic, May 2011.

[Kuo & Gao 06] H.K.J. Kuo, Y. Gao: Maximum entropy direct models for speech recognition. *IEEE Trans. on Audio, Speech, and Language Process.*, Vol. 14, No. 3, pp. 873–881, 2006.

[Lafferty & McCallum$^+$ 01] J.D. Lafferty, A. McCallum, F.C.N. Pereira: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 282–289, Williamstown, MA, USA, July 2001.

[Lamel & Gauvain$^+$ 02] L. Lamel, J. Gauvain, G. Adda: Lightly supervised and unsupervised acoustic model training. *Computer Speech and Language*, Vol. 16, No. 1, pp. 115–129, 2002.

[Layton & Gales 06] M.I. Layton, M.J. Gales: Augmented statistical models for speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 129–.132, Tolouse, France, May 2006.

[LeCun & Bottou$^+$ 98] Y. LeCun, L. Bottou, G.B. Orr, K.R. Müller: Efficient BackProp. In *Neural Networks: Tricks of the Trade*, pp. 9–50. Springer, 1998.

[LeCun & Kanter$^+$ 90] Y. LeCun, I. Kanter, S. Solla: Second order properties of error surfaces: Learning time and generalization. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 918–924, Denver, CO, USA, Nov. 1990.

[LeCun & Kanter$^+$ 91] Y. LeCun, I. Kanter, S. Solla: Eigenvalues of covariance matrices: application to neural-network learning. *Physical Review*, Vol. 66, No. 18, pp. 2396–2399, May 1991.

[Levenshtein 66] V.I. Levenshtein: Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Soviet Physics*, Vol. 10, pp. 707 – 710, 1966.

[Li & Zhao$^+$ 14] J. Li, R. Zhao, J. Huang, Y. Gong: Learning small-size DNN with output-distribution-based criteria. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1910–1914, Singapore, Sept. 2014.

[Liu & Nocedal 89] D. Liu, J. Nocedal: On the limited memory BFGS method for large-scale optimization. *Math. Programming*, Vol. 45, No. 1, pp. 503–528, 1989.

[Lowerre 76] B. Lowerre: *A comparative performance analysis of speech understanding systems.* Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1976.

[Luenberger & Ye 08] D. Luenberger, Y. Ye: *Linear and Nonlinear Programming.* Springer Verlag, 2008.

[Macherey & Ney 03] W. Macherey, H. Ney: A comparative study on maximum entropy and discriminative training for acoustic modeling in automatic speech recognition. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 493–496, Geneva, Switzerland, Sept. 2003.

[Malouf 02] R. Malouf: A comparison of algorithms for maximum entropy parameter estimation. In *Proc. of the Conf. on Computational Natural Language Learning (CoNLL)*, pp. 1–7, Aug. 2002.

[Martens 10] J. Martens: Deep learning via Hessian-free optimization. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 735–742, Haifa, Israel, June 2010.

[Marti & Bunke 02] U. Marti, H. Bunke: The IAM-database: an English sentence database for offline handwriting recognition. *Int. J. on Document Anal. and Recognition*, Vol. 5, No. 1, pp. 39–46, 2002.

[McCallum & Freitag$^+$ 00] A. McCallum, D. Freitag, F.C. Pereira: Maximum entropy Markov models for information extraction and segmentation. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 591–598, Stanford, CA, USA, July 2000.

[McDermott & Heigold$^+$ 14] E. McDermott, G. Heigold, P. Moreno, A. Senior, M. Bacchiani: Asynchronous stochastic optimization for sequence training of deep neural networks: towards big data. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1224–1228, Singapore, Sept. 2014.

[Mikolov & Karafiát$^+$ 10] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, S. Khudanpur: Recurrent neural network based language model. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1045–1048, Makuhari, Japan, Aug. 2010.

[Minka 01] T. Minka: Algorithms for maximum-likelihood logistic regression. Technical report, Carnegie Mellon University, 2001.

[Mohamed & Dahl$^+$ 09] A.r. Mohamed, G. Dahl, G. Hinton: Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, Whistler, Canada, Dec. 2009.

[Mohamed & Dahl$^+$ 12] A.r. Mohamed, G.E. Dahl, G. Hinton: Acoustic modeling using deep belief networks. *IEEE Trans. on Audio, Speech, and Language Process.*, Vol. 20, No. 1, pp. 14–22, 2012.

[Mohri & Riley 97] M. Mohri, M. Riley: Weighted determinization and minimization for large vocabulary speech recognition. In *eurospeech1997*, pp. 131–134, Rhodes, Greece, Sept. 1997.

[Moré & Sorensen 83] J.J. Moré, D.C. Sorensen: Computing a trust region step. *SIAM J. on Scientific and Statistical Computing*, Vol. 4, No. 3, pp. 553–572, 1983.

[Morgan & Bourlard 89] N. Morgan, H. Bourlard: Generalization and parameter estimation in feedforward nets: some experiments. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 630–637, Denver, CO, USA, Nov. 1989.

[Nair & Hinton 10] V. Nair, G.E. Hinton: Rectified linear units improve restricted Boltzmann machines. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 807–814, Haifa, Israel, June 2010.

[Ney 84] H. Ney: The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Trans. on Speech and Audio Process.*, Vol. 32, No. 2, pp. 263–271, April 1984.

[Ney 90] H. Ney: Acoustic modeling of phoneme units for continuous speech recognition. In *Proc. of the European Signal Process. Conf. (EUSIPCO)*, pp. 65–72, Barcelona, Spain, Sept. 1990.

[Ney & Aubert 96] H. Ney, X. Aubert: Dynamic programming search strategies: From digit strings to large vocabulary word graphs. In C.H. Lee, F. Soong, K. Paliwal, editors, *Automatic Speech and Speaker Recognition*, Vol. 355 of *The Kluwer International Series in Engineering and Computer Science*, pp. 385–411. Springer US, 1996.

[Ney & Essen$^+$ 94] H. Ney, U. Essen, R. Kneser: On structuring probabilistic dependencies in language modeling. *Computer Speech and Language*, Vol. 2, No. 8, pp. 1–38, 1994.

[Ney & Häb-Umbach$^+$ 92] H. Ney, R. Häb-Umbach, B.H. Tran, M. Oerder: Improvements in beam search for 10000-word continuous speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 9–12, San Francisco, CA, USA, March 1992.

[Ney & Mergel$^+$ 87] H. Ney, D. Mergel, A. Noll, A. Paeseler: A data-driven organization of the dynamic programming beam search for continuous speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 833–836, Dallas, TX, USA, April 1987.

[Ney & Ortmanns 00] H. Ney, S. Ortmanns: Progress in dynamic programming search for LVCSR. *Proc. of the IEEE*, Vol. 88, No. 8, pp. 1224–1240, Aug. 2000.

[Ng 04] A.Y. Ng: Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, 78, Banff, Canada, June 2004.

[Nocedal & Wright 06] J. Nocedal, S.J. Wright: Numerical Optimization. *Numerical optimization*, Vol. 2, 2006.

[Nolden & Ney$^+$ 11] D. Nolden, H. Ney, R. Schlüter: Exploiting sparseness of backing-off language models for efficient look-ahead in LVCSR. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 4684–4687, Prague, Czech Republic, May 2011.

[Nolden & Schlüter$^+$ 11] D. Nolden, R. Schlüter, H. Ney: Acoustic look-ahead for more efficient decoding in LVCSR. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 893–896, Florence, Italy, Aug. 2011.

[Normandin 96] Y. Normandin: Maximum Mutual Information Estimation of Hidden Markov Models. In C.H. Lee, F.K. Soong, K.K. Paliwal, editors, *Automatic Speech and Speaker Recognition: Advanced Topics*, pp. 57–81. Springer, 1996.

[Notay 90] Y. Notay: Solving positive (semi)definite linear systems by preconditioned iterative methods. In *Preconditioned Conjugate Gradient Methods*, Vol. 1457 of *Lecture Notes in Mathematics*, pp. 105–125. Springer, 1990.

[Odell & Valtchev$^+$ 94] J.J. Odell, V. Valtchev, P.C. Woodland, S.J. Young: A one-pass decoder design for large vocabulary recognition. In *Proc. of the ARPA Spoken Language Technology Workshop*, pp. 405–410, Plainsboro, NJ, USA, March 1994.

[Ortmanns & Eiden$^+$ 98] S. Ortmanns, A. Eiden, H. Ney: Improved lexical tree search for large vocabulary recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 817–820, Seattle, WA, USA, May 1998.

[Ortmanns & Ney 95] S. Ortmanns, H. Ney: An experimental study of the search space for 20000-word speech recognition. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 901–904, Madrid, Spain, Sept. 1995.

[Ortmanns & Ney$^+$ 97] S. Ortmanns, H. Ney, X. Aubert: A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language*, Vol. 11, No. 1, pp. 43–72, Jan. 1997.

[Ortmanns & Ney 00] S. Ortmanns, H. Ney: Look-ahead techniques for fast beam search. *Computer Speech and Language*, Vol. 14, No. 1, pp. 15–32, Jan. 2000.

[Palaz & Collobert$^+$ 13] D. Palaz, R. Collobert, M. Magimai.-Doss: Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1766–1770, Lyon, France, Aug. 2013.

[Paul 91] D.B. Paul: Algorithms for an optimal $A^*$ search and linearizing the search in the stack decoder. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 693–696, Toronto, Canada, May 1991.

[Pearlmutter 94] B.A. Pearlmutter: Fast exact multiplication by the Hessian. *Neural Computation*, Vol. 6, No. 1, pp. 147–160, 1994.

[Peeling & Moore$^+$ 86] S. Peeling, R. Moore, M. Tomlinson: The multi-layer perceptron as a tool for speech pattern processing research. In *Proc. of the Conf. on Speech and Hearing*, pp. 307–314, Edinburgh, Great Britain, Nov. 1986.

[Pham & Bluche$^+$ 14] V. Pham, T. Bluche, C. Kermorvant, J. Louradour: Dropout improves recurrent neural networks for handwriting recognition. In *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, pp. 285–290, Crete, Greece, Sept. 2014.

[Plaut & Nowlan$^+$ 86] D. Plaut, S. Nowlan, G. Hinton: Experiments on learning by back propagation. Technical Report CMU-CS-86-126, Carnegie Mellon University, 1986.

[Povey 04] D. Povey: *Discriminative training for large vocabulary speech recognition.* Ph.D. thesis, Cambridge University, Cambridge, UK, 2004.

[Povey & Kingsbury+ 05] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, G. Zweig: fMPE: discriminatively trained features for speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 961–964, Philadelphia, PA, USA, March 2005.

[Povey & Woodland 02] D. Povey, P.C. Woodland: Minimum phone error and I-smoothing for improved discriminative training. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 105 – 108, Orlando, FL, USA, May 2002.

[Rabiner & Juang 86] L. Rabiner, B.H. Juang: An introduction to hidden Markov models. *IEEE Trans. on Acoust., Speech, and Signal Process.*, Vol. 3, No. 1, pp. 4–16, 1986.

[Rabiner & Schafer 79] L.R. Rabiner, R.W. Schafer: *Digital Processing of Speech Signals*. Prentice-Hall Signal Processing Series, Englewood Cliffs, NJ, 1979.

[Ragni & Gales 11] A. Ragni, M.J. Gales: Derivative kernels for noise robust ASR. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 119–124, Waikoloa, HI, USA, Dec. 2011.

[Raiko & Valpola+ 12] T. Raiko, H. Valpola, Y. LeCun: Deep learning made easier by linear transformations in perceptrons. In *Proc. of the Int. Conf. on Artificial Intell. and Stat. (AISTATS)*, pp. 924–932, La Palma, Spain, April 2012.

[Ratnaparkhi et al. 96] A. Ratnaparkhi et al.: A maximum entropy model for part-of-speech tagging. In *Proc. of the Conf. on Empirical Methods in Natural Language Process. (EMNLP)*, pp. 133–142, Philadelphia, PA, USA, May 1996.

[Riedmiller & Braun 93] M. Riedmiller, H. Braun: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of Int. Conf. on Neural Networks (ICNN)*, pp. 586–591, San Francisco, CA, USA, March 1993.

[Robbins & Monro 51] H. Robbins, S. Monro: A stochastic approximation method. *Ann. of the Math. Stat.*, Vol. 22, No. 3, pp. 400–407, 1951.

[Robbins & Siegmund 71] H.E. Robbins, D.O. Siegmund: A convergence theorem for non negative almost supermartingales and some applications. In *Proc. Symp. Optimizing Methods in Statistics*, pp. 233–257, Ohio, USA, March 1971.

[Robinson 95] T. Robinson: BEEP - The British English Example Pronunciation Dictionary. ftp://svr-ftp.eng.cam.ac.uk/comp.speech/dictionaries/, 1995.

[Rosenfeld 94] R. Rosenfeld: *Adaptive statistical language modeling: a maximum entropy approach*. Ph.D. thesis, Carnegie Mellon University, 1994.

[Rudin 76] W. Rudin: *Principles of mathematical analysis*. McGraw-Hill New York, 1976.

[Rumelhart & Hinton$^+$ 86] D.E. Rumelhart, G.E. Hinton, R.J. Williams: Learning representations by back-propagating errors. *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

[Rybach 14] D. Rybach: *Investigations on search methods for speech recognition using weighted finite-state transducers.* Ph.D. thesis, RWTH Aachen University, Aachen, Germany, April 2014.

[Rybach & Gollan$^+$ 09] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, H. Ney: The RWTH Aachen University open source speech recognition system. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 2111–2114, Brighton, UK, Sept. 2009.

[Sainath & Kingsbury$^+$ 11] T.N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novák, A. rahman Mohamed: Making deep belief networks effective for large vocabulary continuous speech recognition. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 30–35, Waikoloa, HI, USA, Dec. 2011.

[Sainath & Kingsbury$^+$ 13] T.N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran: Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, Vancouver, Canada, May 2013.

[Sak & Senior$^+$ 15a] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, J. Schalkwyk: Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 4280–4284, Brisbane, Australia, April 2015.

[Sak & Senior$^+$ 15b] H. Sak, A.W. Senior, K. Rao, F. Beaufays: Fast and accurate recurrent neural network acoustic models for speech recognition, 2015. arXiv preprint `http://arxiv.org/abs/1507.06947`.

[Sak & Senior$^+$ 15c] H. Sak, A.W. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, J. Schalkwyk: Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 4280–4284, Brisbane, Australia, April 2015.

[Sak & Vinyals$^+$ 14] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, M. Mao: Sequence discriminative distributed training of long short-term memory recurrent neural networks. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1209–1213, Singapore, Sept. 2014.

[Salakhutdinov & Roweis$^+$ 03] R. Salakhutdinov, S. Roweis, Z. Ghahramani: On the convergence of bound optimization algorithms. In *Proc. of the Conf. on Uncertainty in Artificial Intell. (UAI)*, pp. 509–516, Acapulco, Mexico, Aug. 2003.

[Saon & Soltau 14] G. Saon, H. Soltau: A comparison of two optimization techniques for sequence discriminative training of deep neural networks. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 5567–5571, Florence, Italy, May 2014.

[Schlüter 00] R. Schlüter: *Investigations on discriminative training criteria.* Ph.D. thesis, RWTH Aachen University, Aachen, Germany, Sept. 2000.

[Schlüter & Müller$^+$ 99] R. Schlüter, B. Müller, F. Wessel, H. Ney: Interdependence of language models and discriminative training. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 119–122, Keystone, CO, USA, Dec. 1999.

[Schmidhuber 14] J. Schmidhuber: Deep learning in neural networks: an overview. Technical Report IDSIA-03-14 / arXiv:1404.7828v1 [cs.NE], The Swiss AI Lab IDSIA, April 2014.

[Schraudolph 02] N.N. Schraudolph: Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, Vol. 14, No. 7, pp. 1723–1738, 2002.

[Schraudolph & Yu$^+$ 07] N.N. Schraudolph, J. Yu, S. Günter: A stochastic quasi-Newton method for online convex optimization. In *Proc. of the Int. Conf. on Artificial Intell. and Stat. (AISTATS)*, pp. 436–443, San Juan, Puerto Rico, March 2007.

[Schwenk 07] H. Schwenk: Continuous space language models. *Computer Speech and Language*, Vol. 21, No. 3, pp. 492–518, 2007.

[Seide & Fu$^+$ 14] F. Seide, H. Fu, J. Droppo, G. Li, D. Yu: 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1058–1062, Singapore, Sept. 2014.

[Seide & Li$^+$ 11a] F. Seide, G. Li, X. Chen, D. Yu: Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 24–29, Waikoloa, HI, USA, Dec. 2011.

[Seide & Li$^+$ 11b] F. Seide, G. Li, D. Yu: Conversational speech transcription using context-dependent deep neural networks. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 437–440, Florence, Italy, Aug. 2011.

[Senior & Heigold$^+$ 13] A. Senior, G. Heigold, M. Ranzato, K. Yang: An empirical study of learning rates in deep neural networks for speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 6724–6728, Vancouver, Canada, May 2013.

[Sha & Pereira 03] F. Sha, F.C.N. Pereira: Shallow parsing with conditional random fields. In *Proc. of the Human Language Technology Conf. of the North Amer. Chapter of the Assoc. for Computational Linguistics (HLT-NAACL)*, pp. 134–141, Edmonton, Canada, May 2003.

[Shalev-Shwartz & Singer$^+$ 07] S. Shalev-Shwartz, Y. Singer, N. Srebro: Pegasos: primal estimated sub-gradient solver for SVM. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 807–814, Corvallis, OR, USA, June 2007.

[Sixtus 03] A. Sixtus: *Across-word phoneme models for large vocabulary continuous speech recognition.* Ph.D. thesis, RWTH Aachen, Jan. 2003.

[Sixtus & Ortmanns 99] A. Sixtus, S. Ortmanns: High quality word graphs using forward-backward pruning. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 593–596, Phoenix, AZ, USA, March 1999.

[Solla & Levin$^+$ 88] S.A. Solla, E. Levin, M. Fleisher: Accelerated learning in layered neural networks. *Complex Systems*, Vol. 2, No. 6, pp. 625–639, Dec. 1988.

[Steihaug 83] T. Steihaug: The conjugate gradient method and trust regions in large scale optimization. *SIAM J. on Numerical Analysis*, Vol. 20, No. 3, pp. 626–637, 1983.

[Strom 15] N. Strom: Scalable distributed DNN training using commodity GPU cloud computing. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1488–1492, Dresden, Germany, Sept. 2015.

[Su & Li$^+$ 13] H. Su, G. Li, D. Yu, F. Seide: Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 6664–6668, Vancouver, Canada, May 2013.

[Sundermeyer & Nußbaum-Thom$^+$ 11] M. Sundermeyer, M. Nußbaum-Thom, S. Wiesler, C. Plahl, A. El-Desoky Mousa, S. Hahn, D. Nolden, R. Schlüter, H. Ney: The RWTH 2010 Quaero ASR evaluation system for English, French, and German. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 2212–2215, Prague, Czech Republic, May 2011.

[Sundermeyer & Schlüter$^+$ 12] M. Sundermeyer, R. Schlüter, H. Ney: LSTM Neural Networks for Language Modeling. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 194–197, Portland, OR, USA, Sept. 2012.

[Sunehag & Trumpf$^+$ 09] P. Sunehag, J. Trumpf, S.V.N. Vishwanathan, N.N. Schraudolph: Variable metric stochastic approximation theory. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pp. 560–566, Clearwater Beach, FL, USA, April 2009.

[Sutton & McCallum 12] C. Sutton, A. McCallum: An introduction to conditional random fields. *Found. and Trends in Mach. Learning*, Vol. 4, No. 4, pp. 267–373, 2012.

[Tibshirani 96] R. Tibshirani: Regression shrinkage and selection via the lasso. *J. of the Roy. Statistical Soc. Series B (Methodological)*, Vol. 58, No. 1, pp. 267–288, 1996.

[Tsochantaridis & Joachims$^+$ 05] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun: Large margin methods for structured and interdependent output variables. *J. of Mach. Learning Research*, Vol. 6, pp. 1453–1484, 2005.

[Tüske & Golik$^+$ 14] Z. Tüske, P. Golik, R. Schlüter, H. Ney: Acoustic modeling with deep neural networks using raw time signal for LVCSR. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 890–894, Singapore, Sept. 2014.

[Valtchev & Odell$^+$ 97] V. Valtchev, J. Odell, P.C. Woodland, S.J. Young: MMIE training of large vocabulary recognition systems. *Speech Commun.*, Vol. 22, No. 4, pp. 303–314, 1997.

[van Dalen & Ragni$^+$ 13] R.C. van Dalen, A. Ragni, M.J.F. Gales: Efficient decoding with generative score-spaces using the expectation semiring. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 7619–7623, Vancouver, Canada, May 2013.

[Veselý & Ghoshal$^+$ 13] K. Veselý, A. Ghoshal, L. Burget, D. Povey: Sequence-discriminative training of deep neural networks. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 2345–2349, Lyon, France, Aug. 2013.

[Veselý & Karafiát$^+$ 11] K. Veselý, M. Karafiát, F. Grézl: Convolutive bottleneck network features for LVCSR. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 42–47, Waikoloa, HI, USA, Dec. 2011.

[Vishwanathan & Schraudolph$^+$ 06] S. Vishwanathan, N. Schraudolph, M. Schmidt, K. Murphy: Accelerated training of conditional random fields with stochastic gradient methods. In *Proc. of the Int. Conf. on Mach. Learning (ICML)*, pp. 969–976, Pittsburgh, PA, USA, June 2006.

[Viterbi 67] A. Viterbi: Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. on Inform. Theory*, Vol. 13, pp. 260–269, 1967.

[Voigtlaender 14] P. Voigtlaender: Sequence training of recurrent neural networks for Handwriting Recognition. Bachelor's thesis, RWTH Aachen University, 2014.

[Voigtlaender & Doetsch$^+$ 15] P. Voigtlaender, P. Doetsch, S. Wiesler, R. Schlüter, H. Ney: Sequence-discriminative training of recurrent neural networks. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 2100–2104, Brisbane, Australia, April 2015.

[Waibel & Hanazawa+ 89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang: Phoneme recognition using time-delay neural networks. *IEEE Trans. on Acoust., Speech, and Signal Process.*, Vol. 37, No. 3, pp. 328–339, Mar 1989.

[Wallach 03] H. Wallach: Efficient training of conditional random fields. In *Proc. Ann. Computational Linguistics United Kingdom Research Colloquium*, Edinburgh, Great Britain, 2003.

[Wang & Gales+ 07] L. Wang, M.J.F. Gales, P.C. Woodland: Unsupervised training for mandarin broadcast news and conversation transcription. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 353–356, Honolulu, HI, USA, April 2007.

[Welling & Kanthak+ 99] L. Welling, S. Kanthak, H. Ney: Improved methods for vocal tract normalization. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, Vol. 2, pp. 761–764. IEEE, March 1999.

[Wiesler & Golik+ 15] S. Wiesler, P. Golik, R. Schlüter, H. Ney: Investigations on sequence training of neural networks. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 4565–4569, Brisbane, Australia, April 2015.

[Wiesler & Li+ 13] S. Wiesler, J. Li, J. Xue: Investigations on Hessian-free optimization for cross-entropy training of deep neural networks. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 3317–3321, Lyon, France, Aug. 2013.

[Wiesler & Ney 11] S. Wiesler, H. Ney: A convergence analysis of log-linear training. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pp. 657–665, Granada, Spain, Dec. 2011.

[Wiesler & Nußbaum+ 09] S. Wiesler, M. Nußbaum, G. Heigold, R. Schlüter, H. Ney: Investigations on features for log-linear acoustic models in continuous speech recognition. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 52–57, Merano, Italy, Dec. 2009.

[Wiesler & Richard+ 11] S. Wiesler, A. Richard, Y. Kubo, R. Schlüter, H. Ney: Feature selection for log-linear acoustic models. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 5324–5327, Prague, Czech Republic, May 2011.

[Wiesler & Richard+ 13] S. Wiesler, A. Richard, R. Schlüter, H. Ney: A critical evaluation of stochastic algorithms for convex optimization. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 6955–6959, Vancouver, Canada, May 2013.

[Wiesler & Richard$^+$ 14a] S. Wiesler, A. Richard, P. Golik, R. Schlüter, H. Ney: RASR/NN: The RWTH neural network toolkit for speech recognition. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 3313–3317, Florence, Italy, May 2014.

[Wiesler & Richard$^+$ 14b] S. Wiesler, A. Richard, R. Schlüter, H. Ney: Mean-normalized stochastic gradient for large-scale deep learning. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 180–184, Florence, Italy, May 2014.

[Wiesler & Schlüter$^+$ 11] S. Wiesler, R. Schlüter, H. Ney: A convergence analysis of log-linear training and its application to speech recognition. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 1–6, Waikoloa, HI, USA, Dec. 2011.

[Wiesler & Schlüter$^+$ 12] S. Wiesler, R. Schlüter, H. Ney: Accelerated batch learning of convex log-linear models for LVCSR. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 1207–1210, Portland, OR, USA, Sept. 2012.

[Wolfe 69] P. Wolfe: Convergence conditions for ascent methods. *SIAM Review*, Vol. 11, No. 2, pp. 226–235, 1969.

[Woodland & Povey 02] P.C. Woodland, D. Povey: Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, Vol. 16, No. 1, pp. 25–47, 2002.

[Xue & Li$^+$ 13] J. Xue, J. Li, Y. Gong: Restructuring of deep neural network acoustic models with singular value decomposition. In *Proc. of the Ann. Conf. of the Int. Speech Commun. Assoc. (Interspeech)*, pp. 2365–2368, Lyon, France, Aug. 2013.

[Young 92] S.J. Young: The general use of tying in phoneme based HMM recognizers. In *Proc. of the IEEE Int. Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, pp. 569–572, San Francisco, CA, USA, March 1992.

[Young & Odell$^+$ 94] S.J. Young, J. Odell, P.C. Woodland: Tree-based state tying for high accuracy acoustic modelling. In *Proc. of the Workshop on Human Language Technology (HLT)*, pp. 307–312, Plainsboro, NJ, USA, March 1994.

[Yu & Deng$^+$ 09] D. Yu, L. Deng, A. Acero: Using continuous features in the maximum entropy model. *Pattern Recognition Letters*, Vol. 30, No. 14, pp. 1295–1300, 2009.

[Zhang & Ragni$^+$ 10] S.X. Zhang, A. Ragni, M.J. Gales: Structured log linear models for noise robust speech recognition. *IEEE Signal Process. Lett.*, Vol. 17, No. 11, pp. 945–948, Nov. 2010.

[Zheng & Stolcke 05] J. Zheng, A. Stolcke: Improved discriminative training using phone lattices. In *Proc. of the European Conf. on Speech Commun. and Technology (Eurospeech)*, pp. 2125–2128, Lisbon, Portugal, Sept. 2005.

[Zweig & Nguyen 09] G. Zweig, P. Nguyen: A segmental CRF approach to large vocabulary continuous speech recognition. In *Proc. of the IEEE Workshop on Automat. Speech Recognition and Understanding (ASRU)*, pp. 152–157, Merano, Italy, Dec. 2009.