Singidunum University

Department of Postgraduate Studies

DOCTORAL DISSERTATION

Speech recognition in noisy environment using deep learning neural network

Supervisor:

Prof. dr Marina Marjanović-Jakovljević

Candidate:

Ashrf Nasef

Belgrade, 2017

Sažetak

Rezultati aktuelnih istraživanja u oblasti automatskog prepoznavanja govornika pokazuju da metode dubokog mašinskog učenja bazirane na neuralnim mrežama omogućavaju bolje performance of drugih klasifikatora baziranih na skrivenim Markovljevim modelima i Gausovim smešama. Sa druge strane, ove metode zahtevaju dosta podešavanja vrednosti parametara da bi se njihove performanse optimizovale za različite zadatke nadgledanog mašinskog učenja.

Cilj ove teze je da se demonstrira da adekvatni izbor vrednosti parametara može značajno da unapredi performanse metoda automatskog prepoznavanja govornika baziranih na dubokim neuralnim mrežama. Izložena studija predlaže jedan pristup automatskom prepoznavanju govornika zasnovan na neuralnim mrežama i algoritmu za stohastički gradijent. Poseban fokus se stavlja na tri parametra algoritma za stohastički gradijent: stopa učenja, stopa uklanjanja ulaznih neurona, i stopa uklanjanja skrivenih neurona. Dodatna pažnja je posvećena istraživačkom pitanju prepoznavanja govornika u uslovima povećane buke.

U skladu sa tim su izvedena dva eksperimenta. Cilj prvog eksperimenta je da se demonstrira da se optimizacijom vrednosti posmatranih parametara algoritma za stohastički gradijent mogu unaprediti performanse prepoznavanja govornika u uslovima bez buke. Ovaj eksperiment je organizovan u dve faze. U prvoj fazi je posmatrana stopa prepoznavanja za različite vrednosti stope uklanjanja skrivenih neurona i stope učenja, dok je stopa uklanjanja ulaznih neurona bila konstantna. U drugoj fazi ovog eksperimenta je posmatrana stopa prepoznavanja za različite vrednosti stope uklanjanja ulaznih neurona i stope učenja, dok je stopa uklanjanja skrivenih neurona bila konstantna.

Cilj drugog eksperimenta je da se demonstrira da se optimizacijom vrednosti posmatranih parametara algoritma za stohastički gradijent mogu unaprediti performanse prepoznavanja

govornika u uslovima povećane buke. Stoga su različiti nivoi buke veštački dodati izvornom govornom signalu.

Dobijeni rezultati pokazuju da se optimizovanjem stope uklanjanja neurona unapređuju performanse prepoznavanja govornika baziranog na stohastičkom gradijentu, čak i u uslovima povećane buke. Takođe je pokazano da izbor adekvatne vrednosti stope učenja predstavlja važan zadatak, jer neke vrednosti ovog parametra negativno utiču na performanse prepoznavanja

Abstract

Recent researches in the field of automatic speaker recognition have shown that methods based on deep learning neural networks provide better performance than other statistical classifiers. On the other hand, these methods usually require adjustment of a significant number of parameters.

The goal of this thesis is to show that selecting appropriate value of parameters can significantly improve speaker recognition performance of methods based on deep learning neural networks. The reported study introduces an approach to automatic speaker recognition based on deep neural networks and the stochastic gradient descent algorithm. It particularly focuses on three parameters of the stochastic gradient descent algorithm: the learning rate, and the hidden and input layer dropout rates. Additional attention was devoted to the research question of speaker recognition under noisy conditions.

Thus, two experiments were conducted in the scope of this thesis. The first experiment was intended to demonstrate that the optimization of the observed parameters of the stochastic gradient descent algorithm can improve speaker recognition performance under no presence of noise. This experiment was conducted in two phases. In the first phase, the recognition rate is observed when the hidden layer dropout rate and the learning rate are varied, while the input layer dropout rate was constant. In the second phase of this experiment, the recognition rate is observed when the input layers dropout rate and learning rate are varied, while the hidden layer dropout rate was constant. The second experiment was intended to show that the optimization of the observed parameters of the stochastic gradient descent algorithm can improve speaker recognition performance even under noisy conditions. Thus, different noise levels were artificially applied on the original speech signal.

The obtained results show that dropout optimization can significantly enhance the performance of stochastic gradient descent method in automatic speaker recognition even under noisy conditions. It is also shown that selecting an appropriate value of the learning rate is also a very important task, since for some values of this parameter, the performance of the method is negatively affected.

Acknowledgments

I am grateful to my supervisor professor dr. Marina Marjanović-Jakovljević for the continuous guidance and encouragement throughout this thesis. I remain highly indebted to her for all support I have received. Finally, I thankfully acknowledge the financial support received from the Ministry of Education in Libya.

Contents

List of figures

List of tables

Chapter 1
Introduction

1.1 Motivation and aim of research

The speech signal contains the information about the language (acoustic phonetic symbols), prosody (intonation signals), gender (vocal tract and pitch – frequency of voiced sounds), age, accent (formants), speaker's identity, emotion and health (Gold et al. 2011, Vaseghi 2007, Rashmi 2014). While the aim of speech recognition is to recognize the spoken words in speech, speaker recognition identifies the speaker by recognizing the spoken phrase, and verifies the speaker (Srinivas et al 2014). The abilities of decoding the speech signals, understanding the linguistic and speaker information in speech, and recognizing the speaker, are needed in many speech aided applications, such as access control, access to confidential information, voice command control, transaction authentication, and audio archive indexing (Desai and Joshi 2014, Beigi 2011, pp.16-22).

Recent researches have shown that speaker recognition methods based on deep learning neural network provide better performance than other classifiers (e.g. based on n-grams, hidden Markov models and Gaussian mixture models, cf. Chapter 2). However, these methods usually require adjustment of a significant number of parameters.

The main goal of this thesis is to show that selecting a good combination of parameters can significantly improve speaker recognition performance of methods based on deep learning neural

networks even in a noisy environment. More precisely, the thesis demonstrates that selecting an appropriate combination of parameters of the stochastic gradient descent algorithm (i.e. the learning rate, and the hidden and input layer dropout rates) can improve performance for the speaker recognition task. Two experiments were conducted for this purpose. The first experiment shows that the optimization of the observed parameters of the stochastic gradient descent algorithm can improve speaker recognition performance under no presence of noise. The second experiment shows that the optimization of the observed parameters of the stochastic gradient descent algorithm can improve speaker recognition performance when different noise levels are applied on the original speech signal.

1.2 Outline of the thesis

The rest of this introductory chapter briefly considers the tasks of speech recognition and speaker recognition, and selected applications of these technologies.

The Chapter 2 provides an overview of work in the field of speech recognition. It discusses in detail the statistical approach to speech recognition, including language modeling, acoustic modeling, and decoding, and emphasizes some limitations of the statistical approaches, especially those related to speech variations due to environmental noise. In addition, the chapter overviews of the relevant research on neural networks aimed at overcoming the shortcomings of the statistical approaches, while a more detailed technical discussion on methods based on neural networks is given in Chapter 4.

The speaker recognition process is fundamentally based on acoustic feature matching. Thus, Chapter 3 reports on 83 state-of-the-art acoustic features that are considered in this study, including pitch, intensity, and four orders of formants family, four orders of formants bandwidth, standard deviation, mean autocorrelation, mean noise-to-harmonics ratio and mean harmonics-

to-noise ratio. In addition, this chapter discusses in more detail different phases of the extraction process for the frequently used Mel-Frequency Cepstral Coefficients.

Chapter 4 first considers the theoretical concepts and methods that are relevant to neural networks and deep learning, including the notions of a sigmoid neuron and a deep feedforward neural network, the stochastic gradient descent algorithm, the backpropagation algorithm for computing gradient of a cost function, and the dropout method for addressing the problem of overfitting a neural network. Finally, the chapter introduces the neural network-based approach applied in the scope of this thesis for the purpose of automatic speaker recognition.

Chapter 5 reports on the experiments conducted in order to demonstrate the appropriateness of the approach in Chapter 4 for realistic conditions. The chapter first describes the VidTIMIT corpus that was used in the experiments, and then reports on the experimental settings and discusses the experimental results.

Chapter 6 makes concluding remarks.

1.3 Speech recognition

The term *automatic speech recognition* can be seen as computational mapping an acoustic signal to a sequence of words (Jurafsky and Martin 2009). Specification requirements for speaker recognition systems can vary significantly. Jurafsky and Martin (2009) differentiate among four dimensions of variation:

- *The size of vocabulary.* On the one side of the spectrum there are systems that are designed to recognize a very limited set of words, e.g., digits, etc. On the other side of the spectrum there are systems designed to recognize words form large vocabulary. Currently, large vocabularies contain 20,000 to 60,000 words.

- *Fluency of speech.* This dimension is related to the naturalness of speech. Recognition of isolated words is a relatively simple problem, but a user interface that integrates an isolated word recognition system is limited and not very natural. Continuous speech recognition systems are much more natural and useful, but their design is also more difficult.

- *Variation in channel and noise.* The quality of microphone and the noise level influence speech recognition to a great extent. Thus, design requirements for a speech recognition system intended to be used in a relatively silent laboratory settings and with a head mounted microphone will differ from design requirement for a system intended to be used in noisy settings and with a table microphone.

- *Speaker-class characteristics.* Speech recognition systems are usually designed for a target group of users described by a given speech corpus. Thus, such systems are tuned to recognize speech in that is considered standard with respect to the target group. However, any nonstandard speech makes recognition harder and prone to errors.

One of the fundamental assumptions of automatic speech recognition relates to the noisy channel model (Jurafsky and Martin 2009). Namely, the acoustic waveform is considered as being noisy version of the source sentence due to passing through a communication channel. The speech recognition task can be described as a search through a space of all possible sentences in order to select a sentence with the highest estimated probability of giving the noisy sentence (cf. Fig. 1.1). The process of searching and selection is called decoding, and will be discussed in more detail in Chapter 2.

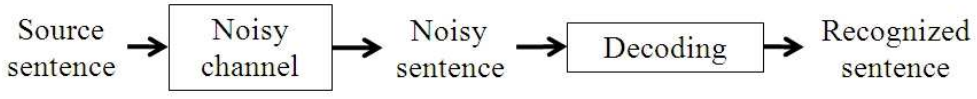

Figure 1.1: The noisy channel model (cf. Jurafsky and Martin 2009).

1.4 Speaker recognition

The term *automatic speaker recognition* is widely and sometimes confusingly used. Beigi (2011) states that this term broadly refers to:

> *[…] any procedure which involves knowledge of the identity of a person based on his/her voice* (Beigi 2011, p. 3).

According to Beigi, speaker recognition braches can be classified in two broad groups. The first group includes self-contained branches: *speaker verification* (or *speaker authentication*), *speaker identification*, and *speaker classification*. The second group includes branches that extend the simple branches by some additional functionality, i.e. *speaker segmentation*, *speaker detection*, and *speaker tracking* (Beigi 2011, pp. 3- 5). The following subsections will briefly discuss these branches.

Alternatively, speaker recognition tasks can be classified as text-dependent or text-independent. Text-dependent speaker recognition systems take into account the linguistic content of the user's utterance. Therefore, the user is required to utter a predefined sequence of words. Text-dependent speaker recognition is nowadays applied only for the verification task, because it is prone to spoofing attack from impostors, such as replay of pre-recorded utterances, advanced text-to-speech syntehsis or voice conversion (Shiota et al. 2016, Beigi 2011, pp. 12-14). In contrast to this, text-independent speaker recognition does not take into account linguistic content of uttered sentences, and is thus applicable to various speaker recognition tasks.

1.4.1 Speaker verification

In the speaker verification task, a given speaker makes an identity claim, and the verification task can be described as determining whether this claim holds or not. A speaker verification system

uses the user's identity claim to retrieve a *target speaker model* from an underlying database of user models, and compares it with the speech signal of the test speaker. The aim of this comparison is to evaluate the similarity of the test and target speaker. Two approaches that are usually applied for the purpose of this comparison rely on a *universal background model* and a *cohort model*, respectively (Beigi 2011, pp. 6-7). A universal background model is a model derived from a large population of speakers, and in this approach the test speaker is compared not only to the target speaker but also to the average population. If the system determines that the test speaker is closer to the average speaker than to the target speaker, the probability that the identity claim holds will be evaluated as low. In the second approach, each target speaker is associated with a cohort of other similar target speakers. If the system determines that the test speaker is closer to the target speaker than to the associated cohort, the probability that the identity claim holds will be evaluated as high (Beigi 2011, pp. 6-7).

In any case, the decision of an automatic verification system is always binary, it either accepts of rejects the identity claim of the test speaker. The identity claim is accepted only if the probability that a given sentence is uttered by the target speaker is greater than a threshold (Wildermoth 2001, p. 3):

$$decision = \begin{cases} accept, P_i(s) > threshold \\ reject, otherwise \end{cases}, \tag{1.1}$$

where:

- s is the uttered sentence,
- $P_i(s)$ is the probability that sentence s is uttered by the i^{th} speaker,
- $threshold$ is an experimentally derived value.

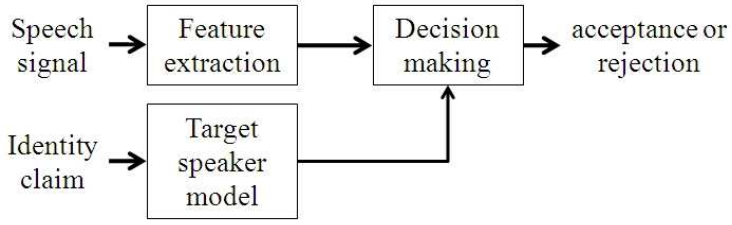

Figure 1.2: Speaker verification diagram.

A diagram describing the speaker verification task is given in Fig. 1.2.

1.4.2 Speaker identification

A speaker identification system is designed to recognize a given set of speakers. The identification task can be described as relating a given speech signal of an unknown speaker to one of the known speakers (Mølgaard and Jørgensen 2005). In addition, this task is further divided in *closed-set identification* and *open-set* identification. In closed-set identification, a given test speaker is compared to all speakers in an underlying database, and related to the most similar of the available speakers. Open-set identification includes a rejection scheme, and may be described as a combination of closed-set identification and verification: first, the test speaker is mapped to one of the speakers available in a database, and then the selected target speaker is verified (Beigi 2011, pp. 7-8).

A diagram describing closed-set identification task is given in Fig. 1.3 and 1.4. For closed-set identification, the resulting speaker belongs to the set the speakers available in the database. For open-set identification, the result may also include a rejection of the test speaker.

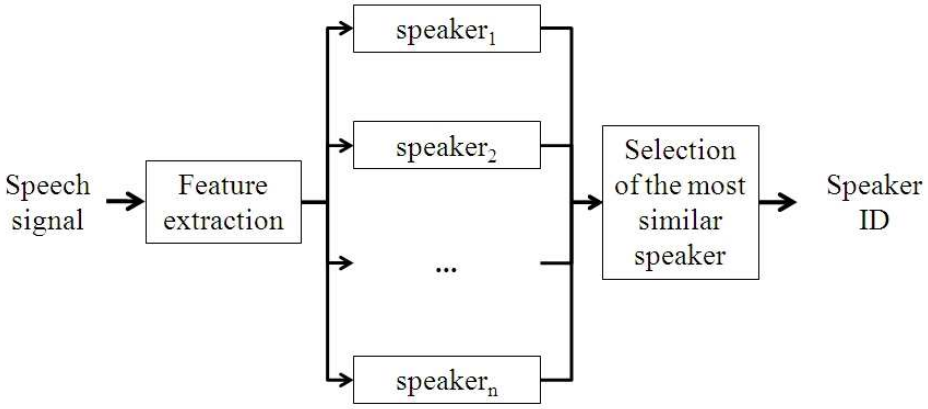

Figure 1.3: Closed-set identification diagram. The resulting speaker ID belongs to one of the speakers available in the database.

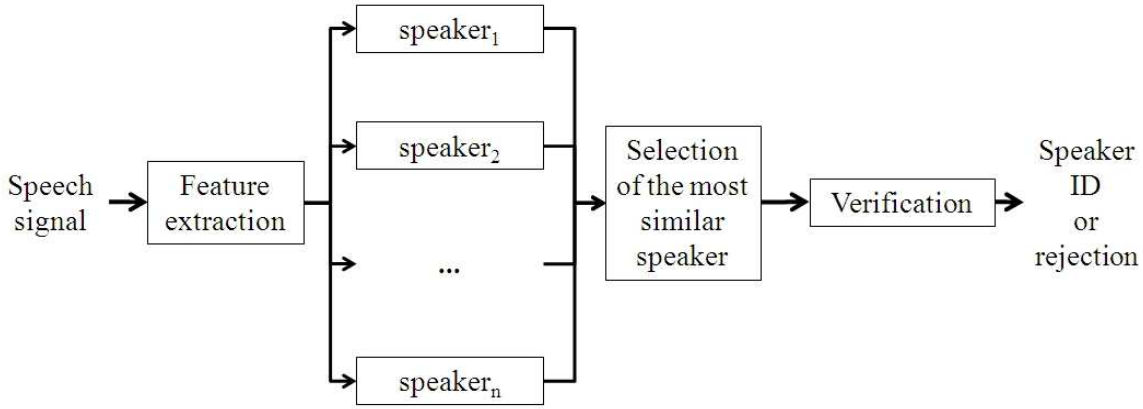

Figure 1.4: Open-set identification diagram. The result may also include a rejection of the test speaker.

1.4.3 Speaker classification

The aim of the speaker classification task is to group speech signals that are similar according to some criterion. Speaker classification has various manifestations, such as gender classification,

age classification, etc. Similarly to speaker verification and identification, the classification task is also based on a set of acoustic features. However, the informative value of a certain feature may vary accross different classification tasks. For example, pitch is often used for the gender classification task, while jitter, shimmer and spectral envelopes are used to determine the age of a speaker (Beigi 2011, pp. 8-9).

1.4.4 Compound branches of speaker recognition

The currently most popular compound branches of automatic speaker recognition are speaker segmentation, speaker detection, and speaker tracking (Beigi 2011, pp.9-12).

The speaker segmentation task can be described as dividing an audio stream into portions that contain speech of separate speakers and other sounds such as music, noise, etc. It is an important practical step both for speech recognition and speaker recognition, since recognizable speech in an input audio stream must first be separated from other sounds. In a general case of speaker segmentation, both the identities of speakers in an audio stream and their number are not known in advance. In a special case when the identities of the speakers are known, the segmentation task is reduced to the identification task.

The main goal of the speaker detection task is to detect speakers in an audio stream. This task includes speaker segmentation, since the observed stream may contains more speakers and additional sounds. In addition, this task also includes speaker verification, speaker identification, or their combination, depending on specification requirements for a speaker detection system. If all speakers in the audio stream are known, and there are no additional sounds, the closed-set identification can be applied for each segment. Otherwise, if not all speakers are known or there are additional sounds, the number of speakers that should be detected is used to determine the approach. If this number is small, verification can be applied on each segment and for each

speaker. If the number of speakers is large, then open-set identification (i.e. with verification of the identified speaker, cf. Fig. 1.4) can be applied.

The goal of the speaker tracking task is to tag different speakers in a given audio stream. Compared with the speaker detection task, speaker tracking does not require enrolment data, i.e. when the identities of speakers are not known, it is enough if a speaker tracking system identifies different speakers with different labels that do not necessarily relate to the speakers' identities (Beigi 2011, pp.9-12).

1.5 Applications

The speaker recognition technology is still not mature, but its applications are manifold, including the following (Beigi 2011, pp.5, 16-22):

- *Forensic, legal applications, surveillance.* The important characteristics of utilizing speech for such applications are that it can be collected relatively easily, on a large scale, and without the user knowldegde. Speaker recognition for these purposes is usually applied in the passive manner, i.e. a speaker recognition system does not influence the flow or type of the speeech data. However, surveillance applications have induced legal and ethnical concerns recently.

- *Audio indexing.* Audio indexing is the most common application of autimatic speaker recognition. It integrates speaker segmentation, detecting and tracking. In addition, indexing can be combined with speech recognition for the purpose of transcription of a given audio stream. One of the domains that can utilize audio indexing is teleconferencing.

- *Access control.* Exploiting speech as biometric for the purpose of access control has a distinguished advantage over other biometric sensors in application scenarios when the user is physically dislocated. In such applications, speech can be transmitted through a computer network or a telephony network, which is usually not a challenging task because these networks are well distributed.

The abovementioned applications certainly do not represent a complete list, and it may be said that speaker recognition has a potential to be utilized in many real-world application, especially keeping in mind ever growing number of mobile phones and other mobile devices with integrated microphones.

Chapter 2
Related work

2.1 Introduction

This chapter provides an overview of related work in the field of speech recognition. The currently dominant approach relates to statistical pattern matching. Thus, this chapter first reports on the statistical approach to speech recognition (cf. Section 2.2), including language modeling based on n-grams (Subsection 2.2.1), acoustic modeling based on hidden Markov models (Subsection 2.2.2), and decoding, i.e. determining the most probable sequence of words (Subsection 2.2.3). Subsection 2.2.4 emphasizes some limitations of the statistical approaches, especially those related to speech variations due to environmental noise. Neural networks represent a promising research direction to overcome the shortcomings of the statistical approaches. Section 2.3 briefly discusses the deep learning for speech recognition (a more detailed discussion is provided in Chapter 4), and provides an overview of the relevant research on neural networks in the context of speech recognition. Section 2.4 concludes this chapter.

2.2 The statistical approach to speech recognition

An important assumption in the context of speech processing is that a speech signal can be observed as a sequence of short segments, each of which can be considered as the output of a linear time-invariant system (Rabiner and Schafer 1978, p. 355). Therefore, each of these

segments can be appropriately described by a numeric value, with respect to its frequency, energy, etc. Let $W = w_1 w_2 \dots w_n$ be a sentence, and $O = o_1 o_2 \dots o_t$ a representation of a corresponding acoustic signal that consists of a sequence of *observations*, where observation $o_i, 1 \le i \le t$, describes the i^{th} segment of the acoustic signal. The probabilistic approach to speech recognition can be formulated as (Jurafsky and Martin 2009):

$$\hat{W} = \arg\max_{W \in L} P(W \mid O), \tag{2.1}$$

i.e. sentence W with the largest probability $P(W \mid O)$ is selected as optimal. To estimate the probabilities in the above equation, the Bayes' rule is applied (Jurafsky and Martin 2009):

$$\hat{W} = \arg\max_{W \in L} \frac{P(O \mid W) P(W)}{P(O)}, \tag{2.2}$$

where the *prior probability* of the sentence $P(W)$ may be calculated by a language model, and the *observation likelihood* $P(O \mid W)$ may be evaluated by an acoustic model. In contrast to these probabilities, the probability of a sequence of observations $P(O)$ cannot be evaluated. However, $P(O)$ is a constant value for each sentence $W \in L$, so the above equation can be simplified (Jurafsky and Martin 2009):

$$\hat{W} = \arg\max_{W \in L} P(O \mid W) P(W). \tag{2.3}$$

The process of selecting an optimal sentence is illustrated in Fig. 2.1.

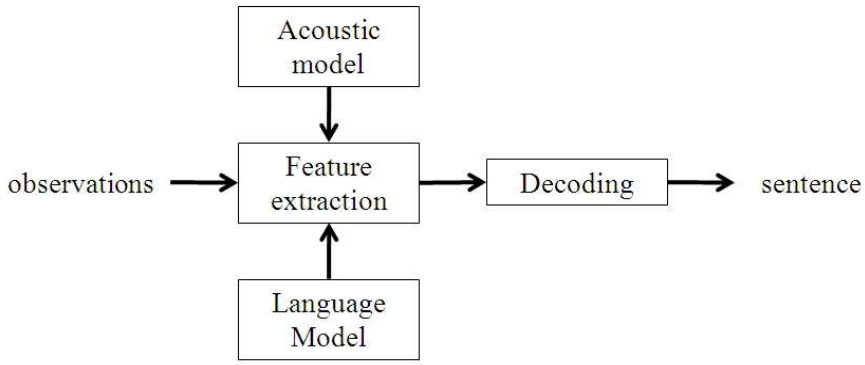

Figure 2.1: Illustration of a speech recognition process (cf. Jurafsky and Martin 2009).

The following three subsections discuss the language and acoustic models, and the decoding process in model details, while Chapter 3 reports on the process of generating acoustic observations (so-called *feature extraction*).

2.2.1 Language modeling

N-grams are statistical language models that can estimate a probability of a sentence or a probability of a word that follows a given sequence of words, based on a given language corpus (Jurafsky and Martin 2009). For the purpose of clarity, and without loss of generality, a special case of n-grams, so-called *bigrams*, is considered here.

To estimate a probability of a sequence of words $W = w_1, w2, \ldots w_n$, the conditional probability rule may be applied, i.e. $P(AB) = P(A)P(B \mid A)$. Thus, the probability of a sequence of words can be represented as (Jurafsky and Martin 2009):

$$P(W) = P(w_1, w2, \ldots w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w2) \cdots P(w_n \mid w_1, w2, \ldots w_{n-1}). \quad (2.4)$$

But the conditional probabilities in the equation (2.4) cannot be estimated directly, due to limited size of underlying linguistic corpora. In order to compute them, the Markov assumption is adopted, i.e., that the conditional probability of a word does not depend on the whole preceding sequence of words, but only on the immediately preceding word (Jurafsky and Martin 2009):

$$P(w_n \mid w_1, w2, \ldots w_{n-1}) = P(w_n \mid w_{n-1}). \tag{2.5}$$

Now, the equation (2.4) can be simplified:

$$P(W) = P(w_1, w2, \ldots w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w2)\cdots P(w_n \mid w_{n-1}) \tag{2.6}$$

In contrast to equation (2.4), the conditional probabilities in equation (2.6) can be estimated directly, using the *maximum likelihood estimation* (Jurafsky and Martin 2009):

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}, \tag{2.7}$$

where:

- $C(w_{n-1}w_n)$ is the number of occurrences of bigrams $w_{n-1}w_n$ in a given language corpus,
- $C(w_{n-1})$ – is the number of occurrences word w_{n-1} in a given language corpus.

In a general case of n-grams, when (*N-1)* preceding words are considered when estimating the probability of a given word, the equation for maximum likelihood estimation is (Jurafsky and Martin 2009):

$$P(w_n \mid w_{n-N+1} \ldots w_{n-1}) = \frac{C(w_{n-N+1} \ldots w_n)}{C(w_{n-N+1} \ldots w_{n-1})}. \tag{2.8}$$

Keeping in mind that the maximum likelihood estimation is based on a given language corpus, it is possible that the probability of some realistic sequences of words is evaluated to zero or close to zero, simply because they contain a bigram that is not present in the corpus or appears rarely. To overcome this problem we can apply the *Laplace smoothing* to n-gram probabilities. For the case of bigrams, the Laplace smoothing is represented as (Jurafsky and Martin 2009):

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}, \tag{2.9}$$

where V is the number of word types in the given corpus.

A language corpus is very important for the purpose of training an n-gram model. A usual approach is to divide a corpus in at least two parts: a training set and a test set. The n-gram probabilities are calculated over the training set, and afterwards tested over the test set. When dividing a language corpus, one must ensure that these two data sets do not contain same sentences, because it would result in a model that is biased, i.e. the probabilities of such sentences would be estimated as higher than they should be. An alternative division of language corpora has tree data sets: a training set, a development set, and a test set. A development set is used to set other parameters of a model or to provide additional test set. Jurafky and Martin (2009) state that, in practice, a language corpus is usually divided into 80% training, 10% development, and 10% test.

Perplexity is often used for the evaluation of n-gram models. The perplexity of a given test sentence $W = w_1, w2, \ldots w_n$ is (Jurafky and Martin 2009):

$$PP(W) = (P(w_1)P(w_2 \mid w_1)P(w_3 \mid w2)\cdots P(w_n \mid w_{n-1}))^{-\frac{1}{N}} \tag{2.10}$$

Lower perplexity signals more adequate test set probabilities. The basic idea is that the quality of a language model can be evaluated to the extent to which it fits to test data. Thus, perplexity can be used to compare between different language models.

2.2.2 Acoustic modeling

Modern speech recognition systems very often apply hidden Markov models for acoustic modeling (Jurafsky and Martin 2009, Bishop 2006, Rabiner and Juang 1993, Rabiner 1989). A hidden Markov model is determined by:

- Q – set of hidden states, including a start stat and a final state.
- $A_{n\times n}$ – transition probability matrix, where element a_{ij} is the probability of transition from state *i* to state *j*,
- $B = b_i(o_t)$ – a sequence of observation likelihoods (i.e. emission probabilities), where element $b_i(o_t)$ is the probability that observation o_t is emitted from state q_i.

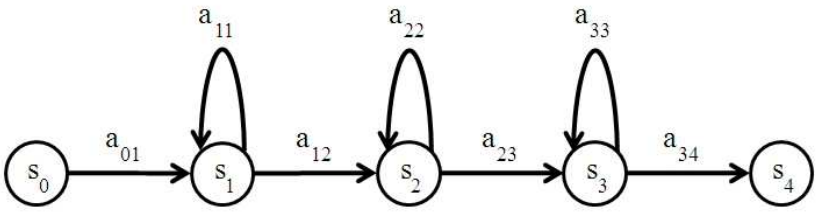

Figure 2.2: Five-state hidden Markov model of a phone. States s0 and s4 are the start and final states, respectively. The emitting states s1, s2 and s3 correspond to entry-transition, steady-state and exit-transition parts of a phone, respectively.

Acoustic models are based on hidden Markov models in which phones represent hidden states, and acoustic feature vectors represent observations. A phone is modeled by a hidden Markov model that contains five states, as illustrated in Figure 2.2. Two of these states are non-emitting states, i.e., the start and the final states, while the rest three states correspond to entry-transition, steady-state and exit-transition parts of a phone (cf. Jurafsky and Martin 2009).

For the purpose of illustration, a hidden Markov model of a word consisting of three phones is depicted in Fig. 2.3. It can be observed that such hidden Markov models have a so-called *Bakis network*, which means that they do not contains transitions to earlier states.

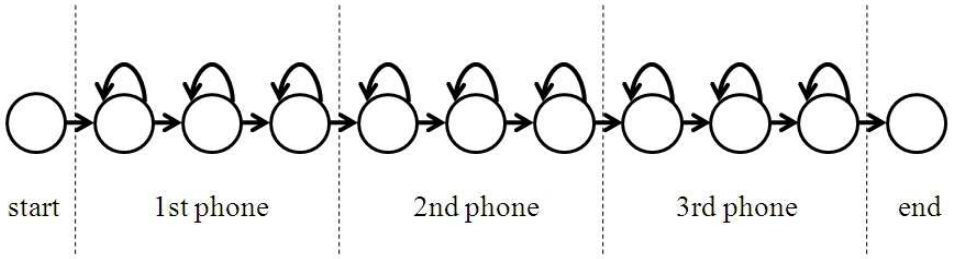

Figure 2.3: A hidden Markov model of a word consisting of three phones.

Let $V = \{w_1, w_2, \ldots, w_k\}$ be a vocabulary consisting of k words. For each word $w_i \in V$, a hidden Markov model λ_i that represents the given word may be defined. The estimation of parameters of these models is based on a training set, i.e. they optimize the likelihood of acoustic observations. Recognition of an isolated word from the vocabulary represented by observation sequence O can be formulated as (Rabiner 1989, p. 276, cf. fig. 2.4):

$$\hat{v} = \arg\max_{1 \le i \le |V|} P(O \mid \lambda_i) \,. \tag{2.11}$$

For each hidden Markov model λ_i, the likelihood $P(O \mid \lambda_i)$ is calculated, and the word represented by a hidden Markov model that generates the maximum likelihood is selected. The likelihood of observation $O = o_1 o_2 \ldots o_t$ is estimated as follows (Jurafsky and Martin 2009):

$$P(O) = \sum_{Q} P(O,Q) = \sum_{Q} P(O \mid Q) P(Q), \tag{2.12}$$

where:

$$P(O,Q) = P(O \mid Q) P(Q) = \prod_{i=1}^{t} P(o_i \mid q_i) \cdot \prod_{i=1}^{t} P(q_i \mid q_{i-1}). \tag{2.13}$$

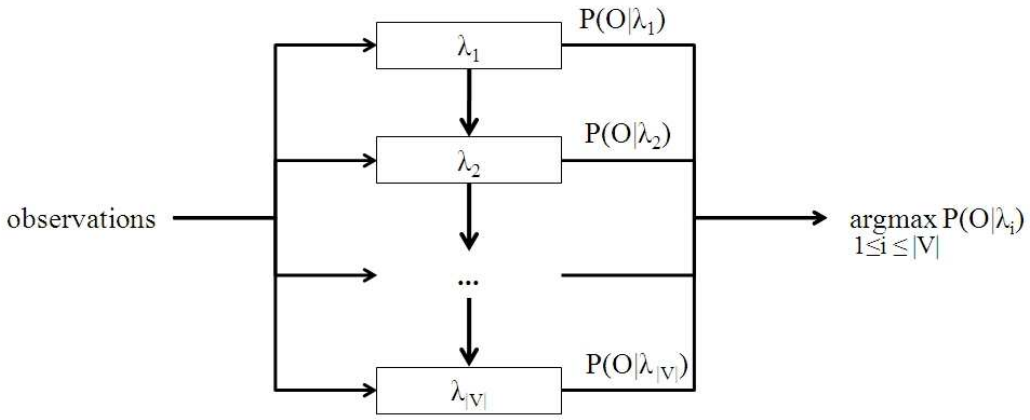

Figure 2.4: Illustration of isolated word recognition. Hidden Markov model λ_i represents word $w_i \in V$ (cf. Rabiner 1989, p. 276).

However, the described calculation is time-consuming and can be optimized by applying the *forward algorithm*, given in fig. 2.5 (Jurafsky and Martin 2009). To explain the algorithm, the forward path probability $\alpha_l(j)$ is conceptualized to represent the probability that a given hidden Markov model will be in state j after the observation sequence $o_1 o_2 \ldots o_l$ is emitted, i.e.,

$$\alpha_l(j) = P(o_1 o_2 \ldots o_l, q_t = j) \text{ .} \tag{2.14}$$

Initialization:

$\alpha_1(j) = a_{0j} b_j(o_1)$, where $1 \le j \le n$.

Recursive step:

$$\alpha_l(j) = \sum_{i=1}^{n} \alpha_{l-1}(i) a_{ij} b_j(o_l) \text{ , where } 1 \le j \le n, 1 \le l \le t \text{ .} \tag{2.15}$$

Final step:

$P(O \mid \lambda) = \alpha_t(q_f) = \sum_{i=1}^{n} \alpha_t(i) a_{if}$, where q_f is the final state.

Figure 2.5: The forward algorithm.

This probability can be defined recursively. If the model was in state i in the previous time step, and its forward path probability was $\alpha_{l-1}(i)$ **,** then the probability $\alpha_l(j)$ is equal to:

$$\alpha_l(j) = \alpha_{l-1}(i) a_{ij} b_j(o_l) \text{ .} \tag{2.16}$$

Since the above probability is evaluated under the assumption that the model was in a specific state i, it does not allow for arbitrary transitions in state j. Thus, it represents only one portion of the probability $\alpha_l(j)$ in a general case. The equation (2.16) can be generalized, so that the forward path probability $\alpha_l(j)$ is calculated by summing partially calculated probabilities over all possible states, as formulated in equation (2.15). This recursive step is illustrated by the forward trellis given in fig. 2.6. The forward algorithm significantly reduces the calculation complexity, i.e. to $O(n^2 t)$.

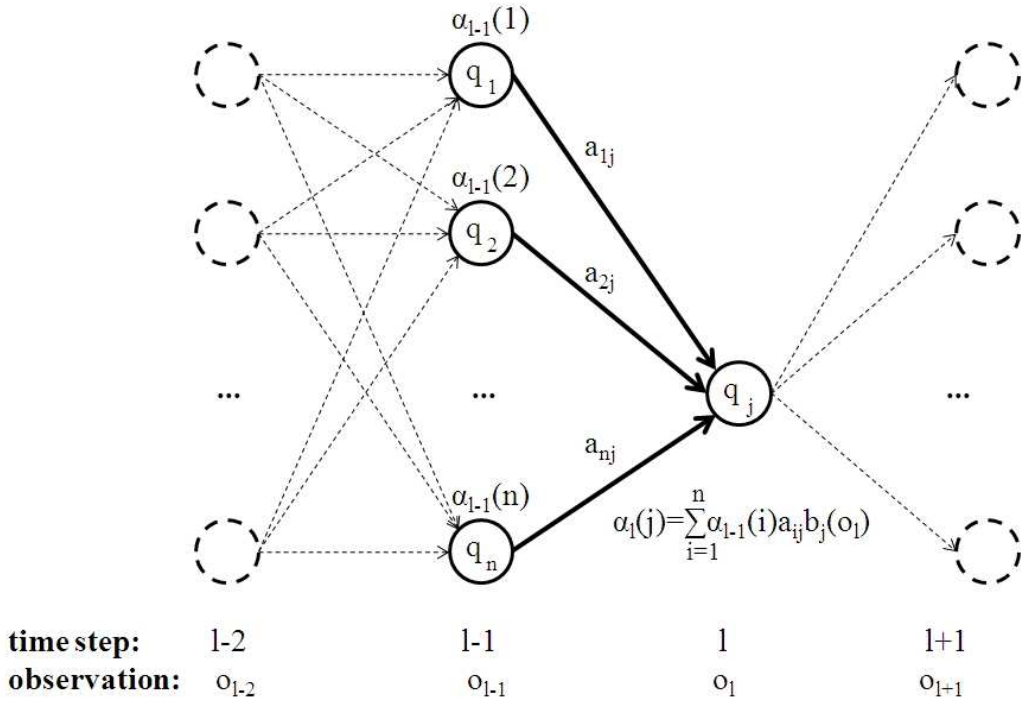

Figure 2.6: The forward trellis.

2.2.3 Decoding

The decoding problem relates to discovering the most probable path through states of a given hidden Markov model for a given sequence of observation. For this purpose, the Viterbi algorithms is often applied (Jurafsky and Martin 2009, cf. fig. 2.7).

The Viterbi path probability $v_l(j)$ represents the probability that a given hidden Markov model will be in state j after the observation sequence $o_1 o_2 \ldots o_l$ has been emitted and the model passed through the most likely sequence of hidden states $q_1 q_2 \ldots q_{l-1}$, i.e.,

$$v_l(j) = P(q_1 q_2 \ldots q_{l-1}, o_1 o_2 \ldots o_l, q_t = j)\,. \tag{2.18}$$

This probability can be defined recursively, similarly as the forward path probability (Jurafsky and Martin 2009):

$$v_l(j) = \max_{1 \le i \le n} v_{l-1}(i) a_{ij} b_j(o_l), \tag{2.19}$$

where:

- $v_{l-1}(i)$ is the Viterbi path probability for state i in the previous time step $l-1$,
- a_{ij} is the transition probability,
- $b_j(o_l)$ is the state observation likelihood,

as defined at the beginning of the previous section. In addition, the Viterbi algorithm recursively defines a sequence of *backpointers* $\beta_l(j)$ that enable backtracking in order to determine the optimal sequence of hidden states (Jurafsky and Martin 2009).

Initialization:

$v_1(j) = a_{0j} b_j(o_1)$, where $1 \le j \le n$,

$\beta_1(j) = 0$, where $1 \le j \le n$.

Recursive step:

$$v_l(j) = \max_{1 \le i \le n} v_{l-1}(i) a_{ij} b_j(o_l), \text{ where } 1 \le j \le n, 1 \le l \le t, \tag{2.17}$$

$$\beta_l(j) = \arg\max_{1 \le i \le n} v_{l-1}(i) a_{ij} b_j(o_l), \text{ where } 1 \le j \le n, 1 \le l \le t.$$

Final step:

$$P = \max_{1 \le i \le n} v_{l-1}(t),$$

$$\beta_t(q_f) = \arg\max_{1 \le i \le n} v_{l-1}(t), \text{ where } q_f \text{ is the final state.}$$

Figure 2.7: The Viterbi algorithm.

Isolated word recognition was already illustrated in fig. 2.4. However, large vocabulary continuous speech recognition requires also *cross-word decoding*, so the observed set of hidden Markov models, each of which represents a word from a vocabulary, has to be extended. For the purpose of clarity, we assume that the observed vocabulary contains only two words: w_1, w_2.

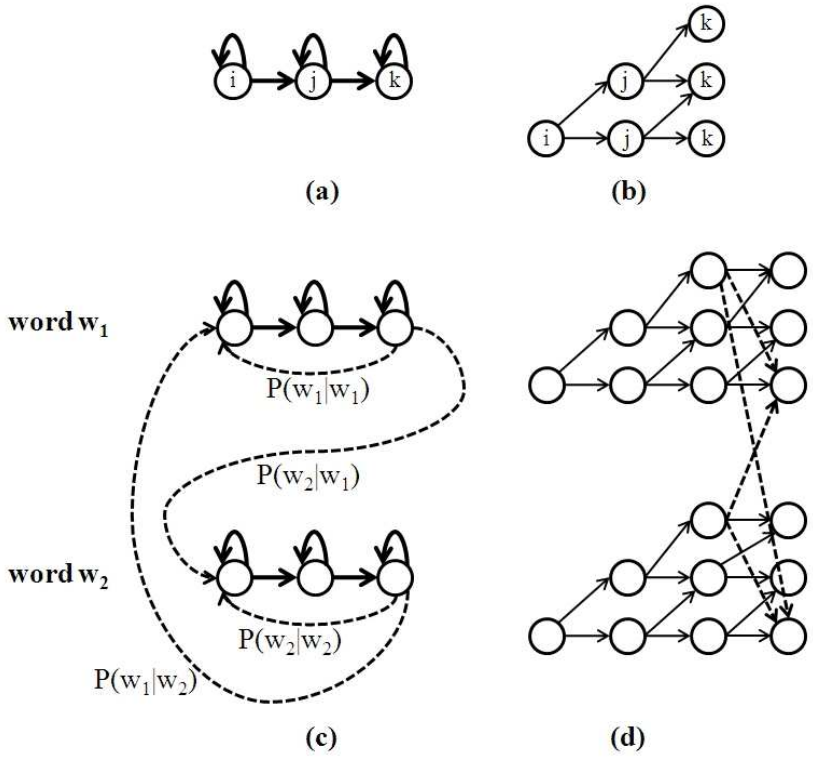

Figure 2.8: Cross-word decoding: (a) illustration of a hidden Markov model of an isolated word, containing only intra-word transitions, (b) Viterbi trellis for an isolated word, (c) illustration of a hidden Markov model for a vocabulary, extended with inter-word transitions, (d) extended Viterbi trellis. The intra-word transitions are denoted by solid arrows. The inter-word transitions are denoted by dashed arrows.

Fig. 2.8(a) illustrates a hidden Markov model that describes an isolated word. This model includes only *intra-word transitions*, denoted by solid arrows. The corresponding Viterbi trellis

is given in Fig. 2.8(b). However, a hidden Markov model that describes both words in the vocabulary consists of models for each word extended with *inter-word transitions*, e.g. the bigram probabilities of transitions from one word to another: $P(w_1 \mid w_1), P(w_2 \mid w_1), P(w_1 \mid w_2), P(w_2 \mid w_2)$, as shown in 2.8(c). The Viterbi trellis for the extended model is given in 2.8(d). The inter-word transitions are denoted by dashed arrows. It is important to emphasize that the probabilities of inter-word transitions are derived from the language model. Such an extension allows for calculation of a Viterbi trellis for a sentence. Still, the Viterbi algorithm has to be further optimized in order to be appropriate for large vocabulary speech recognition. Usually, when considering inter-word transitions, paths of low probability are *pruned*. In other words, not all words are considered in each step (Jurafsky and Martin 2009).

2.2.4 Limitations of the statistical approaches

Dealing with environmental noise (such as car engine, traffic noise, white noise, crowd noise, etc.) and speech signal variations caused by modifications of articulation (that can be found in the speaker's pitch, etc.) represents an important issue in automatic speech recognition (Santos et al. 2015, Kacur 2004). The current speech recognition systems are designed to work in controlled environments using clean speech, and so far have reached high levels of performance (Alam et al. 2015). However, when exposed to noisy environments, the performance of these systems degrades rapidly. Due to ever increasing use of speech-based user-centric applications (such as voice interactions with mobile devices like Bing voice search, Siri on iPhone, etc.), noise robustness is becoming an important requirement (Li et al. 2014, Cui and Alwan 2005).

Noise may be broadly classified as additive noise generated by external sound sources, and convolutional noise caused by channel characteristics (Jurafsky and Martin 2009). Diverse techniques are applied to improve speech recognition in noisy conditions. Noise resistance features and similarity measurements techniques focus on the effects of noise on the speech signal, rather than on the noise removal, attempting to derive features which are noise resistant.

Speech enhancement techniques attempt to remove the corrupting noise without altering parameters of the acoustic model (Kumar and Florencio 2016). Model adaptation technique aim at changing acoustic model parameters in accordance with the noisy speech signal, i.e. the statistical modeling techniques are trained using clean speech and then are adapted to noisy speech.

The majority of speech recognition systems use hidden Markov models combined with Gaussian mixture model probability density functions to compute the likelihood of an acoustic feature vector. The output likelihood probability function is defined as (Jurafsky and Martin 2009):

$$b_j(o_l) = \sum_{i=1}^{G_j} \frac{c_{ji}}{\sqrt{2\pi |\Sigma_{ji}|}} \cdot \exp((x-\mu_{ji})^t \Sigma_{ji}^{-1}(o_t - \mu_{ji})), \tag{2.20}$$

where:

- G_j is the total number of Gaussian mixture models probability density functions assigned to state j,
- c_{ji} is the mixture weight assigned to i^{th} mixture model,
- μ_{ji} – mean of the Gaussian distribution assigned to i^{th} model,
- Σ_{ji} – covariance matrix of the Gaussian distribution assigned to i^{th} model.

However, Gaussian mixture models suffer from severe disadvantage reflected in the fact that the modeling of even small non-linear deviations may require a large number of Gaussians, which makes them inefficient for acoustic modeling of data (Hinton et al. 2012). Thus, this approach is easily affected by speech variations in daily conversations, particularly are sensitive to mismatch introduced by environmental noise (Seltzer et al. 2013).

2.3 Deep learning for speech recognition

Developments in the field of deep neural networks demonstrated convincing improvements in speech recognition performance, and suggest that neural networks may be applied to overcoming the disadvantages of the statistical approaches. The fundamental architecture across deep neural network systems is a network that consists of several hidden layers of connected neurons whose activations are a nonlinear function of a linear combination of the activations of the previous layer. The most used hidden neuron activation function is the sigmoid function. However, in this thesis the rectified linear unit is applied (cf. Chapter 4, Nasef et al. 2017a, Nasef and Marjanović-Jakovljević 2017b). Compared to the sigmoid function, it is found that a rectified linear unit significantly accelerates the convergence of stochastic gradient descent, since the activation is simply setting the threshold at zero (Maas et al. 2013). Networks with such a function are often trained with a dropout regularization technique for improved generalization for large models (Srivastava et al. 2015). The final layer usually does not have an activation function, because it is taken to represent the class scores which are either real-valued numbers or a target.

Gradient descent learning algorithms minimize neural network loss functions by iteratively computing the gradient to adjust the weights, and using them to update parameters at every step. Parameter update requires the learning rate to be set to an appropriate value. The learning rate determines how fast the algorithm moves towards the optimal weights. If the learning rate is too low, then the algorithm will perform too many iterations while converging to optimal values, and thus be inefficient, and if the learning rate is too high, the progress will be faster, but with a risk that the optimal solution will be omitted. Therefore, using a good learning rate is crucial.

Gradient descent learning algorithms estimate the gradient on a large dataset (batch), performing redundant computations (as recomputed gradients for similar examples before each parameter update). The stochastic gradient descent is usually much faster because it estimates the gradient from just a few examples at a time instead of the entire training set. Mini-batch stochastic

gradient descent takes the best of the both approaches and performs an update for every mini-batch whose size is usually between 50 and 256. A more detailed elaboration of neural networks is provided in Chapter 4.

Recently, researchers have started to explore several different strategies for using deep neural networks for speech recognition, speaker recognition, and spoken language recognition tasks (Srinivas et al. 2014, Richardson et al. 2015, McLaren et al. 2015). Among the first results, in 2000, was the use of deep belief networks, which are based on restricted Boltzmann machines, and deep autoencoders (Vasilakakis et al. 2013). However, training deep neural networks with big number of hidden layers with autoencoders has shown to be a quite difficult task (Le 2015). From 2006, dimensionality of data with autoencoder networks was reduced by gradient descent which is used for fine-tuning the weights (Hinton and Salakhutdinov 2006). Furthermore, this approach has branched into major variants, such as batch gradient descent, stochastic gradient descent, and mini-batch gradient descent (Le et al. 2011). When dealing with continuous speech recognition, the recurrent neural networks were proposed (Robinson et al. 1996).

One of the recent advances in deep neural networks, that improve its performance, optimization, and prediction quality, are rectified linear units, and dropout (to overcome the problem of overfitting) (Dahl et al. 2013). However, there are still challenges that remain to be addressed. Sutskever et al. (2013) showed the importance of momentum-accelerated stochastic gradient descent that uses well-designed random initialization. Le et al. (2011) introduced more sophisticated optimization methods such as Limited memory Broyden-Fletcher-Goldfarb-Shanno and conjugate gradient that simplify and speed up the process of pre-training deep algorithms. Senior, et al. (2013) trained deep neural networks for large vocabulary speech recognition with mini-batch stochastic gradient descent by using a variety of learning rate schemes. They show that adequate choice of learning rate schemes leads to faster convergence, and lower word error rates.

There are few research works that have addressed the robustness of automatic speech recognition systems under noisy conditions using deep neural networks. Kumar and Florencio (2016) studied speech enhancement in office environment conditions where multiple stationary or non-stationary noises can be simultaneously present in speech. They collected 95 noise samples observed in office environment that are then mixed and added to the clean utterance of the TIMIT training set at a random signal-to-noise ratio chosen uniformly from -5dB to 20dB. Their results show that strategies based on deep neural networks provide an increment in average perceptual evaluation of speech quality of 24%.

In order to evaluate the performance of a deep neural network-based acoustic model for noise robust speech recognition, Seltzer et al. (2013) performed a series of experiments on the Aurora 4 medium vocabulary task that is based on the Wall Street Journal corpus. The 7137 utterances recorded from 83 speakers include a combination of clean speech and speech corrupted by one of the six noises (car, street traffic, train, airport, restaurant and babble) at 10-20dB signal-to-noise ratio. They obtained the best performance when applying the combination of noise-aware training and dropout, with improvement of 7.5%.

Mitra et al. (2014) evaluated robust features on deep and convolutional neural networks for noisy English continuous speech recognition task of Aurora 4, and demonstrated that they can improve the recognition performance compared to the mel-filterbank energies. They show that the vocal tract length normalization has a positive impact on improving the performance of the robust acoustic features.

De-la-Calle-Silos et al. (2014) tested the robustness of different automatic speech recognition systems based on deep neural networks (e.g., basic deep neural networks, deep neural networks with dropout, and deep maxout networks) by digitally adding four different types of noises (white, street, music, and speaker) at four different signal-to-noise ratios to the clean speech. The experiments, performed on the TIMIT corpus, show improvement in the recognition accuracy over traditional techniques for both clean and noisy conditions.

Noda et al. (2015) demonstrated that a deep denoising autoencoder can effectively filter out the effect of noise superimposed on original clean audio inputs, and such denoised audio features attain noise robustness in an isolated word recognition task.

Misamadi and Hansen (2015) explored deep neural network acoustic model adaptation in order to achieve improved noisy robust automatic speech recognition systems. They adapted the clean-trained neural network model to speech data selected from the Aspire challenge data. The experiment uses 10 passes of adaptation data, with a mini-batch size of 256, and a fixed learning rate of 0.001. They obtained relative word error rate improvement of 16%.

Kim et al. (2016) proposed a noise adaptation framework that employs knowledge of background noise and learns low-dimensional noise features from a trained deep neural network. In order to evaluate the proposed method, they trained the model using datasets, RM (Resource Management), and CHiME-3, and then tested it with the Aurora 4 task. They verified the effectiveness of the proposed noise adaptation approach in which a trained deep neural network dynamically adapts a speech recognition system to its usage environment.

2.4 Conclusion

This chapter provided an overview of related work in two broad research directions. The first research direction, which is currently dominant in the field, is primarily statistical and based on a combination of hidden Markov models and Gaussian mixture models for modeling acoustic representations of features extracted from the signal. Another research direction is related to promising and rapidly emerging methodological approach based on neural networks.

The further chapters of this thesis will focus on the latter research direction. However, the next chapter considers acoustic feature extraction, which is an aspect of pattern recognition that is

important for both these research directions. Special attention will be devoted to acoustic features relevant for speaker recognition.

Chapter 3
Feature extraction

3.1 Introduction

Feature matching is an important part of the speaker recognition process. In its first part, this chapter introduces the basic notion of feature extraction (Section 3.2), and reports on the selected 83 state-of-the-art acoustic features that are applied in the scope of this research (Section 3.3). In the second part (Section 3.4), this chapter discusses in more detail different phases of the extraction process for the frequently used Mel-Frequency Cepstral Coefficients, including preemphasis, windowing, Discrete Fourier Transform, Mel filter-bank, log, Inverse Discrete Fourier Transform, etc. Section 3.5 concludes the chapter.

3.2 Basic notion of feature extraction

The purpose of feature matching is to find the best match that is used to identify the unknown speaker (cf. Nasef et al. 2017a). E.g., in speaker verification, the unknown speaker first claims identity, and then one-to-one matching is done, i.e. the claimed model is used for the identification. As already mentioned in Chapter 1, if the match is above a predefined threshold,

the identity claim is accepted. A general speaker recognition system architecture is shown in Fig. 3.1 (Sturim et al. 2007).

To recognize the speaker, the first step in speaker recognition system is to convert the speech waveform, using digital signal processing tools to a set of features for further analysis. During signal preprocessing, sometimes when removing unwanted information, some useful information can be lost. After preprocessing of the speech signal in the signal modeling, the next step is parameterization of speech signal, which is called feature extraction. The aim of this process is to produce a meaningful representation of the speech signal. Some main tasks of feature extraction are the conversion of the speech signal to a digital form (signal conditioning), measuring important characters of the signal (signal measurement), augmenting these measurements with derived measurements (signal parameterization), and statistical modeling. For parametrically representing the speech signals there are several common methods, such as Mel-Frequency Cepstrum Coefficients, Filter Bank Energy analysis, and others (Anusuya and Katti 2009).

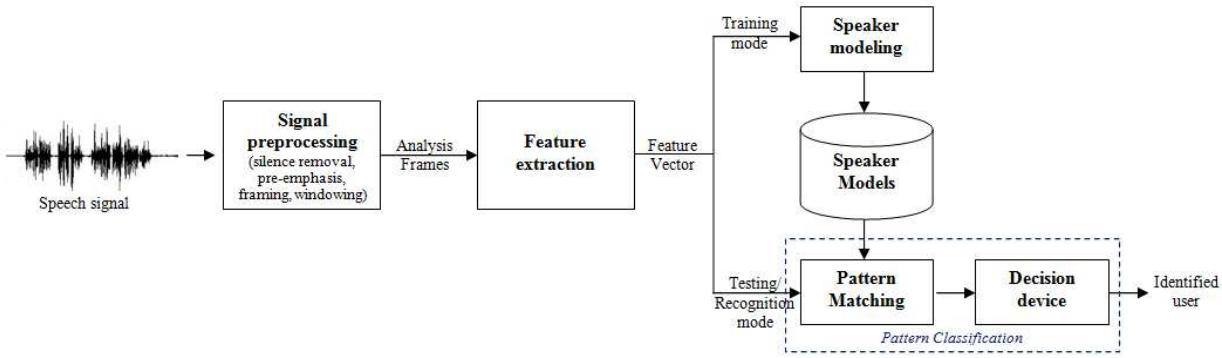

Figure 3.1: General speaker recognition system architecture (Sturim et al. 2007).

The result of feature extraction is a sequence of acoustic vectors that serve as a set of training vectors for the observed speaker (Hinton et al. 2012). The next step is pattern/feature matching that identifies a test speaker by matching extracted features with a set of known speakers. The pattern classification measures the similarity of the input feature vectors, and groups the patterns

that share the same properties. According to the result of the pattern classification, a recognition system decides whether to accept or reject a speaker (Campbell 1997).

3.3 Selected features

Mølgaard and Jørgensen (2005, p. 2) state some of the requirements for acoustic features intended to be used for the purpose of automatic speech recognition. According to them, optimal acoustic features:

- differ between speakers, but are resistant to intra-speaker variations,
- can be easily measured,
- do not vary, or vary slowly over time,
- are frequent and naturally-occurring,
- do not change significantly across different environments,
- are hard to imitate.

In addition, Mølgaard and Jørgensen emphasize that the optimal features are based on spectral analysis. Indeed, speaker recognition systems usually exploit spectral features obtained from short time speech segments (Shriberg et al. 2005).

In this study, we use 83 state-of-the-art features that are extracted using MATLAB or PRAAT software (Boersma and Weenink 2016), namely pitch, intensity, and four orders of formants family, four orders of formants bandwidth, standard deviation, mean autocorrelation, mean

noise-to-harmonics ratio and mean harmonics-to-noise ratio. These features are explained in more detail below (cf. Nasef et al. 2017a).

(1) Formant frequencies (*F*) indicate resonating frequencies of the vocal tract. *F* is determined as:

$$F = \frac{F_s}{2\pi} \arctan \frac{im(s)}{re(s)}, \tag{3.1}$$

where:

- F_s is sampling frequency,
- $im(s)$ and $re(s)$ are imaginary and real parts of the sound signal *s*.

For the purpose of this study, the maximum value (formant_max), the minimum value (formant_min), the standard deviation, (formant_std), the mean (formant_mean), and the median (formant_median) for formant bandwidths are calculated.

(2) Pitch estimation algorithms can be divided in three approaches: time domain, frequency domain, and statistical approaches (Joho et al. 2007). In the scope of time domain approaches, for the purpose of this work, the Zero Crossing Rate (ZCR) and autocorrelation are used. From speech signal *s*, pitch can be calculated as:

$$\rho_0(s) = \psi\left\{\log\left|\psi(s.\omega_n^H\left(\|s\|\right)\right|\right\}, \tag{3.2}$$

where:

- $\rho_0(s)$ is the pitch,

- ψ shows the Discrete Fourier transform function,
- $\|s\|$ represents a length of the signal *s*,
- $\omega_n^H(\|s\|)$ is the Hamming window, calculated as follows:

$$\varpi_n^H = 0.54 - 0.46\cos\left(\frac{2\pi n}{L}\right),\ 1 \le n \le N-1. \tag{3.3}$$

For every sequence of pitch values, a set of features is calculated, including the maximum, the mean and the minimum pitch, etc. These features are normalized using percentile value.

(3) Zero Crossing Rate (ZCR) is calculated using Chen's formula (Chen 1988):

$$ZCR = \frac{1}{T-1}\sum_{t=1}^{T-1}\Pi\{s_t s_{t+1} < 0\}, \tag{3.4}$$

where s_t, and s_{t-1} are the values of the sound signals at time *t* and *t*-1, respectively and $\Pi\{A\}$ is the indicator function, i.e., if the argument A is true then the function Π is equal to 1, otherwise it is 0.

(4) Autocorrelation function $r(\tau)$ of a signal with time lag τ aims to maximize the product between the waveform and its shifted version. It is defined as:

$$r(\tau) = \frac{1}{N}\sum_{n=0}^{N-1} s(n)s(n+\tau). \tag{3.5}$$

(5) Harmonics-to-Noise ratio (HNR) features can be used to quantify a perceptual impression of a rough voice. For example, if 99% of the energy of the signal is periodic and 1% is noise,

then *HNR*=20dB. Usually young speakers can produce approximately 20 dB of *HNR* (Friedland et al. 2009).

(6) Cepstrum coefficients (CC) are useful because they separate source and filter. Truncating the cepstrum at different frequency values allows for preserving different amounts of spectral detail. Cepstrum is defined as the inverse Discrete Fourier Transform of the log magnitude of the Discrete Fourier Transform of a signal.

(7) Mel-filterbank is typical short-term spectral analysis technique where speech data is split into overlapping time-frames where spectrum of each frame is analyzed with Discrete Fourier Transform.

(8) Mel-frequency cepstral coefficients (MFCC) are calculated using Davis and Mermelstein model (Davis and Mermelstein 1980, Maesa et al.2012):

$$MFCC_i = \sum_{k=1}^{N} X_k \cos\left[i(k-1)\frac{\pi}{N} \right], i = 1,2,...,M \,, \tag{3.6}$$

where:

- M is the number of cepstrum coefficients,
- X_k, k=1,2,...,N are log energy output of the k^{th} filter,
- N represents the number of triangular band pass filters.

(9) Filter-bank energy (FBE) is a small set of parameters describing the speech spectrum envelope in the observed frame. In order to take out information about pitch and to lower down estimation error, this approach integrates the periodogram in frequency bands (Hernando and Nadeu 1997).

(10) Percentile normalization of region duration (Holube et al. 2010).

Figures 3.2 and 3.3 show the speech signal in time domain and spectrogram of the speech signals for a female sample and a male sample, respectively. Both speakers utter the same sentence. Observing the time domain of the speech signal from these figures during the interval of silence, it can be concluded that the recording was performed in a noisy environment. The lower parts of the figures show the spectrograms of the signals where formants are denoted in red, intensity is denoted in yellow, and pitches are denoted in blue.

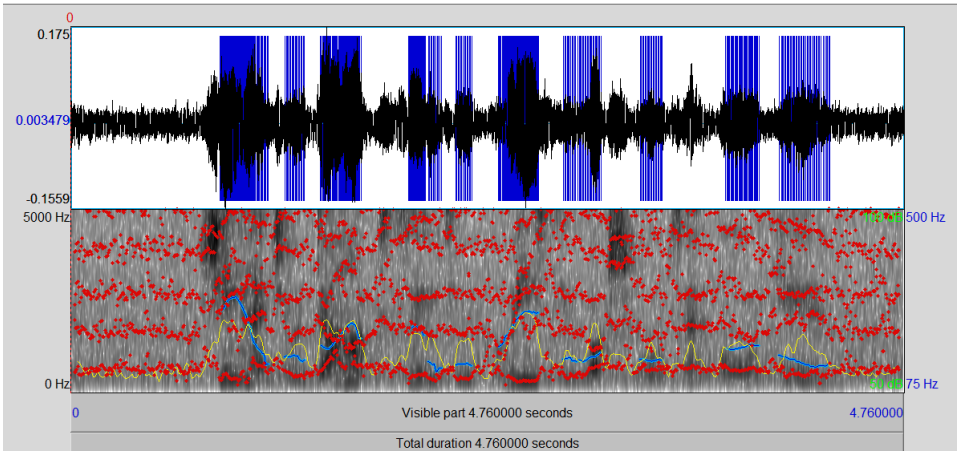

Figure 3.2. Speech signal in time and frequency domains for a female sample (and the same utterance as in Fig. 3 3). The lower part shows the spectrogram of the signal where formants (denoted in red), intensity (denoted in yellow) and pitches (denoted in blue) are extracted using the software PRAAT (Boersma and Weenink 2016).

Comparing the given female and male spectrograms for the same utterance, one can observe that formant frequencies of the female speaker are higher. This was expected because women generally have shorter vocal tracts than man. The higher voice fundamental frequency means that there is a longer interval between voice harmonics and therefore weaker definition of formants. Additionally, one can observe that pitch of a man's voice is lower than pitch of a woman's voice. Pitches and intensity are proportional to each other, therefore it is expected that women speak

higher than men. In general, speakers significantly differ among each other in term of prosodic patterns.

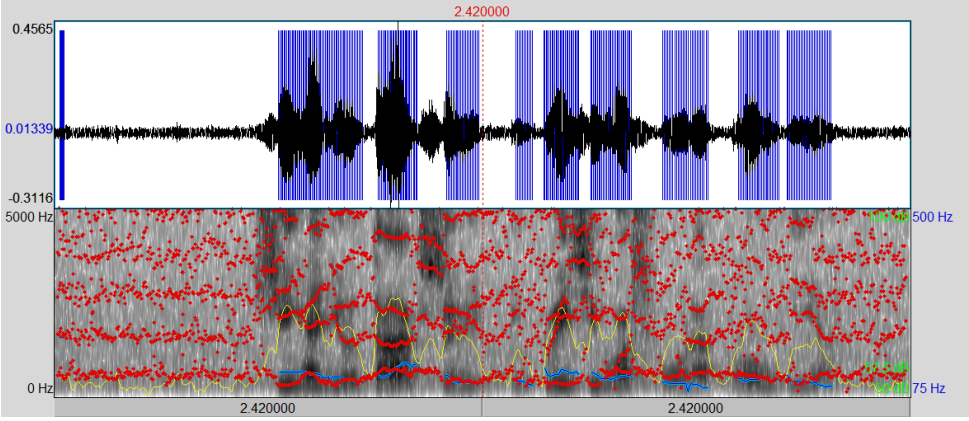

Figure 3.3. Speech signal in time and frequency domains for a male sample (and the same utterance as in Fig. 3.2). The lower part shows the spectrogram of the signal where formants (denoted in red), intensity (denoted in yellow) and pitches (denoted in blue) are extracted using the software PRAAT (Boersma and Weenink 2016).

3.4 Extraction of Mel-Frequency Cepstral Coefficients

As already noted, feature extraction may be described as a process of mapping an input waveform into acoustic feature vectors, each of which represents a short time frame of the signal. The Mel-Frequency Cepstral Coefficients are important for this discussion in so far as they are one of the most popular feature group in the field of speech and speaker recognition. The extraction process for the Mel-Frequency Cepstral Coefficients includes several phases, as denoted in Fig 3.4, and this section discusses all of them.

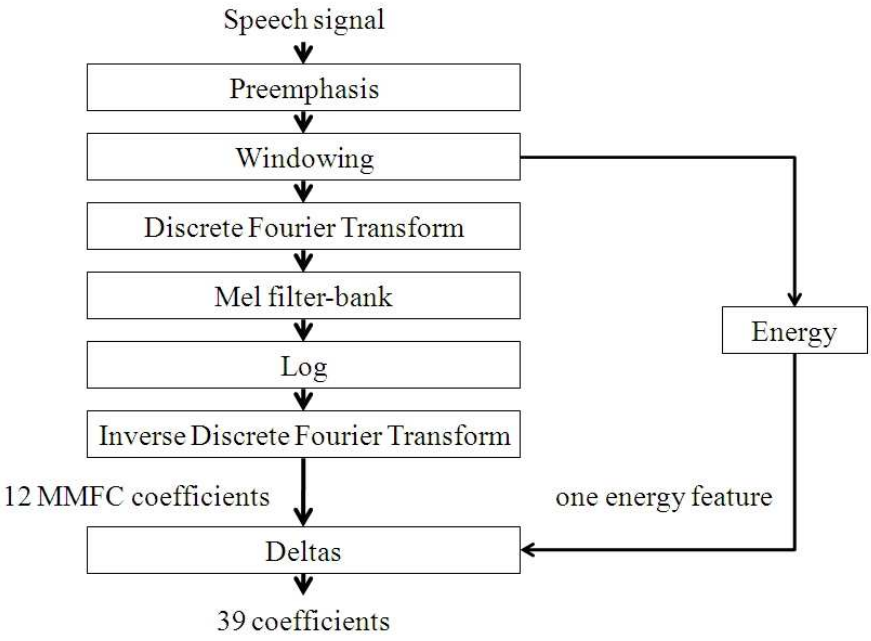

Figure 3.4. Block diagram illustrating the extraction of the Mel-Frequency Cepstral Coefficients (cf. Jurafsky and Martin 2009).

3.4.1 Preemphasis

A fundamental characteristics of speech is that lower frequencies are assigned more energy than the higher frequencies. Beigi (2011, pp. 154-156) notes that it has been estimated that approximately 80% of the power in a speech signal is related to frequencies lower than 1,000 Hz. In the range from 1,000 Hz to 8,000Hz, the power is reduced at an approximate rate $-12db/Octave^2$, while for frequencies greater than 8,000Hz it becomes practically insignificant. In order to make information from higher formants more available, the higher frequency energy is boosted, which is referred to as the preemphasis of the signal (Jurafsky and Martin 2009). Preemphasis is conducted on analog signals, before the sampling, e.g. applying a differentiator:

$$H_p(z) = 1 - \alpha z^{-1}, \tag{3.7}$$

where the value of parameter α is usually set somewhere between 0.95 and 0.97 (Beigi 2011, p. 155).

3.4.2 Sampling and quantization of a speech signal

Periodic sampling at a fixed frequency is usually applied for the purpose of sampling a speech signal. The sampling frequency is determined in accordance with the Sampling theorem: if the maximum frequency of a function is f_c (which is referred to as the Nyquist Critical Frequency), then the sampling rate should be greater or equal to $2f_c$, i.e. the function should be sampled at a period less or equal to $\frac{1}{2f_c}$. Such a selection of a sampling rate ensures that the observed signal can be reconstructed with a sufficient level of accuracy (Beigi 2011, pp. 79-81):

$$h(t) = \sum_{n=-\infty}^{\infty} h_n \frac{\sin(w_c t - n\pi)}{w_c t - n\pi}, \tag{3.8}$$

where:

- $h_n = h\left(\frac{n}{2f_c}\right)$,
- $w_c = 2\pi f_c$ is the Nyquist Critical Angular Frequency.

The theorem is illustrated in Fig. 3.5, for the sampling frequency set to *2f*, *3f*, *4f*, *8f*, respectively, where *f* is the signal frequency. It may be observed that sampling frequences lower than *2f* do not allow for complete reconstruction accuracy. A proof of the Sampling Theorem is provided by Beigi (2011, pp. 80-83). As illustration of practical applications, Jurafky and Martin (2009) note that frequencies of human speech are primarily below 10,000Hz, so the sampling rate of

20,000Hz is necessary for the accuracy. However, frequencies of telephone speech is under 4,000Hz, so the sampling rate for telephone speech is 8,000Hz, while the sampling rate for microphone speech is usually set to 16,000Hz. The sampled values are further quantized, e.g. stored as 8-bit or 16-bit integer values.

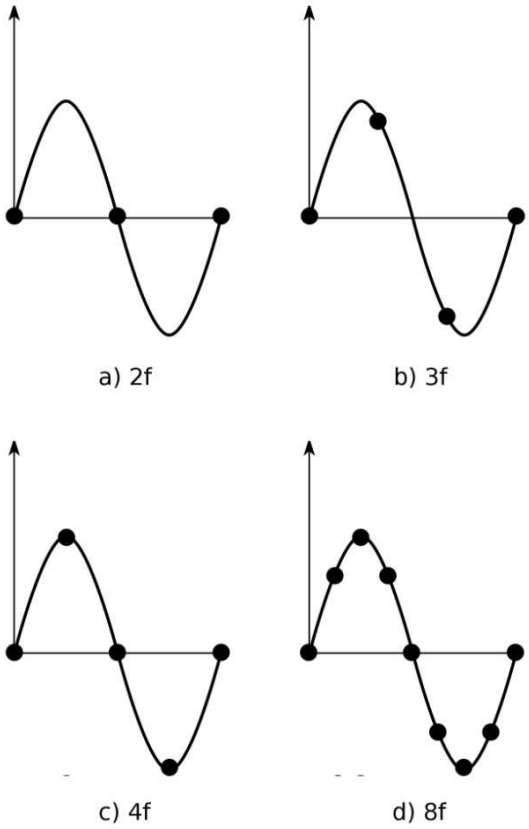

Figure 3.5. Illustration of the Sampling theorem. The sampling frequency is set to a) 2f, b) 3f, c)4f, and d) 8f, respectively, where f is the signal frequency. Sampling frequencies lower than 2f do not allow for complete reconstruction accuracy.

3.4.3 Framing and windowing a speech signal

Speech may be treated as a stationary signal over short time periods, and thus is usually segmented into frames for the purpose of analysis. Although these time periods range to 100ms, a typical frame width is much smaller (e.g. 20-30ms), and in addition frames are overlapped (e.g. shifted by 10ms), in order not to miss short speech phenomena such as stops, and onsets and offsets of phones (Beigi 2011, p. 161, Mølgaard and Jørgensen 2005, p. 2).

The windowing process can be described as multiplication of each frame by a windowing function (Beigi 2011, p. 162):

$$ {}_l\overline{h_n} = {}_l h_n w(n) \; (\forall \, 0 \le n \le N-1) \, , \tag{3.9} $$

where:

- N is the length of a frame (i.e. number of samples in a frame),
- ${}_l h_n$ is the n^{th} sample of the l^{th} frame,
- $w(n)$ is a window function.

Beigi (2011, p.162) makes a practical remark that frames positioned at the start or the end of a speech sequence do not necessarily contain data for each sample in a frame. Data in such frames are padded with zeros.

Table 3.1 gives contains definitions of selected window functions. Each of these window functions has its advantages and disdvantages, as discussed by Beigi (2011, p.163-167). For an illustration, Figures 3.6, 3.7 and 3.8 illustrate the Hamming window, the Hann window and the Triangualer window, respectively. The Hamming window is selected as the most popular, the Hann window as a variation of the Hammings window, and, finally, the Triangular window is

illustrated because it is used for the extraction of Mel-Frequency Cepstral Coefficients. The advantage of the the Hamming window is that its spectrum drops quickly, which in turn enables isolation. On the other hand, its side-lobes relating to the higher frequencies remain flat. When compared to the Haming window, it can be observed that the Hann window falls off more slowly at lower frequncies, but quickly at higher frequencies. In contrast to them, the Triangular window drops rapidly and has significantly wider side-lobes.

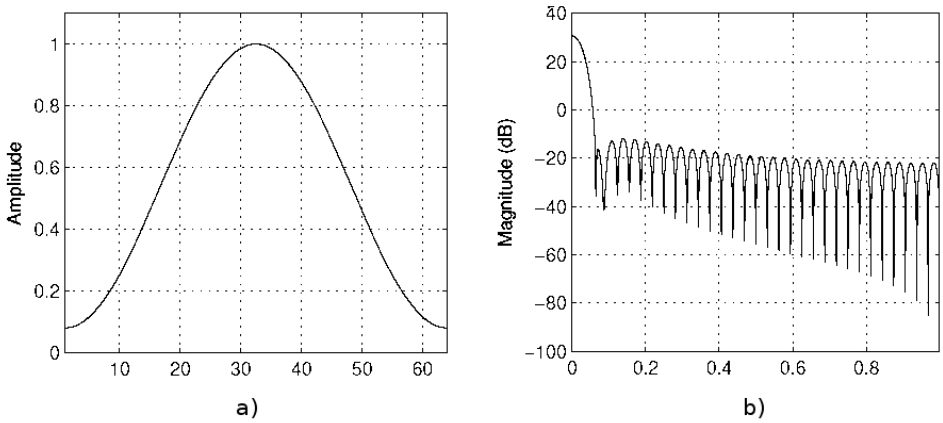

Figure 3.6. The Hamming window and its spectrum: a) time domain, b) frequency domain (adjusted from Beigi 2011, p.164).

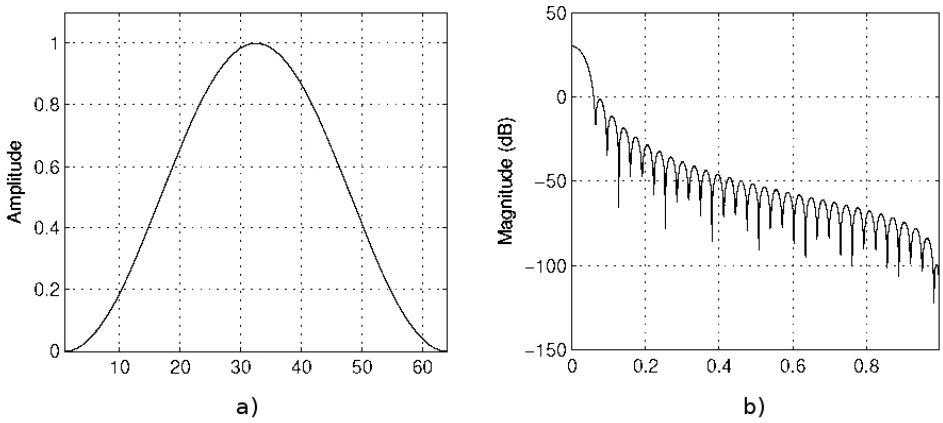

Figure 3.7. The Hann window and its spectrum: a) time domain, b) frequency domain (adjusted from Beigi 2011, p.164).

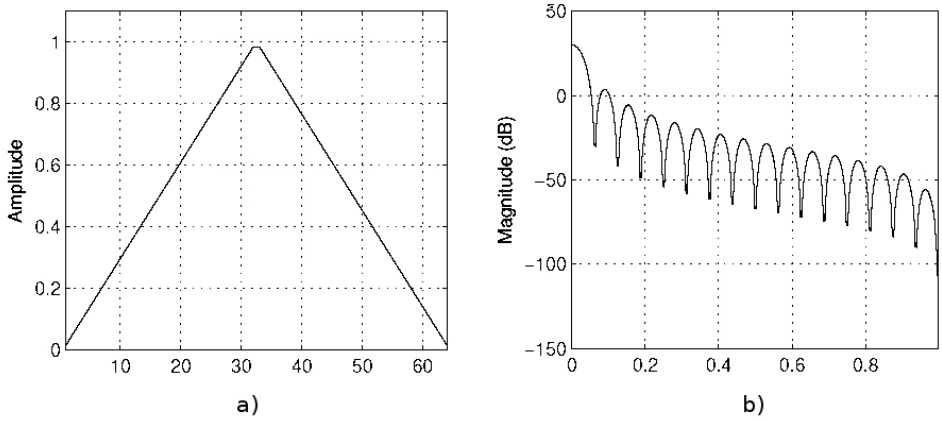

Figure 3.8. The Triangular window: a) time domain, b) frequency domain (adjusted from Beigi 2011, p.166).

3.4.4 Discrete Fourier Transform

For extracting spectral information from a windowed signal, the Discrete Fourier Transform (DFT) is applied. It is often based on the Fast Fourier Transform algorithm (Beigi 2011, pp. 167-168):

$$ {}_{l}H_{k} = \sum_{n=0}^{N-1} {}_{l}h_{n}\, w(n) \exp\left(-i\frac{2k\pi n}{N}\right) \quad (\forall\, 0 \le k \le N-1), $$

where:

- N is the length of a frame,
- ${}_{l}h_{n}$ is the n^{th} sample of the l^{th} frame,
- ${}_{l}H_{n}$ is the Descrete Fourier Transform of ${}_{l}h_{n}$,
- *w(n)* is a window function,

after which magnitudes of the calculated Fast Fourier Transform bins are determined:

$$\left|{}_l H_k\right| = \sqrt{\mathrm{Re}({}_l H_k)^2 + \mathrm{Im}({}_l H_k)^2}\ . \tag{3.10}$$

Table 3.1 Selected window functions. N is the length of a frame (cf. Beigi 2011, pp.163-167).

Window	***Definition***
Hamming window	$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$
Hann window	$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$
Welch window	$w(n) = 1 - \left(\frac{n - \frac{N-1}{2}}{\frac{N-1}{2}}\right)^2$
Triangular window	$w(n) = 1 - \left\|\frac{2n - N + 1}{N-1}\right\|$
Blackman window family	$w(n) = a_0 - a_1\cos\left(\frac{2\pi n}{N-1}\right) + a_2\cos\left(\frac{4\pi n}{N-1}\right)$, where: $a_0 = \frac{1-\alpha}{2}, a_1 = \frac{1}{2}, a_2 = \frac{\alpha}{2}$
Gauss window	$w(n) = e^{-\frac{1}{2}\left(\frac{n - \frac{N-1}{2}}{\sigma\frac{N-1}{2}}\right)^2}$, where $\sigma \le \frac{1}{2}$

3.4.5 Mel filter-bank and log

An important property of human hearing is less sensitive to frequencies greater than 1,000Hz. Mel-Frequency Cepstral Coefficients are intended to model this property. In line with this, the frequency scale is mapped onto the *mel* scale:

$$mel(f) = 1127 \ln\left(1 + \frac{f}{700}\right), \tag{3.11}$$

for frequencies below 1,000Hz, the mapping is linear, and for frequencies above 1,000Hz it is logarithmic. In this way, perceptually equidistant sounds are separated by equal number of mels (Jurafsky and Martin 2009).

In a typical implementation, a mel filter bank contains triangular filters that collect energy from different frequency ranges. Ten filters are set linearly for frequencies lower than 1,000Hz, while the others are set logarithmically for frequencies greater than 1,000Hz. A mel filter bank is illustrated in Fig. 3.9. The obtained values are represented as log-values.

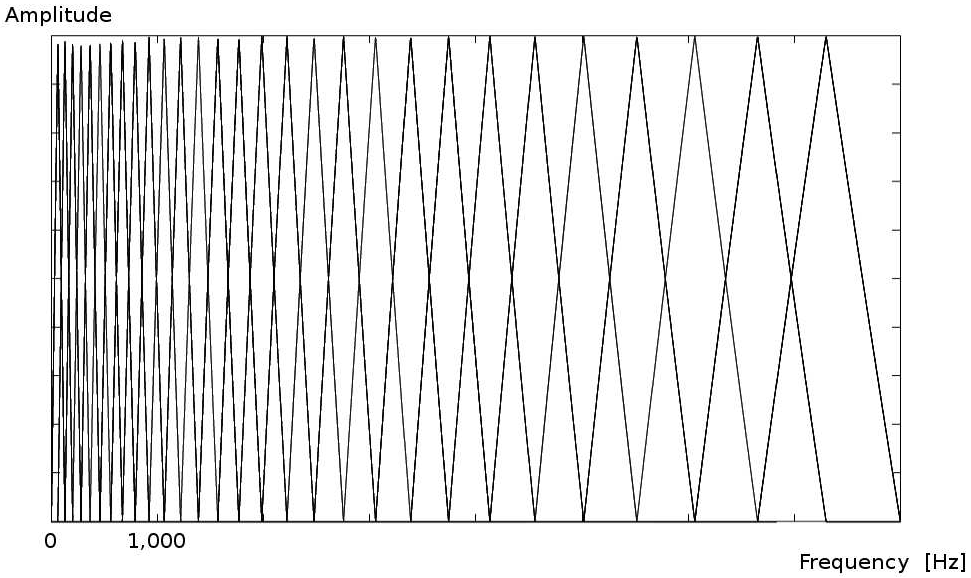

Figure 3.9. Illustration of a mel filter bank (adjusted from Mølgaard and Jørgensen 2005, p. 6).

3.4.6 Inverse Discrete Fourier Transform

The cepstrum is defined as the inverse Discrete Fourier Transform of the log magnitude of the Discrete Fourier Transform of a signal (Jurafsky and Martin 2009):

$$c[n]=\sum_{n=0}^{N-1}\log\left(\left|{}_{l}h_{n}\,w(n)\exp\left(-i\frac{2k\pi n}{N}\right)\right|\right)\exp\left(i\frac{2k\pi n}{N}\right), \tag{3.12}$$

where:

- N is the length of a frame,
- ${}_{l}h_{n}$ is the n^{th} sample of the l^{th} frame,
- $w(n)$ is a window function.

Typically, the first twelve cepstral values are used. These coefficients carry information on the vocal tract filter, excluding information on the glottal source, and have a practical advantage that their variance is uncorrelated (Jurafsky and Martin 2009).

3.4.7 Energy and delta coefficients

The set containing twelve Mel-Frequency Cepstral Coefficients per frame calculated by the Inverse Discrete Fourier Transform can be further extended. First, it is extended by the energy coefficient (i.e. energy from the frame) that is calculated as follows (Jurafsky and Martin 2009):

$${}_{l}energy=\sum_{n=0}^{N-1}\left({}_{l}h_{n}\,w(n)\right)^{2}, \tag{3.13}$$

where

- N is the length of a frame,
- ${}_{l}h_{n}$ is the n^{th} sample of the l^{th} frame,
- $w(n)$ is a window function.

In addition, the dynamics of Mel-Frequency Cepstral Coefficients are reflected through the first and second order differences of these thirteenth features. The first order difference is also called a *delta* or *velocity* feature, while the second order difference is called a *double delta* or *acceleration* feature.

Deltas can be computed in a simple manner, e.g.: the delta value of a cepstral value c at time t may be obtained as (Jurafsky and Martin 2009):

$$d(t) = \frac{c(t+1) - c(t-1)}{2}, \tag{3.14}$$

or it can be estimated in a more advanced manner.

In total, there are 39 Mel-Frequency Cepstral Coefficients, including twelve cepstral coefficients, twelve delta cepstral coefficients, twelve double delta cepstral coefficients, one energy coefficient, one delta energy coefficient and one double delta energy coefficient (Jurafsky and Martin 2009). Beigi notes that for practical purposes (e.g. for the purpose of optimization), smaller number of delta and double delta coefficients can be applied without significant decries in recognition performance (Beigi 2011, p. 175-176).

3.5 Conclusion

This chapter discussed different phases of the feature extraction process for the purpose of speaker recognition. Thus, in the context of the statistical approaches to automatic speaker recognition, discussed in Section 2.2 of the previous chapter, the Mel-Frequency Cepstral Coefficients represent input for Gaussian Mixture Models. However, these features can serve as input in a neural network model, which is discussed in the next chapter.

Chapter 4
Learning with neural networks

4.1 Introduction

The previous chapter described the feature extraction for the purpose of automatic speaker recognition. Here we discuss additional aspect of our approach to speaker recognition. Namely, the observed acoustic features serve as input to a neural network model for speaker classification. Therefore, we first consider the theoretical concepts and methods that are relevant to neural networks and deep learning, and then report on how we apply neural network learning for the purpose of automatic speaker recognition (Nasef et al. 2017a).

The structure of this chapter is as follows. Section 4.2 introduces the basic notions of a sigmoid neuron and a deep feedforward neural network. Section 4.3 discusses the stochastic gradient descent algorithm for determining appropriate values of weights and biases in the process of the training a deep feedforward neural network. A general algorithm for stochastic gradient update at a training iteration is given. Section 4.4 introduces the backpropagation algorithm for computing gradient of a cost function. Section 4.5 discusses the problem of overfitting a neural network, and one of the regularization methods intended to address this problem, i.e. the dropout method. Section 4.6 briefly describes convolutional neural networks and three important underlying concepts, i.e. sparse interactions, parameter sharing, and equivariant representations. Finally,

Section 4.7 introduces our approach to automatic speaker recognition based on deep neural networks and the stochastic gradient descent algorithm. Section 4.8 concludes this chapter.

4.2 The notions of sigmoid neuron and deep feedforward neural network

One of the most fundamental conceptualizations in the field of neural networks relates to a model of artificial neuron (Nielsen 2015, cf. the first chapter). The basic model of artificial neuron is a *perceptron*. In its basic form, a perceptron takes a set of binary values as its input $\{x_1, x_2, \ldots, x_n\}$, and generates a binary output o, as illustrated in fig. 4.1.

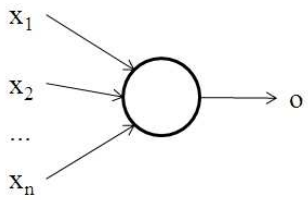

Figure 4.1: Illustration of a perceptron.

Each input value x_i, $1 \le i \le n$, is assigned a real-valued weight $w_i \in \Re$, $1 \le i \le n$, that reflects the importance of variable x_i to the output o. The output is defined as follows:

$$o = \begin{cases} 0, & w \cdot x + b \le 0, \\ 1, & otherwise, \end{cases} \tag{4.1}$$

where:

- x is the vector of inputs $x_1, x_2, \ldots, x_n$,
- w is the vector of weights $w_1, w_2, \ldots, w_n$,

- $w \cdot x$ is a dot product of vectors x and w, i.e., $\sum_{1 \le i \le n} w_i x_i$.

- $b \in \Re$ is a real-valued bias assigned to the perceptron.

The output of a perceptron is always a binary value, i.e. either zero or one. This means that even a small change in input bias or weights can complement the output. For practical purposes of machine learning, it is often more convenient to have a more-fine grained output. To achieve such an output, the *sigmoid neuron* model is introduced. The sigmoid neuron model is an extension of the perceptron model of artificial neuron that allows the output to take value from zero to one. The output of a sigmoid neuron is defined by the so-called *sigmoid function*:

$$\sigma(wx+b) = \frac{1}{1+e^{-(wx+b)}} = \frac{1}{1+\exp(-\sum_{1 \le i \le n} w_i x_i - b)} \tag{4.2}$$

Fig. **4.2** illustrates the step function relating to the output of a perceptron, and the sigmoid function relating to the output of a sigmoid neuron. It may be observed that the perceptron model is just a special case of the sigmoid neuron model. In comparison to the step function, the sigmoid function is rather smooth, which enables a more-fine grained output. The main idea of the sigmoid neuron model is that small changes in weights and bias produce a small change in output.

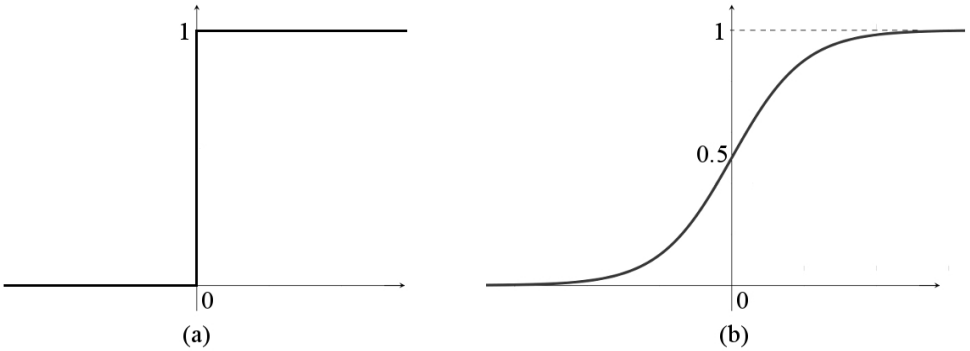

Figure 4.2: Illustration of (a) the step function and (b) the sigmoid function.

The output of a sigmoid function may be approximated as follows:

$$\Delta o \approx \sum_{1 \le i \le n} \frac{\partial o}{\partial w_i} \Delta w_i + \frac{\partial o}{\partial b} \Delta b \,, \tag{4.3}$$

where $\frac{\partial o}{\partial w_i}$ and $\frac{\partial o}{\partial b}$ represent partial derivates. This also implies that the change in output may be approximated as a linear function of changes in bias and weights. This property of a sigmoid neuron is important for neural network training. This is discussed in the following sections.

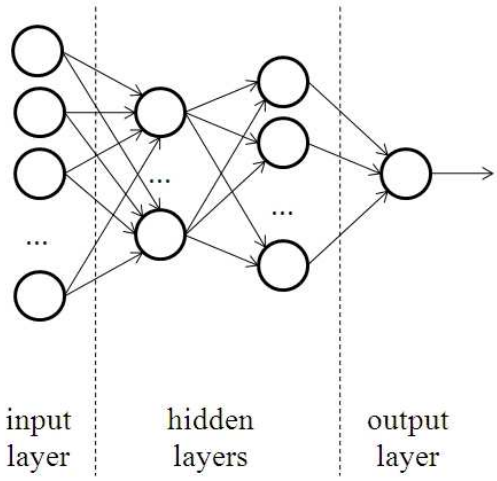

Figure 4.3: Direct, acyclic graph illustrating a deep feedforward neural network.

At the structural level, a neural network consists of a set of neurons whose organization is illustrated by a directed graph given in fig. 4.3. Each node in this graph represents a neuron. The output of the first layer of neurons serves as the input for the second layer of neurons, the second layer of neurons produces the input for the third layer of neurons, and so on. This transition of information through the network is denoted by directed edges. Neural networks that contain no loops (i.e. no feedback connections) are called *feedforward neural networks*. They may be represented as a directed, acyclic graph, as already shown in fig. 4.3. The length of a path

starting at a node in the input layer and ending at a node in the output layer defines the depth of the model (hence the adjective *deep* in *deep feedforward networks*).

The main purpose of deep feedforward neural networks is to approximate a given function $f^*(x)$ (Goodfellow et al. 2016, pp. 100, 168–169; Nielsen 2015, cf. the first chapter). Without loss of generality, let us assume that function $f^*(x)$ classifies input x to one of given categories, e.g., $f^*: \Re^n \to \{1,2,\ldots,k\}$. The task of a deep feedforward neural network is to define function $f(x,\theta)$ and to learn the value of θ, so that function f approximates sufficiently well function f^*.

4.3 The stochastic gradient descent algorithm

As stated above, the training of a deep feedforward neural network may be briefly described as finding weights and biases so that the output approximates, as good as possible, a given function for all training inputs. To assess how well a neural network approximate a given function, a cost function is applied, which may be defined as an average over the training set, e.g. (Goodfellow et al. 2016, pp. 275–276):

$$J(\theta) = \mathrm{E}_{(x,y)\sim \mathrm{p}_{\mathrm{data}}} L(f(x,\theta), y), \tag{4.4}$$

where:

- L represents a loss function,
- $f(x,\theta)$ represents the output from a given network when the input is x,
- y represents function that should be approximated,

- p_{data} represents the empirical data distribution.

In the context of machine learning, the true data distribution is usually not available, so the empirical data distribution is used. The training of a neural network is based on minimizing the cost function $J(\theta)$, i.e. (Goodfellow et al. 2016, pp. 275–276):

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}, \theta), y^{(i)}), \tag{4.5}$$

where m represents the number of training samples. This training process is also called *empirical risk minimization*. To illustrate this, let as assume that we apply the quadratic cost function to assess how well a neural network approximate a given function (Nielsen 2015, cf. the first chapter):

$$C(w,b) = \frac{1}{2n} \sum_{x} \left\| y(x) - a \right\|^2, \tag{4.6}$$

where:

- y represents function that should be approximated,
- w represents all weights in neural network,
- b represents all biases in neural network,
- x represents all training inputs,
- a represents the vector of neural network outputs corresponding to given inputs.

When the cost $C(w,b)$ is close to zero, it means that function defined by the observed neural network is approximatelly equal to function y.

To determine appropriate values of weights and biases, the *gradient descent* algorithm may be applied (Nielsen 2015, cf. the first chapter). For the purpose of explanation, we assume that the cost function has a set of real-valued variables $v = v_1, v_2, \ldots v_m$. The main idea of the algorithm may be briefly described as follows. We may randomly choose a starting point $(v_1, v_2, \ldots, v_m)$ in the observed vector space. Than we can modify each variable v_i by a small amount Δv_i, which produce a small change of the value of the cost function:

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2 + \cdots + \frac{\partial C}{\partial v_m}\Delta v_m. \tag{4.7}$$

If we choose $\Delta v_1, \Delta v_2, \ldots, \Delta v_m$ such that ΔC is negative, and apply a sequence of such modifications, we may eventually reach a global minimum of the cost function. A vector representation of equation (4.7) is:

$$\Delta C = \nabla C \cdot \Delta v, \tag{4.8}$$

where:

- $\Delta v = (\Delta v_1, \Delta v_2, \ldots, \Delta v_m)^T$ is the vector of changes in v,
- $\nabla C = (\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}, \cdots, \frac{\partial C}{\partial v_m})^T$ is the gradient vector of the cost function.

To assure that the change of the value of the cost function is negative, we may choose:

$$\Delta v = -\eta \nabla C, \tag{4.9}$$

where η is a small positive value, called *the learning rate*. Now we have:

$$\Delta C = -\eta \|\nabla C\|^2, \tag{4.10}$$

which implies that $\Delta C \leq 0$. Thus, in each iteration of the algorithm, the gradient vector ∇C of the cost function is computed, and then the vector of changes Δv is determined, according to the equation:

$$v' = v - \eta \nabla C, \tag{4.11}$$

so that the value of the cost function decreases with each iteration. The selected value of the learning rate is also important. It should be small enough, in order that the algorithm could work properly, but it must not be too small because it could significantly slow down the algorithm's performance.

To illustrate this, we go back to the above example with the quadratic cost function. The last equation can be reformulated to reflect the changes in weights and biases:

$$w'_k = w_k - \eta \frac{\partial C}{\partial w_k}, \tag{4.12}$$

$$b'_l = b_l - \eta \frac{\partial C}{\partial b_l}. \tag{4.13}$$

However, it is important to note that to compute the gradient vector ∇C of the cost function, it is mandatory to compute the gradients of each term in the sum given in equation (4.6), separately for each training input. This is time consuming when we have a very large number of inputs. To improve the efficiency, the gradients of each term in the sum given in equation (4.6) are computed only for a restricted sample or randomly selected inputs, instead for each training

input. A set of randomly selected training inputs is called *mini-batch*, and this modification of the algorithm is known as *stochastic gradient descent*.

Require: Learning rate η.

Require: Initial parameter θ.

while not(stopping criterion) do

Select a mini-batch containing m training inputs: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ with corresponding targets $\{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$.

Compute gradient: $g \leftarrow + \frac{1}{m} \nabla\theta \sum_i L(f(x^{(i)}, \theta), y^{(i)})$.

Update θ: $\theta \leftarrow \theta - \eta g$.

end while

Figure 4.4: A general algorithm for stochastic gradient update at training iteration k (Goodfellow et al. 2016, p. 294).

Determining the size of a mini-batch is a trade-off. Goodfellow et al. (2016, p. 279) discuss that applying batches of larger size can result in a more accurate, but less than linear estimate. In addition, some hardware architectures, such as multicore architectures, graphical processing units, and architectures for parallel processing, imply restrictions on the size of a mini-batch – they are either very small or of a fixed size. A general algorithm for stochastic gradient update at training iteration k is given in fig. 4.4 (Goodfellow et al. 2016, p. 294).

4.4 The backpropagation algorithm

The backpropagation algorithm is a fast algorithm for computing gradient of a cost function. To introduce this algorithm, we first adopt the following notation (Nielsen 2015, cf. the second chapter):

- w_{jk}^{l} – weight of the edge starting at the k^{th} neuron in the $(l-1)^{th}$ layer and ending at the j^{th} neuron in the l^{th} layer,
- b_{j}^{l} – bias of the j^{th} neuron in the l^{th} layer,
- a_{j}^{l} – activation of the j^{th} neuron in the l^{th} layer,

which is also illustrated in fig. 4.5.

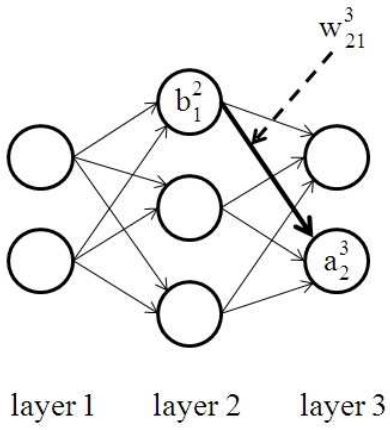

Figure 4.5: Illustration of the adopted notation for the backpropagation algorithm (Nielsen 2015, cf. the second chapter).

Recalling equation (4.2), the activation a_j^l can now be represented as the following sum over neurons in the $(l-1)^{th}$ layer:

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l). \tag{4.14}$$

In a vector form, the above equation is:

$$a^l = \sigma(w^l a^{l-1} + b^l) = \sigma(z^l), \tag{4.15}$$

where:

- w^l – the weight matrix for the l^{th} layer,
- b^l – the bias vector for the l^{th} layer,
- a^l – the activation vector for the l^{th} layer.

The parameter of function σ in the above equation, i.e. $z^l = w^l a^{l-1} + b^l$, is known as the *weighted input* of the l^{th} layer. For each j^{th} neuron in the l^{th} layer, the *error of neuron* is defined as:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}, \tag{4.16}$$

while, in line with the adopted notation, δ^l represents the error vector for the l^{th} layer.

Here we briefly describe the main idea of the backpropagation algorithm, while more detailed explanations of this algorithm, including selected aspects of its implementation, are provided by Nielsen (2015, cf. the second chapter), Goodfellow et al. (2016, pp. 180–192), and others. Let as

assume that a small change Δw_{jk}^{l} is made in the weight of the edge starting at the k^{th} neuron in the $(l-1)^{th}$ layer and ending at the j^{th} neuron in the l^{th} layer. It affects the output of the latter neuron and produces a small change in it, Δa_j^l. However, this change affects all activations in the next layers, and eventually, in the output layer and the cost function. If we consider only one path of activations starting at the j^{th} neuron in the l^{th} layer and ending at the output node a_m^L, e.g., $a_j^l, a_q^{l+1}, \ldots, a_n^{L-1}, a_m^L$, it produces the following change in the cost function: $\frac{\partial C}{\partial a_m^l}\frac{\partial a_m^l}{\partial a_n^{L-1}}\frac{\partial a_n^{L-1}}{\partial a_p^{L-2}}\cdots\frac{\partial a_q^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial w_{jk}^l}\Delta w_{jk}^l$. The total change in the cost function, calculated for all available paths, may be approximated as:

$$\Delta C = \sum_{mnp\ldots q} \frac{\partial C}{\partial a_m^l}\frac{\partial a_m^l}{\partial a_n^{L-1}}\frac{\partial a_n^{L-1}}{\partial a_p^{L-2}}\cdots\frac{\partial a_q^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial w_{jk}^l}\Delta w_{jk}^l \,. \tag{4.17}$$

Thus, a way to compute $\frac{\partial C}{\partial w_{jk}^l}$ is to analyze how a small change Δw causes a small change ΔC. Four equations that underlay the backpropagation algorithm are (Nielsen 2015, cf. the second chapter):

$$\delta^L = \nabla_a C * \sigma'(z^L), \tag{4.18}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) * \sigma'(z^l), \tag{4.19}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l, \tag{4.20}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l, \tag{4.21}$$

where $*$ is the element-wise product, and $\nabla_a C$ in equation (4.18) is a vector whose elements $\frac{\partial C}{\partial a_j^L}$ is the rate of change of the cost function. A general backpropagation algorithm is given in fig. 4.6 (Nielsen 2015, cf. the second chapter).

1. **Input:** a^1 , the activation of the input layer
2. **Feedforward:**

   for l = 2 to L

   Compute $z^l = w^l a^{l-1} + b^l$

   Compute $a^l = \sigma(z^l)$

   end for
3. **Output error:**

   Compute $\delta^L = \nabla_a C * \sigma'(z^L)$
4. **Backpropagate the error**

   for l = L-1 down to 2

   Compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) * \sigma'(z^l)$

   end for
5. **Output (i.e. the gradient of the cost function):**

   $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l$ and $\frac{\partial C}{\partial b_j^l} = \delta_j^l$.

Figure 4.6: A general backpropagation algorithm (Nielsen 2015, cf. the second chapter).

Finally, there are two assumptions about the cost functions that are needed for the backpropagation algorithm. The first assumption is that the cost function can be represented as an average $C = \frac{1}{n}\sum_x Cx$. This assumption allows for computing of partial derivates for a single training input, and then averaging them over a training corpus. The second assumption is that the cost function can be represented as a function of the output activations a^l.

4.5 The problem of overfitting a neural network and the dropout method

Mathematical models, including neural networks, with a large number of free parameters can successfully cover diverse phenomena (Nielsen 2015, cf. the third chapter). A large number of free parameters allows for adjusting the observed model to describe diverse data sets. However, although such a model works successfully for the given training data, it may perform poor for new test data. This problem is known as *overfitting*.

The dropout method is a regularization method intended to address this problem. In this method, we train the ensemble containing all sub-networks derived from the observed neural network by removing non-output nodes, as illustrated in fig. 4.7, where dashed nodes represent dropout neurons (Goodfellow 2016 et al., pp.258– 259). For the purpose of this discussion, we reduce the process of removing a node from a network to multiplication of its output by zero. Each time we select a mini-batch, we also select a binary mask that is applied to all input and hidden nodes in the observed network in order to derive a sub-network. A mask is selected randomly and independently from other masks. Usually, an input node is included in a sub-network with probability 0.8, and a hidden node is included with probability 0.5. In any case, these probability values are set in advance, and remain constant through the training. After applying a binary mask, the training is performed as ordinarily, including forward propagation, back-propagation, and the learning update.

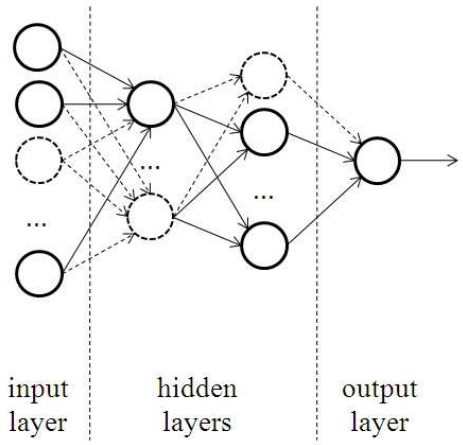

Figure 4.7: Illustration of the dropout method. The dashed nodes represent dropout neurons.

Let μ be a mask vector, and $J(\theta,\mu)$ the cost of the model derived when μ is applied. The dropout training of a neural network can be described as minimizing the average over the training set, i.e., $\mathrm{E}_{\mu} J(\theta,\mu)$. For a given neural network, we can derive exponentially many models, and it is not possible to train all of them. However, it is not necessary, because all models derived from the starting neural network share parameters. Thus, only a small portion of all possible sub-networks are trained, each of which for a single step (Goodfellow et al. 2016, pp.258–259). It should be noted that although different derived sub-networks may overfit, the averaging presented in the dropout method reduces the overfitting of a neural network (Nielsen 2015, cf. the third chapter).

4.6 Convolutional neural networks

Convolutional neural networks represent a class of neural networks suitable for processing time-series data, such as one-dimensional audio signals, two-dimensional image data, and other data with grid-like structure. More formally, a convolutional neural network is a neural network that

uses the convolution mathematical operation instead of general matrix multiplication in at least one of its layers (Goodfellow et al. 2016, pp.330– 339).

Convolution is an operation of two functions. It will be illustrated here for two functions, both of which take one real-valued argument. Let $x(t)$ be a noisy function that provides a single real-valued output at any time t. A way to improve a better estimation of the output of this function is to calculate its weighted average, i.e. more weight will be assigned to recent measurements. Thus, let $w(a)$ be a weight function, where a is the age of a measurement. In this example, convolution is represented by the function (Goodfellow et al. 2016, p.331):

$$s(t) = \int x(a)w(t-a)da\,, \tag{4.22}$$

which is usually written as:

$$s(t) = (x * w)(t)\,. \tag{4.23}$$

A discretized form of the convolution equation, which is more suitable for managing discrete, computer-generated time-series data is (Goodfellow et al. 2016, p.332):

$$s(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a)\,. \tag{4.24}$$

When convolution is applied to a neural network, function x represents the input, while function w represents the kernel. The input and the kernel are usually represented as multidimensional arrays of parameters. Thus, for a two-dimensional input that represents image I, the above equation may be reformulated as (Goodfellow et al. 2016, p.332):

$$s(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)\,, \tag{4.25}$$

where K is two-dimensional kernel, which is illustrated in fig. 4.8.

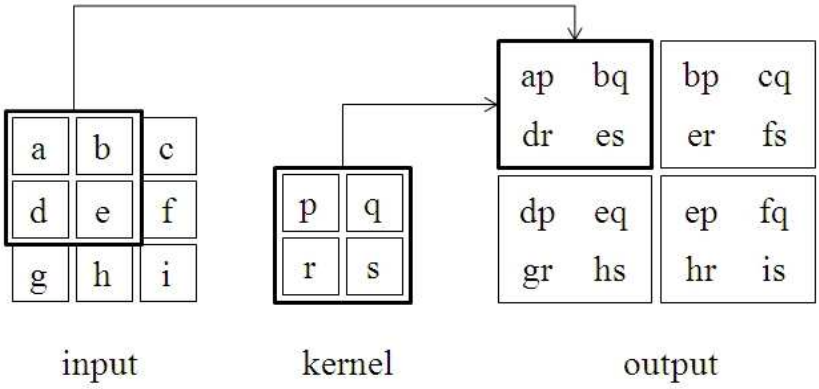

Figure 4.8: Illustration of two-dimensional discrete convolution (Goodfellow et al. 2016, p.334).

Three important concepts that underlay the convolution operation are: *sparse interactions*, *parameter sharing*, and *equivariant representations*, and they will be briefly discussed (Goodfellow et al. 2016, pp.335–339).

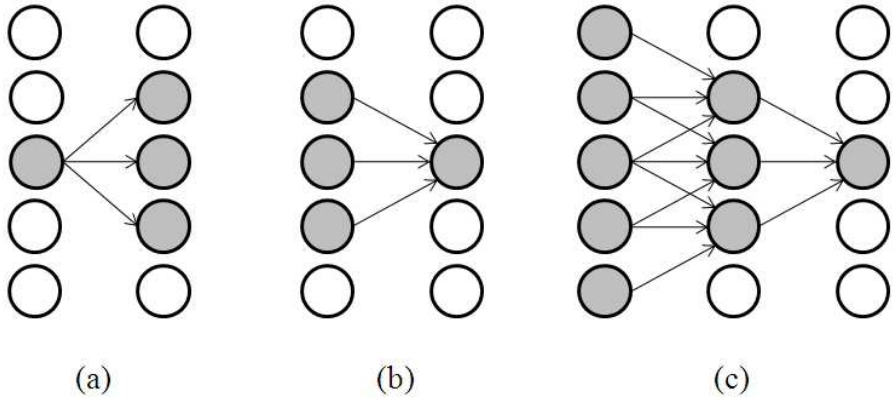

Figure 4.9: Illustration of sparse interactions in convolutional neural networks. (a) Each input neuron directly affects only *k* output neuron. (b) Each output neuron is directly affected by only *k* input neuron. (c) Indirect interaction between neurons is possible (Goodfellow et al. 2016, pp.336–337). The affecting and affected nodes are gray. Only relevant interactions are represented.

Sparse interactions. In traditional neural networks, each input node interacts with each output neuron, which is represented through a time-consuming matrix manipulation. In contrast to this, convolution neural networks improve the efficiency of the learning process by using a kernel that is significantly smaller than the input. This means that a convolutional network is not fully connected, but that each output has only a limited number of connections. Practically, the number of connections is significantly reduced when compared with the original number of connections in a fully-connected neural network, which improves the learning performance. It should be also noted that a restricted number of connections still does not prevent nodes at different levels to interact indirectly. This is illustrated in fig. 4.9.

Parameter sharing. The parameter sharing in convolutional neural networks means that each member of the kernel is applied to each input position. In other words, the same set of parameters is learned for all positions, which further reduces the memory-consumption of the learning algorithm.

Equivariant representations. An important property of the convolution function is that it is equivariant to translation of the input. E.g. in the context of processing time series data, if an event is shifted later in time in the input, its representation will be also shifted in the output. Or in the context of image processing, if an object is translated in the input, its representation will be also translated by the same amount in the output. More generally, the equivariance of the convolution function means that the input and the output change in the same way.

4.7 The introduced approach to speaker recognition with neural networks

In this section, we introduce our approach (Nasef et al. 2017a) to automatic speaker recognition based on deep neural networks and the stochastic gradient descent algorithm (Bottou 2010, Robinds and Monro 1951).

Consider that $\left\{x^{(i)}, y^{(i)}\right\}$ represents a training set, where $x^{(i)} \in R^{Nf}$ is a vector of extracted features for the i^{th} sample, while vector $y^{(i)}$ represents its associated class. The objective is to obtain decision function $h(x)$ in order to approximate y.

This function can be represented as a linear function of the extracted features:

$$h(x;\theta,b) = \theta_1 x_1 + \theta_2 x_2 + ... + \theta_{Nf} x_{Nf} + b \tag{4.26}$$

This equation can also be presented in a vector form as: $h(x;\theta,b) = \theta^T x + b$.

In order to limit the decision function $h(x)$ so that its output value is always in the range [0,1], it is mapped by another function known as *rectified linear units function*, i.e.:

$$g(z) = \log(1 + e^z)\,. \tag{4.27}$$

Therefore, the decision function from (4.26) can be presented in a form:

$$h(x;\theta,b) = g(\theta^T x + b)\,, \tag{4.28}$$

where $g(\theta^T x + b) = \log(1 + e^{(\theta^T x + b)})$ represents the rectified linear units function.

The objective is to obtain parameters θ and b in order to minimize function $(h(x;\theta,b) - y)^2$. This is achieved by updating the parameters θ and b in the following way:

$$\theta_j = \theta_j - \alpha \Delta\theta_j,\ 1 \le j \le N_f\,, \tag{4.29}$$

$$b = b - \alpha \Delta b\,, \tag{4.30}$$

where:

$$\Delta\theta_j = 2\left[g(\theta^T x+b)-y\right]\left[1-g(\theta^T x+b)\right]g(\theta^T x+b)x_j\,, \tag{4.31}$$

$$\Delta b = 2\left[g(\theta^T x\ +b)-y\ \right]\left[1-g(\theta^T x\ +b)\right]g(\theta^T x+b) \tag{4.32}$$

represent the partial derivatives of θ_j and b, respectively, and α represents the learning rate.

In the literature, there is no strict rule how to pick the value for this parameter. If the learning rate is too large, it can speed up learning, but can also change the parameters too aggressively. On the other hand, if it is too small, it can change the parameters too conservatively. Robinds and Monro (1951) recommended that in order to select a good value for the learning rate, the progress of the training should be monitored.

The algorithm for stochastic gradient descent is given in fig. 4.10 (Nasef et al. 2017a).

Require: θ,b : input random variables

Require: $\{x^{(i)}, y^{(i)}\}$: training set

Ensure: N : number of labeled samples

Ensure: α : input learning rate

for i = 1 **to** N

Compute $g(\theta^T x^{(i)}+b) = \log(1+e^{(\theta^T x^{(i)}+b)})$

Compute $\Delta\theta_j$ and Δb according to equations (4.31) and (4.32)

Update θ and b according to equations (4.26) and (4.28)

end for

Figure 4.10: The algorithm for stochastic gradient descent (Nasef et al. 2017a).

Overfitting of deep neural networks may cause a big problem. This can be solved using input and hidden layer dropout (Srivastava et al. 2014). The basic idea is to drop out random units of neural network during the training. This is achieved by retaining the hidden unit with determined probability *p*. This probability is called *probability of retaining*. This parameter controls the density of dropout. The higher the value *p*, the less dropout is applied. Selection of value is a difficult task and depends on dataset that is used.

For the sake of simplicity, we use a neural network without dropout, and set its weights to the trained weights multiplied by *p*. This model is verified by the fact that the outgoing weights of a neuron that was retained with probability *p* are multiplied by *p*. Fig. 4.11 illustrates the adopted dropout neural network model, where vector $r^{(j)}$ contains independent Bernoulli random variables for the j^{th} hidden unit, i.e. $r^{(j)}$~Bernoulli(p) and $*$ is the element-wise product (Srivastava et al. 2014).

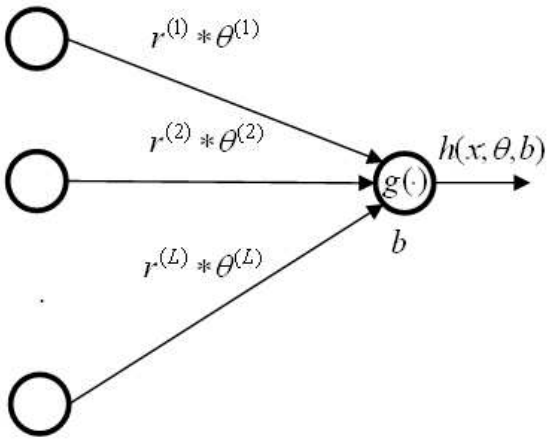

Figure 4.11: Dropout Neural Network model. The weights are multiplied by *p* (*L* is the number of hidden units) (Srivastava et al. 2014, Nasef et al. 2017a).

As discussed above, the difference between the stochastic gradient descent algorithms with and without dropout is that for each training input in the latter case we create a subnetwork by

applying drop out of random units. In this case, the decision function defined by (4.28) may be presented as:

$$h(x;\theta,b) = g(r^{(j)} * \theta^T x + b) = \log(1 + e^{(r^{(j)} * \theta^T x + b)}) \tag{4.33}$$

The algorithm for stochastic gradient descent with dropout is given in fig. 4.12 (Nasef et al. 2017a).

Require: θ, b : input random variables

Require: $\{x^{(i)}, y^{(i)}\}$: training set

Ensure: N : number of labeled samples

Ensure: α : input learning rate

Ensure: p: probability of retaining

for i = 1 **to** N

Compute $r^{(j)} \sim \text{Bernoulli}(p)$

Compute $g(r^{(j)} * \theta^T x^{(i)} + b) = \log(1 + e^{(r^{(j)} * \theta^T x^{(i)} + b)})$

Compute $\Delta\theta_j$ and Δb according to equations (4.31) and (4.32)

Update θ and b according to equations (4.26) and (4.28)

end for

Figure 4.12: The algorithm for stochastic gradient descent with dropout (Nasef et al. 2017a).

4.8 Conclusion

This chapter and the previous chapter introduced the theoretical foundations of our approach to automatic speaker recognition. In order to demonstrate the appropriateness of the introduced approach under realistic conditions, we conducted two classification experiments using a spoken corpus. The next chapter reports on the experimental settings of the conducted experiments and discusses the obtained results.

Chapter 5
Experiments and results

5.1 Introduction

This chapter reports on two experiments conducted in the scope of this thesis. Both experiments relate to the model of stochastic gradient descent proposed in the previous chapter. In the first experiment, the speaker recognition performance is observed when optimizing the parameters of the stochastic gradient descent algorithm: the learning rate, and the hidden and input layer dropout rates. The second experiment basically focuses on the improvement of speaker identification in noisy environment using deep neural networks with stochastic gradient descent. We analyze how different combinations of its parameters, such as the learning rate and the dropout rate, influence automatic speaker recognition performances when different noise levels are applied on the original speech signal.

The chapter is organized as follows. Section 5.2 describes the VidTIMIT corpus that was used in both experiments. Sections 5.3 and 5.4 report on the experimental settings and results. Section 5.5 concludes the chapter.

5.2 The corpus

VidTIMIT is a multimodal corpus produced for the purposes of speech recognition and person authentication (Sanderson 2002). It contains video and audio recordings of 43 subjects (19 female, 24 male) uttering short sentences in a noisy office environment, with mean duration 4.25 seconds per sentence. In further text, only details relevant to audio recordings are discussed. Each subject uttered ten sentences, two of which were common for all subjects, and the other sentences were different for each subject. All sentences were selected from the NTIMIT corpus (Jankowski et al. 1990). The production of the VidTIMIT corpus was conducted in three phases. The first session may be considered as the training set, and the other two sessions as the test set.

It is important to note that there is no sentence overlapping between these sets. This is an important requirement, because if a training sentence was also included in a test set, it would cause the following bias reflected in higher probability assigned to the overlapped sentence and inaccuracy in perplexity. This bias is also referred to as training on the test set (Jurafsky and Martin 2009). In addition, the mean delay between the first and the second phases was seven days, and six days between the second and the third phases. These delays were introduced to make a possibility for change in a subject's voice (e.g. due to the change of mood, etc.). The audio recordings were produced as mono 16 bit, 32 kHz WAV files (Sanderson 2002, Sanderson and Paliwal 2002).

5.3 Experiment 1

In this experiment (cf. Nasef et al. 2017a), a deep neural network with stochastic gradient descent is trained for a classification problem on a data set in the domain of speaker recognition. For the sake of simplicity, the neural network uses only one hidden layer with L=100 units. The experiment was conducted in two phases.

In the first phase, the recognition rate is observed when the hidden layer dropout rate and learning rate are varied. The dropout is applied at all layers. The hidden layer dropout rate was varied from 0.0 to 0.9, with the step of 0.1, while the input layer dropout rate was set to 0.2. The learning rate was varied from $\alpha = 0.0$ (auto detection) to $\alpha = 0.9$, with the step of 0.1. The results are given in Table 5.1.

Table 5.1: Recognition rate [%] when the hidden layer dropout rate and learning rate are varied.

	learning rate									
dropout rate	**α=0.0**	**α=0.1**	**α=0.2**	**α=0.3**	**α=0.4**	**α=0.5**	**α=0.6**	**α=0.7**	**α=0.8**	**α=0.9**
0.0	67.90	65.81	68.60	68.60	68.60	67.67	70	67.67	68.83	65.58
0.1	71.16	63.72	67.44	70.46	72.09	71.86	69.30	71.39	73.02	71.86
0.2	71.86	6.51	66.97	69.76	71.62	71.39	70.46	71.62	70.46	70.46
0.3	70.93	5.81	65.81	68.60	70.23	70.46	71.86	70	69.53	72.32
0.4	71.16	3.72	64.65	67.67	69.30	70.93	70.69	70.93	72.09	71.62
0.5	70.46	3.25	8.13	63.72	66.51	67.90	68.83	69.30	71.62	70.23
0.6	64.18	2.79	4.88	7.67	12.09	64.65	66.27	68.60	67.20	67.20
0.7	62.79	3.02	5.11	7.20	6.27	9.53	10.46	24.41	58.13	64.41
0.8	63.25	3.48	4.65	4.65	4.65	6.04	8.60	8.83	11.39	11.16
0.9	55.34	2.79	3.02	2.79	4.18	3.95	3.48	5.11	7.20	6.74

Table 5.2: Recognition rate [%] when the input layers dropout rate and learning rate are varied.

	learning rate									
dropout rate	**α=0.0**	**α=0.1**	**α=0.2**	**α=0.3**	**α=0.4**	**α=0.5**	**α=0.6**	**α=0.7**	**α=0.8**	**α=0.9**
0.0	66.74	4.41	66.74	69.30	69.06	70.69	70.69	70	69.76	71.86
0.1	71.16	5.34	5.81	66.74	69.76	71.39	70.93	69.06	70	71.86
0.2	68.13	3.48	7.20	65.58	66.74	66.74	69.06	69.76	68.83	68.83
0.3	63.95	4.41	5.11	7.67	64.18	65.81	68.83	68.37	67.90	69.76
0.4	62.32	3.48	3.72	7.20	7.44	62.32	64.65	65.11	66.97	66.74
0.5	57.90	3.48	2.79	4.65	10.69	10.69	22.55	62.32	62.79	62.79
0.6	53.95	2.55	1.62	4.88	8.37	7.20	11.62	10.93	14.18	38.13
0.7	23.02	2.32	2.55	4.65	5.11	5.81	9.76	8.83	12.55	13.02
0.8	14.18	2.55	2.09	3.95	3.02	3.25	5.58	4.65	6.04	7.90
0.9	3.48	2.09	2.32	2.79	2.79	2.79	2.55	2.79	3.72	3.72

In the second phase, the recognition rate is observed when the input layers dropout rate and learning rate are varied. The dropout is again applied at all layers. In this phase of the experiment, the probability of the input layer dropout rate was varied from 0.0 (i.e. p=1) to 0.9, with the step of 0.1. The retaining probability for a hidden node was p=0.5. The learning rate was

varied in the same way as in the first phase of the experiment, from $\alpha = 0.0$ (auto detection) to $\alpha = 0.9$, with the step of 0.1. The results are given in Table 5.2.

According to the obtained results provided in Table 5.1, it may be observed that the best performance is achieved for the dropout rates 0.1, 0.2, 0.3 and 0.4 for all the considered learning rates except for $\alpha = 0.1$. On the other hand, according to the obtained results provided in Table 5.2, it may be observed that when the dropout is applied only to the input layer, it does not have significant influence on the recognition rate. However, it can be concluded that for the input layer dropout rate greater than 0.5, the recognition rate significantly decreases. If we focus on the influence of the learning rate on the performance of the method, it is evident that for some values of learning rate, the performance of the method is poor. This decrement in performance is especially evident when dropout is applied. This is because dropout introduces bigger amount of noise comparing with standard stochastic gradient descent. Therefore, stochastic gradient descent with applied dropout requires bigger learning rates than standard stochastic gradient descent. Depending on the combination of all three parameters, the performance can change significantly, from 2.09% to 73.02%.

5.4 Experiment 2

The second experiment (cf. Nasef and Marjanović-Jakovljević 2017b) observes the improvement of speaker identification in noisy environment. In order to find the optimal stochastic gradient descent parameters in the noisy environment, the white Gaussian noise was artificially added, and the signal-to-noise ratio (SNR) was set to 8dB, 12dB and 16dB, respectively (using MATLAB). Thus, four independent databases including the original database cleaned from noise were created. From these databases, we extracted 83 state-of-the-art features using signal processing techniques, as described in Chapter 3. For the classification, we trained the deep neural network with stochastic gradient descent implemented with the dropout regularization and rectified linear units. The training is conducted with 100 training examples in each mini-batch.

Different parameters, such as the input layer dropout rate, the learning rate and the hidden layers dropout rate, were analyzed for different values of signal-to-noise ratio. The proposed speaker recognition system architecture is shown in fig. 5.1.

Different curves, depicted in fig. 5.2, represent the recognition rate performance for different learning rates, changing values from 0.1 to 0.9 (with the step of 0.1) for the fixed value of the dropout rate. The “optimal parameters” curve presents the recognition rate with the best performance values, i.e. when the learning rate and the dropout rate are optimized for each signal-to-noise ratio (cf. Table 5.3). It is shown that the optimized performance, tuning both values for the dropout and learning rates, outperforms other performances when values are not optimized for each signal-to-noise value approximately in range from 5% to 7.5%.

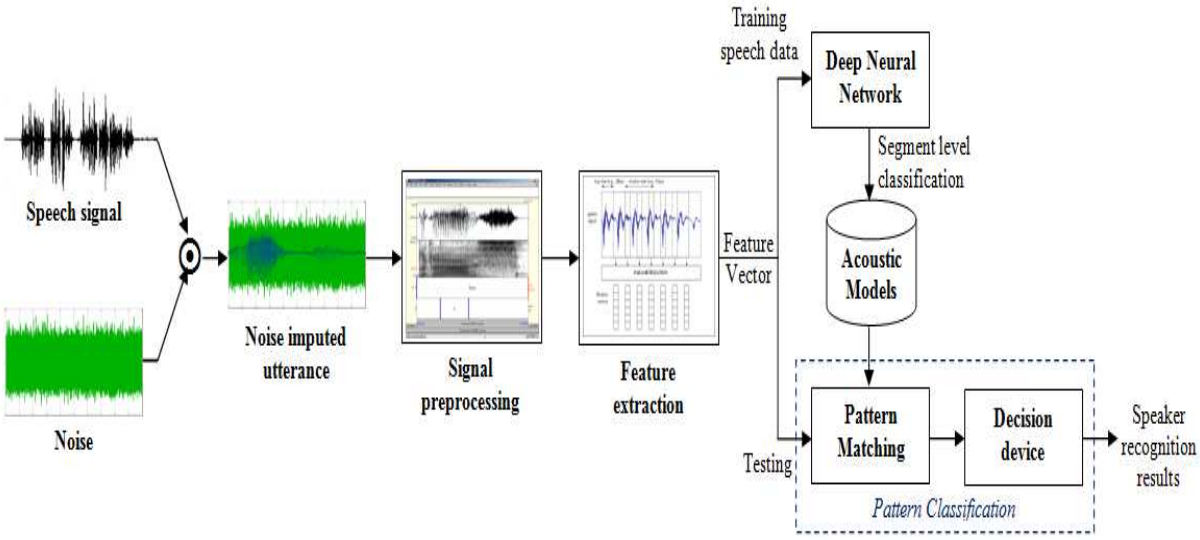

Figure 5.1: Automatic speaker recognition system architecture.

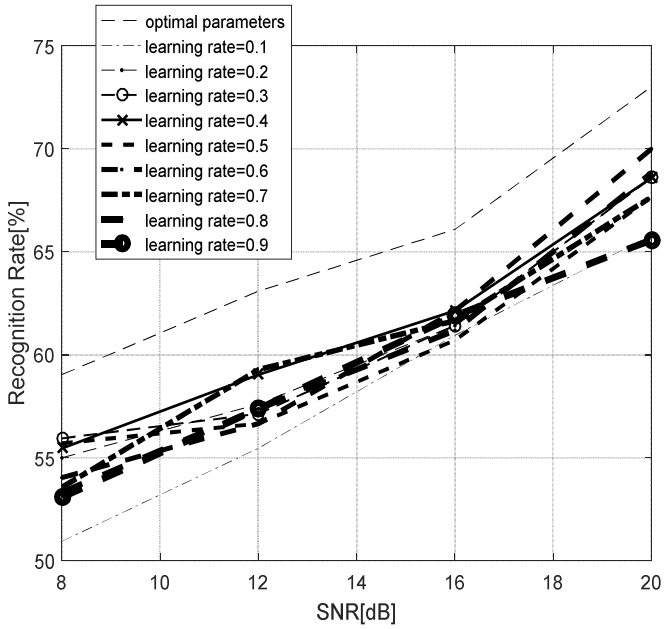

Figure 5.2: Speaker recognition rate performance for different signal-to noise ration levels and different learning rates.

Table 5.3: Optimal values of the learning and dropout rates for different signal-to-noise ratio levels.

SNR[dB]	8	12	16	*Cleaned signal*
Learning rate	0.1	0.0	0.7	0.8
Dropout rate	0.1	0.0	0.2	0.1

5.5 Conclusion

The experimental results reported in this chapter show that the dropout regularization may boost the performance of the stochastic gradient descent method in the task of automatic speaker recognition, even in a noisy environment. In this technique, sampling a thinned network by

dropping out units prevents co-adaptations of neurons and overfitting of hidden units. This is discussed in the following chapter.

Chapter 6
Conclusion

Methods based on deep learning neural network have been already demonstrated to provide better speaker recognition performance than other classifiers, but also to require considerable parameter tuning. This thesis aimed at showing that selecting appropriate value of parameters can significantly improve the performance of neural network methods on the task of automatic speaker recognition.

The contribution of this thesis is both theoretical and experimental. The thesis proposed an approach to automatic speaker recognition based on deep neural networks and the stochastic gradient descent algorithm. In addition, two experiments were designed and conducted in order to demonstrate that the optimization of the parameters of the stochastic gradient descent algorithm can improve automatic speaker recognition performance under no presence of noise and under noisy conditions, respectively. Three parameters of the stochastic gradient descent algorithm were considered in these experiments: the learning rate, and the hidden and input layer dropout rates. They were systematically changed in order to pick a model with the best performance on the speech recognition dataset.

It has been shown how dropout optimization can significantly enhance the performance of stochastic gradient descent method in automatic speaker recognition even in a noisy environment. This can be explained by the facts that the dropout technique represents an approximation to training of exponentially many neural networks that is inexpensive in terms of

cost and memory, and that it results in robust features. Sub-networks contained in a neural network are not independent because they share parameters. This implies that it is not necessary to train all possible sub-networks, and often it is not even feasible in acceptable time. Thus, the dropout regularization technique targets only a small portion of all possible sub-networks, and due to the parameter sharing, the other sub-networks are indirectly trained as well. In other words, dropout allows for an inexpensive training of neural networks (Goodfellow et al. 2016, pp. 258-259). Even more important, the training of a particular neuron does not rely on its connection with a particular set of other neurons. In a general case, a given neuron will be related to different sets of neurons across different sub-networks. Therefore, the dropout regularization technique prevents complex co-adaptations of neurons and results in more robust features (Krizhevsky et al. 2012, p. 6; Hinton et al 2012b; Nielsen 2015, cf. third chapter).

It has been also shown that picking a learning rate can also be very important task. As discussed in Chapter 4, the learning rate should be small enough, in order that the algorithm could work properly, but not too small because it could significantly slow down the algorithm's performance. However, the reported experimental results have also showed that for some values of the learning rate, the performance of the method is very poor, and that the decrement in performance is bigger when dropout is applied. This effect can be explained by the fact that dropout introduces bigger amount of noise comparing with the standard stochastic gradient descent algorithm. Therefore, stochastic gradient descent with applied dropout requires an appropriately adjusted learning rate.

Finally, a general conclusion may be drawn that the reported experimental results demonstrated the appropriateness of the proposed approach to automatic speech recognition under realistic conditions.

It is reasonable to expect that other parameters (such as the number of layers, the number of hidden units in layers, etc.) also affect automatic speech recognition performance. An analysis of their influence on the stochastic gradient descent algorithm performance is a rather challenging

task since it includes a specific trade-off. Namely, if a neural network model overfits, it may require a reduction of the number of hidden layers or the number of units in hidden layers. On the other hand, if the stochastic gradient descent method provides bad performance for a training set, it may require an increase of the number of hidden layers or the number of units in hidden layers. These research questions will be addressed as part of future work.

Index

References

Alam, J., Gupta, V., Kenny, P., Dumouchel, P., Speech recognition in reverberant and noisy environments employing multiple feature extractors and i-vector speaker adaptation, EURASIP Journal on Advances in Signal Processing, Vol.50, pp. 1-13 (2015).

Anusuya, M.A., Katti, S.K. Speech Recognition by Machine: A Review, International Journal of Computer Science and Information Security, 6 (3), pp. 181-205 (2009).

Beigi, H., Fundamentals of Speaker Recognition, first ed., Springer US, Springer Science+Business Media, LLC (2011).

Bishop, C. M., Pattern recognition and machine learning, Springer-Verlag New York Inc. (2006).

Boersma, P., Weenink, D. Praat software, University of Amsterdam, Retrieved from http://www.fon.hum.uva.nl/praat/ (Accessed April 22, 2016).

Bottou, L. Large-scale machine learning with stochastic gradient descent. In International Conference on Computational Statistics, 177-186 (2010).

Campbell, J.P. Speaker Recognition: a tutorial, In Proc. of the IEEE, 85 (9), 1437-1462, (1997).

Chen, C. H. Signal processing handbook, Dekker, New York (1988).

Cui X., Alwan, A., Noise robust speech recognition using feature compensation based on polynomial regression of utterance SNR, in IEEE Transactions on Speech and Audio Processing, vol. 13, no. 6, pp. 1161-1172 (2005).

Dahl, G.E., Sainath, T.N., Hinton, G.E. Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout, In IEEE International Conference on Acoustics, Speech, and Signal Processing, Vancouver, (2013).

Davis, S. Mermelstein, P. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. In IEEE Transactions on Acoustics, Speech, and Signal Processing, 28 (4), pp. 357-366 (1980).

De-la-Calle-Silos, F., Gallardo-Antolin, A., Pelaez-Moreno, C., Deep Maxout Networks Applied to Noise-Robust Speech Recognition, Proceedings of the Second International Conference on Advances in Speech and Language Technologies for Iberian Languages - IberSPEECH 2014, Vol. 8854, Springer-Verlag New York, NY, USA, pp. 109-118 (2014).

Desai, D. and Joshi, M. Speaker Recognition using MFCC and Hybrid Model of VQ and GMM, Proc. of the Second International Symposium on Intelligent Informatics (ISI'13), Mysore, India, (2014).

Friedland, G., Vinyals, O., Huang, Yan., and Muller, C. Prosodic and other long-term features for speaker diarization, IEEE Transactions on Audio, Speech, and Language Processing, 17 (5), pp. 985-993 (2009).

Gold, B., Morgan, N., Ellis, D. Speech and Audio Signal Processing: Processing and Perception of Speech and Music. Second Edition. Wiley, Englend, (2011).

Goodfellow, I., Bengio, Y., Courville, A. Deep Learning, MIT Press, http://www.deeplearningbook.org (2016).

Hernando, J., Nadeu, C. CDHMM speaker recognition by means of frequency filtering of filter-bank energies, Proc. Eurospeech, 5, pp. 2363-2366 (1997).

Hinton G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.-r., Jaitly, N, Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B., Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, IEEE Signal Processing Magazine, Volume: 29, Issue: 6, pp. 82-97 (2012).

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. Deep Neual Networks for Acoustic Modeling in Speech Recognition", IEEE Signal Processing Magazine, 29 (6), pp. 82-97 (2012).

Hinton, G.E., Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science*, **313**, 504-507, (2006).

Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., Improving neural networks by preventing co-adaptation of feature detectors, ArXiv, https://arxiv.org/pdf/1207.0580.pdf (2012b)

Holube, I., Fredelake, S., Vlaming, M. and Kollmeier, B. Development and analysis of an international speech test signal (ISTS). International Journal of Audiology, 49 (12), pp. 891-903 (2010).

Jankowski, C., Kalyanswamy, A., Basson, S. and Spitz, J. NTIMIT: A Phonetically Balanced, Continuous Speech Telephone Bandwidth Speech Database, Proc. of International Conf. Acoustics, Speech and Signal Processing, Albuquerque, 109-112, (1990).

Joho, D., Bennewitz, M., Behnke, S. Pitch estimation using models of voiced speech on three levels, In Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, Honolulu, Hawaii, pp. 1077-1080, (2007).

Jurafsky, D., Martin, J.H. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 2nd edition. Prentice-Hall (2009).

Kacur, J., Rozinaj, G., Herrera-Garcia, S., Speech Signal Detection In A Noisy Environment Using Neural Networks And Cepstral Matrices, Journal of Electrical Engineering, Vol 55, 05-06, pp 131-137 (2004).

Kim, S., Raj, B., Lane, I., Environmental Noise Embeddings For Robust Speech Recognition, Computing Research Repository (CoRR) - arXiv:1601.02553 (2016).

Krizhevsky, A., Sutskever, I., Hinton, G., ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems 25 (2012).

Kumar, A., Florencio, D., Speech Enhancement in Multiple-Noise Conditions using Deep Neural Networks, Cornel University, Available at: https://arxiv.org/abs/1605.02427v1 (2016).

Le, Q.V. A Tutorial on Deep Learning, Lecture Notes, Retrieved from https://cs.stanford.edu/~quocle/tutorial2.pdf, (Accessed April 25, 2015).

Le, Q.V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B. Ng, A.Y. On Optimization Methods for Deep Learning. *In Proc. of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, (2011).

Li, J., Deng, L., Gong Y., Haeb-Umbach, R., An Overview of Noise-Robust Automatic Speech Recognition, in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 4, pp. 745-777 (2014).

Maas, A.L., Hannun, A.Y., Ng, A.Y., Rectifier Nonlinerities Improve Neural Network Acoustic Models, In 30th International Conference on Machine Learning (ICML 2013) - Workshop on

Deep Learning for Audio, Speech and Language Processing. Atlanta, USA, June 16-21, 2013. Vol. 30, Iss. 6 (2013).

Maesa, A., Garzia, F., Scarpiniti, M., Cusani, R. Text Independent Automatic Speaker Recognition System Using Mel-Frequency Cepstrum Coefficient and Gaussian Mixture Models, InternationalJournal of Information Security, 3 (4), pp. 335-340 (2012).

McLaren, M., Lei, Y., Ferrer, L. Advances in Deep Neural Network Approaches to Speaker Recognition, *In Proc. 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP*), Brisbane, Australia, (2015).

Misamadi, S., Hansen, J.H.L, A Study on Deep Neural Network Acoustic Model Adaptation for Robust Far-field Speech Recognition, In Proc. of INTERSPEECH (2015).

Mitra, V., Wang, W., Franco, H., Lei, Y., Bartels, C., Graciarena, M., Evaluating robust features on Deep Neural Networks for speech recognition in noisy and channel mismatched conditions. INTERSPEECH 2014, Singapore, 14-18th September, 2014. pp. 895-899 (2014).

Mølgaard L.L., Jørgensen W.K., Speaker Recognition – Special Course, IMM-DTU, December 14, (2005),
http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/4414/pdf/imm4414.pdf

Nasef, A., Marjanović-Jakovljević, M., Njeguš, A. Stochastic gradient descent analysis for the evaluation of a speaker recognition. In: Analog Integrated Circuits and Signal Processing, 90(2), pp. 389–397 (2017a).

Nasef, A., Marjanović-Jakovljević M., Optimization of the speaker recognition in noisy environments using a stochastic gradient descent, In Proceedings of the fourth International Conference Sinteza 2017, Singidunum University, Belgrade (2017b).

Nielsen, M.A. Neural Networks and Deep Learning, Determination Press (2015).

Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H.G., Ogata, T., Audio-visual speech recognition using deep learning, Appl Intell 42, Springer, pp. 722-737 (2015).

Rabiner, L.R., A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE, Volume: 77, Issue: 2, pp. 257-286 (1989).

Rabiner, L. and Juang, B.-H., Fundamentals of speech recognition, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

Rabiner, L.R., Schafer R.W., Digital Processing of Speech Signals, Prentice-Hall (1978).

Rashmi C.R. Review of Algorithms and Applications in Speech Recognition System, International Journal of Computer science and Information Technologies, 5 (4), 5258-5262, (2014).

Richardson, F., Reynolds, D., Dehak, N. Deep Neural Network Approaches to Speaker and Language Recognition, *IEEE Signal Processing Letters*, **22** (10), 1671-1675, (2015).

Robinds, H. and Monro, S. A stochastic approximation method, Annals of Mathematical Statistics, 22 (3), 400–407, (1951).

Robinson, T., Hochberg, M., Renals, S. The use of recurrent neural network in continuous speech recognition, In *Automatic Speech and Speaker Recognition*, Lee, C.H., Soong, F.K, Paliwal, K.K., Eds, Kluwer Academic Publishers, Boston, MA, 233-258 (1996).

Sanderson, C. The VidTIMIT Database. IDIAP Communication 02-06, Dalle Molle Institute for Perceptual Artificial Intelligence, Martigny, Switzerland (2002).

Sanderson, C., Paliwal, K.K., Polynomial Features for Robust Face Authentication, Proc. International Conf. Image Processing, Rochester, New York (2002).

Santos, R. M., Matos, L. N., Macedo, H. T., Montalvão, J., Speech Recognition in Noisy Environments with Convolutional Neural Networks, 2015 Brazilian Conference on Intelligent Systems (BRACIS), Natal, pp. 175-179 (2015).

Seltzer, M.L., Yu, D., Wang, Y., An Investigation Of Deep Neural Networks For Noise Robust Speech Recognition, ICASSP 2013, IEEE, pp. 7398-7402 (2013).

Senior, A., Heigold, G., Ranzato, M.A., Yang, K. An empirical study of learning rates in deep neural networks for speech recognition. *In IEEE International Conference on Acoustics, Speech, and Singal Processing*, 6724-6728, (2013).

Shiota, S, Villavicencio, F, Yamagishi, J, Ono, N, Echizen, I, Matsui, T, Voice Liveness Detection for Speaker Verification based on a Tandem. In Odyssey 2016: The Speaker and Language Recognition Workshop. International Speech Communication Association, pp. 259-263 (2016).

Shriberg, E., Ferrer, L., Kajarekar, S., Venkataraman, A., Stolcke, A. Modeling Prosodic Feature Sequences for Speaker Recognition, Speech Communication (Special Issue on Quantitative Prosody Modelling For Natural speech Description and Generation, 46 (3-4), pp. 455-472 (2005).

Srinivas, V., Santhi rani, Ch. and Madhu, T. Neural Network based Classification for Speaker Identification. *International Journal of Signal Processing, Image Processing and Pattern Recognition,* **7** (1), 109-120, (2014).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research, 15 (1), 1929-1958, (2014).

Srivastava, R.K., Masci, J., Gomez, F., Schmidhube, J., Understanding Locally Competitive Networks, Proceedings of 3rd International Conference on Learning Representations (ICLR 2015), May 7-9, 2015, San Diego, CA (2015).

Sturim, D.E., Campbell, W.M. and Reynolds, D.A. Classification Methods for Speaker Recognition. In Speaker Classification I: Fundamentals, Features, and Methods. Muller, C. (Ed.), Springer Berlin Heidelberg, 4343, 278-297, (2007).

Sutskever, I., Martens, J., Dahl, G., Hinton, G. On the importance of initialization and momentum in deep learning. In *Proc. of the 30th International Conference on Machine Learning*, 1139-1147, (2013).

Vaseghi, S.V. Multimedia Signal Processing: Theory and Applications in Speech, Music and Communications. Wiley, England, (2007).

Vasilakakis, V., Cumani, S., and Laface, P. Speaker recognition by means of Deep Belief Networks, *In Proc. of the Biometric Technologies in Forensic Science*, Nijmegen, The Netherlands, (2013).

Wildermoth, R.B., Text-Independent Speaker Recognition Using Source Based Feature, Doctoral Thesis, Griffith University, Brisbane, Australia (2001).