Arild Brandrud Næss

Nearest Neighbor Frame Classification for Articulatory Speech Recognition

Thesis for the degree of Philosophiae Doctor

Trondheim, January 2015

Norwegian University of Science and Technology
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Electronics and Telecommunications

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Electronics and Telecommunications

ISBN 978-82-326-0718-1 (printed ver.)
ISBN 978-82-326-0719-8 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2015:24

Printed by NTNU-trykk

Abstract

The paradigm of phone-based hidden Markov models has dominated automatic speech recognition since the early 1980s, and continuous improvements of this approach combined with the exponential increase in computational power have led to impressive improvements in the performance of such systems in the past 30 years. Of late, however, these gains have seemed to level off, and there is a growing interest in exploring alternative paradigms. This thesis concerns itself with two of these newer approaches: articulatory speech recognition and exemplar-based methods.

Articulatory speech recognition considers speech not as a sequence of phones, but as an interplay between our articulators—the lips, the tongue, the glottis and the velum. This explicit modeling of the pronunciation process in the statistical framework of the speech recognizer allows for a better model of the pronunciation variation that occurs, particularly in spontaneous speech. Exemplar-based methods is a common name for all ways of using the training data directly rather than fitting a global statistical model to it. Most of these methods are based on finding nearest neighbors among the observation vectors.

The main focus of this thesis is on the frame classification of articulatory features by nearest neighbors, and on using this classification to produce input feature vectors for two transcription systems. We consider nearest neighbor-based frame-level classification of a multi-valued set of articulatory features (AFs) inspired by the vocal tract variables of articulatory phonology. This entails that, for each frame of the audio signal, we try to determine the value of each of our eight AFs at the corresponding point in time. Partly for comparison purposes, we do a frame classification of phones in the same way. We explore a variety of linear and nonlinear transformations of the observation vectors, and use the k nearest neighbors in the resulting vector space to do the classification. Our best results compare favorably to a multilayer perceptron (MLP) baseline.

Based on our k-nearest neigbhor (k-NN) frame classification, we make posterior-like feature vectors, which we incorporate into two systems for automatic transcription. The first of these is a conditional random field (CRF) for forced transcription of our set of AFs. The performance of our k-NN-based features in the CRF system is better than that of MLP-based features for most of the AFs, and on par with it for the rest of them. The second transcription system is a standard tandem hidden Markov model for phone recognition, where the k-NN-based features do not do as well as the MLP-based ones. Nevertheless, we argue that the flexibility and transparency of k-NN classification make it a very promising approach for articulatory speech recognition.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree of *philosophiae doctor* (Ph.D.) at the Norwegian University of Science and Technology (NTNU). My advisor has been Professor Torbjørn Svendsen at the Department of Electronics and Telecommunications.

In addition to research activities, the work included compulsory course studies corresponding to one semester of full-time studies, and one year of teaching assistant duties. This work was conducted in the period from August 2008 to October 2014. Most of the period from July 2010 until June 2012 was spent as a visiting student at Toyota Technological Institute at Chicago working under the supervision of my co-advisor, Professor Karen Livescu.

The work was funded by the Norwegian Research Council as part of the project *Spoken Information Retrieval by Knowledge Utilization in Statistical Speech Processing* (SIRKUS). The one-year period of teaching assistant duties was funded by the Department of Electronics and Telecommunications at NTNU.

Acknowledgements

Thanks to my advisor, Professor Torbjørn Svendsen, for convincing me to do my PhD in the fascinating field of speech technology. His long experience in the field has been valuable. I also want to thank all my colleagues at NTNU—especially Timo Mertens, Trond Skogstad and Line Adde—for their companionship.

I would like to extend a particular thanks to my co-advisor, Professor Karen Livescu, who has been my main mentor and collaborator in the research presented herein. During the two years I spent as a visiting student at Toyota Technological Institute at Chicago (TTIC), she gave me more guidance and support than any PhD student could reasonably hope for, and I have benefited greatly from our discussions in the following years as well. Her dedication and insight continue to amaze me. Thanks to everybody at TTIC, for contributing to such a stimulating research environment. I had the pleasure of getting to know too many great people among the students, interns and faculty at TTIC and the University of Chicago to mention but a few of them: Avleen Bijral, Payman Yadollahpour, Hao Tang, Raman Arora, Martin Hjelm, and Preethi Jyothi—thank you all for the discussions, the inspiration and the fun. A special thanks to Rohit Prabhavalkar, for all the good times we had both on and off work—I wish there could have been more of them.

Thanks to my father for instilling in me a sense of academic ambition and an appreciation for mathematical rigor. Thanks to my late mother for always reminding me that there are other things in life. Thanks to my brother for growing up with me, and for being who he is.

Last, but in no way least, my gratitude goes to Julie— for her invaluable support and for the countless sacrifices, large and small, she has made to enable me to finish this thesis. Chance would have it that we met just as I was entering my years as a PhD student, and the love I have for you has only grown since. Without you my life would have been infinitely poorer.

Contents

List of Figures

List of Tables

List of Abbreviations

AF	articulatory feature
ASR	automatic speech recognition
CCA	canonical correlation analysis
CRF	conditional random field
DBN	dynamic Bayesian network
G	glottis/velum (AF)
GLOT	glottis opening degree (AF)
GMM	Gaussian mixture model
HMM	hidden Markov model
IPA	International Phonetic Alphabet
k-NN	k-nearest neighbors
L	lips (AF)
LDA	linear discriminant analysis
LIP-LOC	lip constriction location (AF)
LIP-OPEN	lip opening degree (AF)
LPP	locality preserving projections
MAPSSWE	matched pairs sentence-segment word error
MFCC	mel-frequency cepstral coefficient
MLP	multilayer perceptron
MLPP	multilayer perceptron posterior
MT-LDA	multi-task linear discriminant analysis
PCA	principal component analysis
PLP	perceptual linear predictive coefficient
STP	Switchboard Transcription Project
T	tongue (AF)
TB-LOC	tongue body constriction location (AF)
TB-OPEN	tongue body opening degree (AF)
TT-LOC	tongue tip constriction location (AF)
TT-OPEN	tongue tip opening degree (AF)
VEL	velum opening degree (AF)

Note Abbreviations that are only used within the subsection in which they are defined have been omitted from this list.

Chapter 1

Introduction

What is it that all utterances of a word have in common, for all speakers in all acoustic conditions in which we can hear them, that makes us recognize the word? This might be said to be the fundamental question underlying all attempts at automatic speech recognition (ASR), and it has proven to be a very hard question to answer.

The most common approach is to consider speech as a sequence of *phones*—the fundamental sounds of a given language.[1] But, as anybody who has looked at a phonetic transcription of spontaneous speech is very aware of, words are seldom pronounced in the exact way that their dictionary baseforms say they should be. This problem of pronunciation variation has been dealt with in various ways within the framework of phonetic ASR. Expanding the dictionaries to include common pronunciation variants of each word is one approach, trying to hand-craft or learn substitution rules for the phones is another.

Articulatory ASR arguably provides a more principled way of dealing with varying pronunciations of a word. In this framework, speech is considered as an interplay between the main articulators in our vocal tracts—i.e., the lips, the tongue, the glottis and the velum. Instead of taking the canonical pronunciation of a word as our starting point, we now consider the utterance of a word as a series of targets that the articulators try to reach in a specific order. Pronunciation variation occurs when the articulators fail to reach their targets (e.g., the tongue does not move as far forward as it should) or get out of sync with one another (e.g., the velum closes before

[1]A *phoneme* is the smallest structural unit that distinguishes meaning in a language; a *phone* is the physical realizations of a phoneme in an utterance. Thus, a phoneme can be considered as an abstraction of a set of phones which are perceived as equivalent to each other in a given language.

it should). As we shall see, the articulatory framework can give elegant explanations of many pronunciation variations that are hard to account for within the phonetic framework.

ASR systems are usually based on building a global statistical model for the training data and using this to classify or recognize the test data. This has the advantage of being easily generalizable. The downside is that the training times can be very long. In this thesis, we have chosen an *exemplar-based* approach. Instead of building a global model of the training data, we select a subset of exemplars from the training data to build a local model specifically for every test sample. This is more computationally intensive at test time, but since it more or less eliminates training time, it carries great promise for experimental systems.

1.1 Motivation

The focus of this thesis is on the classification of articulatory features (AFs) from audio by nearest neighbor methods; in this section we present our reasons for choosing this focus. We will first explain our interest in AFs, before moving on to the arguments in favor of using nearest neighbor methods.

The motivation for classifying AFs from audio is twofold: On the one hand it is of interest in its own right as a scientific question with certain direct applications, on the other hand it is an essential step in any end-to-end speech recognition system that uses an articulatory approach. We will begin by considering the arguments for choosing such an articulatory approach rather than the standard phonetic one.

Articulatory Speech Recognition The performance gains for ASR system over the past 20 years have been immense [Saon and Chien, 2012; Heigold et al., 2012]. It has been claimed, however, that the gains have been leveling off in recent years. Lee and Siniscalchi [2013] refer to this as the "S-Curve" of ASR technology progress, where decades of fast advances is followed by a period of slower progress as the possibilities of the current paradigm are gradually exhausted and before a new paradigm is found. At the core of the current paradigm, which we give a brief account of in Section 2.1, is the modeling of speech as a sequence of phones with a hidden Markov model (HMM) in combination with Gaussian Mixture Models (GMMs). We shall refer to this as the phonetic GMM-HMM framework.

In the phonetic GMM-HMM framework, speech is considered as a sequence of distinct phones, somewhat like beads on a string. In linguistics, it is phonological features and not phones that are viewed as the fundamental units of speech [Ostendorf, 1999; Halle, 1992], and various theories of *non-linear phonology* [McCarthy, 2001], where these features are not confined to phone segments, have been gaining prominence. One of these theories is that of Browman and Goldstein [1986; 1989; 1992] where speech is described in terms of the movement of the articulators. This is parameterized as vocal tract variables (closely related to AFs), the values of which determine what sound is produced at a given time. Thus, each word is represented as an interplay between the articulators—more like threads weaved together on a loom than like beads on a string.

One of the problematic aspects of the phonetic GMM-HMM framework is how to model pronunciation variation. Words are seldom pronounced according to their canonical phonetic transcription, particularly in spontaneous speech. Livescu [2005] considers a data set of recorded and manually phonetically transcribed American English conversations consisting of approximately 10 000 spoken word tokens. In this data the word *don't* occurs 89 times. In only four of these cases is it pronounced in its canonical form, which in the Arpabet phonetic alphabet[2] is `[d ow n t]`. Some of the recorded pronunciation variants were `[d ow n]`, `[d ow_n t]` and `[d ah]` (occurring 37, 3 and 1 times, respectively).

Let us consider the pronunciation variant `[d ow_n t]`, where the diacritic `[_n]` denotes nasalization. Compared to the canonical pronunciation, this variant involves the deletion of the phone `[n]` along with nasalization of the diphthong `[ow]`. To model this in the phonetic GMM-HMM framework, one could either have a hand-crafted rule that says that an `[n]` which occurs between an `[ow]` and a `[t]` has a certain probability of being deleted, or one could try to learn this and similar probabilities from a data set by counting the number of times this deletion is observed. The first option is rather labor-intensive if the set of rules is to cover most pronunciation variation; the second option requires a lot of data to give good results. One could also add `[d ow_n t]` and all other observed pronunciation variants to the dictionary as acceptable pronunciations of *don't*. In the data set mentioned above, however, there are 20 pronunciation variants of the word *don't* alone—and since half of these occur only once, one can expect there to be even more in a bigger data set.

The articulatory approach provides a more elegant framework for explaining this and related pronunciation variants. Figure 1.1 shows how the `[n]` deletion in *don't* can be explained merely by asynchrony between voicing and nasality on the one hand and the tongue features on the other. We

[2] An overview of the vowel and consonant symbols in the Arpabet phonetic alphabet is included in Tables A.1 and A.2 in the appendix.

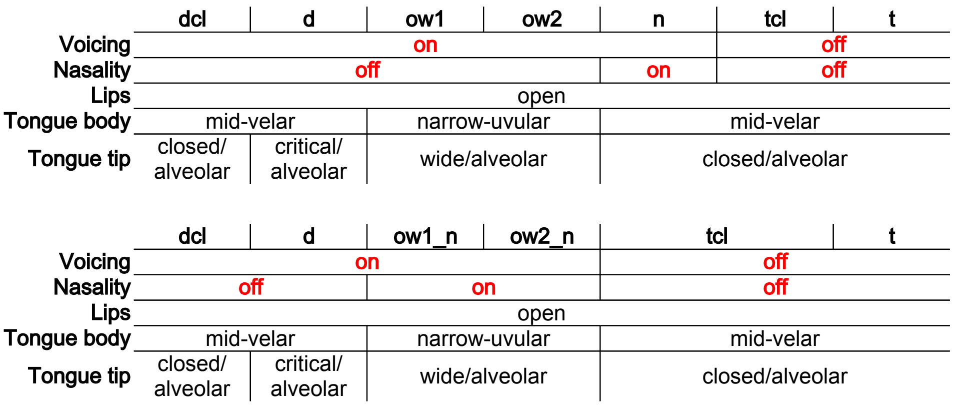


Figure 1.1: The canonical pronunciation of *don't*, [d ow n t], and the pronunciation variant [d ow_n t]. The asynchrony causing the pronunciation variation occurs for the articulatory feature values shown in red. The diphthong [ow] is split in two parts with different articulatory targets, and the diacritic [_n] denotes that the diphthong is nasalized in this pronunciation variant.

see that the tongue features are supposed to change values slightly before the voicing and nasality features turn off. If the voicing and nasality turns off at the same time as the tongue features change value, the result is that [ow] is nasalized and [n] is deleted.

Other frequent pronunciation variations that can be explained merely by asynchrony between these particular AFs include [p] and [t] insertion (e.g., *sense* → [s eh n t s] and *warmth* → [w ao r m p th]) as well as nasalization of other vowels than [ow]. If one, in addition to asynchrony between AFs, allows for *substitution* of AF values, even more pronunciation variation can be explained.

Aside from the modeling of pronunciation variation, an articulatory approach can be said to be beneficial for low-resource languages [Stüker et al., 2003a,b; Qian et al., 2011]. The articulatory framework for ASR is not language independent, but the dependence is far weaker than is the case for the phonetic framework. This makes it more feasible to use existing articulatory models for a low-resource language for which new models are hard to build. An articulatory approach to ASR has also been shown to be potentially beneficial for noise robustness [Kirchhoff et al., 2002; Kirchhoff, 1999].

Other Uses for Articulatory Features The classification of articulatory features can also be valuable in itself, independently of an ASR system.

If a set of AFs has a clear correspondence to the configuration of the major articulators in the vocal tract, a system that can reliably classify these AFs would be useful in a number of ways. Being able to obtain an articulatory representation of speech from the audio signal alone is far more convenient than other methods such as electropalatography (EPG) [Hardcastle, 1972; Wrench and Richmond, 2000], electromagnetic articulography (EMA) [Wrench, 2000] or magnetic resonance imaging (MRI) [Narayanan et al., 2011], all of which require complex apparatus. This could open the way to applications in several fields. One can envision uses in branches of speech therapy and computer-assisted pronunciation training (CAPT). CAPT systems are increasingly incorporated in language courses [Stenson et al., 1992; Neri et al., 2002], and being able to obtain a representation of a language learner's articulators when she tries to speak a sentence would enable a CAPT system to give the language learner very accurate feedback to improve her pronunciation.

Nearest Neighbor Classification We have seen how articulatory approaches pose an alternative to the GMM-HMM framework's view of speech as a sequence of phones. Another aspect of the GMM-HMM framework is how a global statistical model for each of the phones is built from the training data, usually with Gaussian mixture models. Building these models require extensive tuning of parameters and the training times can be very long.

Exemplar-based techniques have been subject of renewed interest recently [Sainath et al., 2012]. Instead of building a global model of the training data, these techniques select a subset of exemplars from the training data to build a local model specifically for every test sample. This is more computationally intensive at test time, but it more or less eliminates the long training times of the GMM-HMM framework and similar approaches.

The k-nearest neighbors (k-NN) classifier is fundamental to most exemplar-based techniques. k-NN classification involves no training; the training data is simply stored, and each test point is classified by finding the nearest neighbors to it among the training points and choosing the class that the majority of these belong to. While nearest neighbor-based methods are not often used directly in speech recognition, there has been some recent effort in this direction. (e.g., Deselaers et al. [2007]; Golipour and O'Shaughnessy [2010]; Labiak and Livescu [2011]).

k-NN classifiers have some advantages that make them attractive for speech recognition: they are inherently discriminative, they avoid the long training times associated with typical Gaussian mixture models on large data sets, and they are nonparametric so they involve relatively little tuning (typically only the number of neighbors k). The nonparametric nature of k-NN classifiers is particularly attractive for new types of models, such as articulatory models, where there is less understanding of the distribution of the data or of the best strategy for training a parametric classifier.

Another advantage of k-NN techniques is that they allow you to do several classifications based on different labelings with a single neighbor search. This is especially relevant for articulatory ASR systems, where there will typically be a set of several AFs to classify. When this is done with general-purpose acoustic observation vectors, like MFCCs, the computational overhead to doing several classifications is minimal. With most other classifiers—e.g., a multilayer perceptron—we would have to train a separate classifier for each task. k-NN is arguably also a very "transparent" classifier, since its decisions are based solely on the nearest neighbors and these can easily be inspected.

Nearest neighbor techniques have one major disadvantage: at test time, they require a search through all of the training data. However, a number of approximate search techniques can greatly improve search speed (see e.g., Friedman et al. [1977]; Arya et al. [1998]; Andoni and Indyk [2006]).

1.2 Proposed Approach

This thesis proposes to do frame classification by nearest neighbor methods of a set of articulatory features, and to apply the results of this classification to define input features to two transcription systems.

The lexical access experiments of Livescu [2005] show how one, given an articulatory labeling of spontaneous speech, can use articulatory models to improve the word recognition rate compared to a standard phonetic system. In this thesis we will attempt to obtain the articulatory labeling from the speech signal, using the same articulatory feature set and data set. Simply put, while Livescu's concern was how to get a word transcription from an AF transcription, our concern here is how to get an AF transcription from the speech signal.

We will do the frame classification by k-nearest neighbor methods, and examine how the performance is impacted by standard transformations like principal component analysis and linear discriminant analysis, as well as by less used transformations, like canonical correlation analysis.

We report frame error rates for these experiments, but it is not straightforward to assess the quality of the classification based on these. One reason for this is that this is the first attempt to classify this particular set of AFs from audio, so there is no comparison baseline. Another reason is that our ground truth labels, being derived from a phonetic transcription, are not completely reliable, especially around phone boundaries, so correct classifier decisions may in some cases be reported as errors. A good test for such a frame classification system, though, is how well it performs as a front-end to a transcription system. The mentioned lexical access system [Livescu, 2005] would be a natural choice, but it was designed for use with ground truth AF labels, and is not easily amended to handle uncertain labels. We have therefore turned to two other systems for automatic transcription: one conditional random field for forced transcription of articulatory features, and one traditional tandem hidden Markov model for phone recognition that we augment with articulatory information. This is both an evaluation of our classifiers and an examination of how k-NN outputs can be employed by automatic transcription systems by way of feature functions.

1.3 Contributions

Following the approach outlined above, the main contributions of this thesis can be summarized in the following five main points.

Frame Classification of Our Articulatory Feature Set The work published in Næss et al. [2011] and presented in Chapter 3, represents the first attempt at classifying the set of articulatory features described in Section 3.1 from speech audio. This set of features is multi-valued, subject-independent and has a close correlation with measurable physical properties—which we argue are beneficial characteristics for a set of AFs—and it has been shown to give good results in lexical access experiments [Livescu, 2005]. We present thorough experiments with various transformations of the acoustic observation vectors.

Phonetic Frame Classification of the STP Corpus Using Nearest Neighbors The Switchboard Transcription Project (STP) data is a particularly challenging corpus for phonetic frame classification because of its large amount of pronunciation variation. Very few phonetic frame classification experiments have been done on this data set (we are aware only of Subramanya and Bilmes [2009]), and, to the best of our knowledge, the experiments we present in Chapter 3 are the first using nearest neighbor classifiers.

Novel Use of Canonical Correlation Analysis The use of Canonical Correlation Analysis (CCA) [Hotelling, 1936; Hardoon et al., 2004] as an unsupervised dimensionality reduction technique is a novel approach that recently has shown great promise [Chaudhuri et al., 2009; Livescu and Stoehr, 2009; Arora and Livescu, 2013]. In Section 3.6 we investigate how CCA can be applied as a supervised dimensionality reduction for our k-NN frame classifiers. We also present an adaptation of this technique, original to this work, which is particularly well-suited for multiple related classification tasks like ours.

Using MLP Posteriors for k-NN Classification In Section 3.7 we show how posterior class probabilities from a multilayer perceptron (MLP) classifier can be used for k-nearest neighbor classification. Although we have since become aware of one other work using a similar method [Asaei et al., 2010a], this is still a relatively novel technique which we believe benefits from this further inquiry. In our experiments this method outperforms the MLP classifier itself.

k-NN-Based Feature Functions for Transcription Systems In Chapter 4 we investigate how different feature functions can be used to apply the results of a k-nearest neighbor classification in two automatic transcription systems, one conditional random field (CRF) system for forced transcription of AFs and one tandem HMM system for phone recognition. We are aware of no other work that systematically examines different feature functions for k-NNs, and the feature functions we construct are our own inventions. Although there have been previous attempts at inputting k-NN features in a CRF, this is to the best of our knowledge the first attempt at building a k-NN-based tandem system.

1.4 Thesis Outline

The remainder of this thesis is organized as follows. In Chapter 2 we present an overview of the relevant background; we give brief accounts of the traditional phonetic, the exemplar-based and the articulatory approaches to ASR, before we go on to review the relevant tests for statistical significance. Chapter 3 presents our experiments with frame classification of both articulatory features and phones, using nearest neighbor classifiers augmented with various transformations of the observation vectors. Chapter 4 begins by showing how k-NN feature vectors can be generated based on these results, and goes on to consider how such vectors perform as observation vectors—first in a CRF for forced transcription (Section 4.2), then in a tandem HMM system for phone recognition (Section 4.3). In Chapter 5 we present our conclusions and ideas for future work.

Chapter 2

Background

In this chapter we will review the necessary background as well as previous work within the main areas of this thesis.

We begin with a presentation of the traditional, phonetic framework of automatic speech recognition in Section 2.1. We go on to present tandem HMM systems and the multilayer perceptron classifier that underlies it, in Section 2.2. In Section 2.3, we present non-parametric and exemplar-based models in speech recognition, with particular emphasis on k-nearest neighbor classifiers. We give a brief presentation of statistical graphical models, which underpin much of the work in articulatory speech recognition, in Section 2.4. Section 2.5 gives a presentation of some of these articulatory systems for ASR, after considering the problem of inferring the articulatory configuration of the vocal tract from the speech signal that is produced. Finally, in Section 2.6, we give a brief presentation of how the results in this field can be evaluated, with the help of statistical significance tests and confidence intervals.

2.1 Traditional Speech Recognition

We will here give a broad overview of standard phone-based speech recognition using hidden Markov models and Gaussian mixture modeling—what we refer to as the phonetic GMM-HMM framework. As mentioned, this has been the dominant paradigm within ASR for three decades or more, and as of today forms the basis of practically all commercially available speech recognizers. It thus constitutes the default procedure that the alternative approaches of this thesis, articulatory modeling and exemplar-based modeling, contrast with. The GMM-HMM framework also forms part of our tandem system for phone recognition presented in Section 4.3.

The fundamental problem of all speech recognition system is to infer the word sequence being uttered from the acoustic signal of the utterance. If we let $\mathbf{w}$ be the sequence of words uttered and $\mathbf{O} \in \mathbb{R}^{m \times T}$ be a sequence of T m-dimensional numeric vectors representing the acoustics of the utterance, we can express the speech recognition task as the equation

$$\begin{aligned} \hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{O}) & (2.1) \\ &= \arg\max_{\mathbf{w}} \frac{p(\mathbf{O}|\mathbf{w})P(\mathbf{w})}{p(\mathbf{O})} & (2.2) \\ &= \arg\max_{\mathbf{w}} \underbrace{p(\mathbf{O}|\mathbf{w})}_{\substack{\text{Acoustic}\\ \text{model}}} \underbrace{P(\mathbf{w})}_{\substack{\text{Language}\\ \text{model}}}, & (2.3) \end{aligned}$$

where $P(\mathbf{w})$ is the prior probability of the word sequence $\mathbf{w}$, and $p(\mathbf{O})$ is the probability density function over the acoustics evaluated at $\mathbf{O}$. Equation (2.1) simply states that we want to find the most probable word sequence given the acoustics. Equation (2.2) follows from Bayes' theorem. And because $\mathbf{O}$ is given, its probability is irrelevant to the maximization and we can disregard it, yielding (2.3), which is sometimes called The Fundamental Equation of Speech Recognition.

Equation (2.3) is a beneficial transformation because it allows us to split a hard problem into two separate parts. Estimating the likelihood $p(\mathbf{O}|\mathbf{w})$ and the prior probability $P(\mathbf{w})$ separately is an easier task than trying to estimate the posterior probability $P(\mathbf{w}|\mathbf{O})$ directly. $p(\mathbf{O}|\mathbf{w})$ is commonly referred to as *the acoustic model*, and $P(\mathbf{w})$ as *the language model.*[1]

Figure 2.1 shows a block diagram of the decoding process. We will now go on to look at each of its parts.

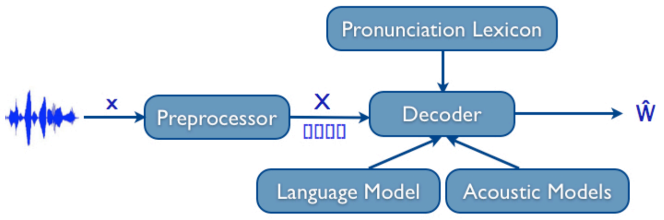


Figure 2.1: A block diagram of the decoding part of an ASR system.

[1]Strictly speaking, $p(\mathbf{O}|\mathbf{w})$ and $P(\mathbf{w})$ cannot be considered as *models* without introducing the hidden Markov model in the distributions—i.e., $p(\mathbf{O}|\mathbf{w}, \mathbf{\Phi})$ and $P(\mathbf{w}|, \mathbf{\Phi})$, where $\mathbf{\Phi}$ indicates the whole parameter set of the HMM. We omit this here for simplicity of notation.

The Preprocessor

The first step is to transform the waveform representing the acoustic signal as a sequence of numeric vectors. We want to retain as much as possible of the information pertinent to what speech sounds are being uttered, and discard as much as possible of the rest.

To extract one observation vector in this sequence, we typically consider a segment of the signal within a certain time window, usually of about 20 milliseconds. We assume the signal to be stationary within this segment (a questionable assumption in some cases) so that we can perform a spectral analysis on it—i.e., estimate the strength of the different frequency components. We then shift our time window one step and repeat the process. The shift is usually shorter than the time window, typically 10 ms, so that our segments overlap. We refer to these time segments of the speech signal as *frames.*

Two commonly used techniques for extracting the observation vector from a frame are mel-frequency cepstral coefficients (MFCCs) [Huang et al., 2001, Section 9.3] and perceptual linear predictive coefficients (PLPs) [Hermansky, 1990]. MFCCs are based on transforming the spectrum of the signal using the *mel scale*, a perceptual scale which models the frequency response of the human ear, and mapping it to the so-called *cepstrum* [Bogert et al., 1963; Oppenheim and Schafer, 2004], which can be thought of as the "spectrum of the spectrum". The cepstrum is useful because the fundamental frequency generated by the vocal cords and the formant filtering of the vocal tract are additive in this domain (they multiply in the frequency domain and convolve in the time domain).

Perceptual linear prediction is very similar to the computation of MFCCs, but has perceptual properties incorporated in a way that is more directly related to physical results [Holmes and Holmes, 2001, p. 165]. PLPs are based on the coefficients of a linear regression model for each sample given the previous L samples, where L is the order of the PLP model. For use in ASR, PLPs are typically mapped to cepstral coefficients, in the same way as MFCCs, but for PLPs the Bark scale rather than the mel scale is used to transform the frequencies. For both MFCC and PLP analysis, 12 coefficients are typically used along with the log energy, and first and second order time derivatives are added to capture dynamic effects, yielding one 39-dimensional observation vector per frame.

The Pronunciation Lexicon

The pronunciation lexicon is an expression of words as subword units. The phoneme is the subword unit most commonly used, but one can also have a pronunciation lexicon based on a different unit, such as syllables. What is important is that the subword unit is acoustically unambiguous. The letter "a" can represent many different sounds in different words, while the phoneme `/aa/` represents one particular sound.[2]

As mentioned, although most words have one canonical pronunciation, they are often pronounced in different ways. Thus, a pronunciation lexicon will often include several pronunciation variants for each word. E.g., the lexicon entry for the word *don't* could list three pronunciations: the canonical `/d ow n t/` along with the variants `/d ow n/` and `/d ow t/`. Sometimes these pronunciation variants are weighted with probabilities, but in the standard system we are focusing on here, they are not.

The Language Model

The role of the language model is to assign a prior probability to a word sequence, $\mathbf{w}$, independently of the acoustics. For some small-vocabulary tasks, the language model can be rule-based, allowing certain sequences of words and forbidding others, but in most cases it is probabilistic.

Most language models are estimated with n-grams. The central assumption of an n-gram language model is that the conditional probability of a word given its complete history is equal to the conditional probability of the word given its $n-1$ closest predecessors. Thus, for a sequence of words $\mathbf{w} = (w_1, w_2, ..., w_N)$, we can estimate the joint probability $P(\mathbf{w})$ by applying the chain rule, as

$$P(w_1, w_2, ..., w_N) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-n+1}, ..., w_{i-1}), \tag{2.4}$$

i.e., as the product of the probabilities of each of the words following its $n-1$ predecessors (or all of the predecessors if there are fewer than $n-1$ of them). For example, the trigram probability ($n = 3$) of the phrase "she

[2]We follow the convention of listing phonemes enclosed in slashes and phones enclosed in square brackets. Note that although the phoneme `/aa/` represents one particular sound, because of pronunciation variation it can be realized as several different sounds, including but not limited to its canonical pronunciation as the phone `[aa]`.

had your dark suit" would be

$$P(\text{“she”}) \cdot P(\text{“had”} \mid \text{“she”}) \cdot P(\text{“your”} \mid \text{“she had”}) \\ \cdot P(\text{“dark”} \mid \text{“had your”}) \cdot P(\text{“suit”} \mid \text{“your dark”}).$$

Each of these probabilities is typically estimated from training data by counting:

$$\hat{P}(w_i \mid w_{i-n+1}, ..., w_{i-1}) = \frac{C(w_{i-n+1}, ..., w_i)}{C(w_{i-n+1}, ..., w_{i-1})}, \tag{2.5}$$

where $C(\mathbf{w})$ is the number of occurrences of the word sequence $\mathbf{w}$ in a set of training data.[3]

$n = 3$ has been a standard value, but lately n-gram models have been used with n as high as 6. Especially for high values of n, the problem of zero counts arises. We see from Equation (2.4) that a single unseen n-gram would make the probability of the whole sequence zero—i.e., a naive language model would assign zero probability to a sentence if it contains any sequence of n words that does not occur in the training data. Various techniques, such as Good-Turing Smoothing [Gale and Sampson, 1995] and Kneser-Ney Smoothing [Kneser and Ney, 1995] are used to deal with this issue.

The Acoustic Model

Some would regard the acoustic model as the core of the speech recognizer. For each frame of the speech signal, it considers the acoustic vector and tries to determine which part of which phone (or other subword unit) has the highest probability of having generated it. Modeling the conditional probability density $p(\mathbf{O}|\mathbf{w})$ directly is infeasible for all but the smallest-vocabulary isolated-word recognition tasks, as there will be too many possible $\mathbf{O}, \mathbf{w}$ combinations. Therefore, the acoustic model is generally further decomposed into multiple factors, most commonly using hidden Markov models.

A *Markov process* is a stochastic process that has a memoryless property. The first-order Markov property implies that the probability distribution of future states of the process, conditioned on both past and present states, depends only upon the present state; i.e., given the present, the future does not depend on the past. When the random variables of a Markov

[3]Note that the training data for the language model is usually different from the training data for the acoustic model. Since the language model does not use any audio, it is customary to use large data sets of textual data to train its parameters.

process take only discrete values, as is the case in a finite-state machine, the process is called a *Markov chain*. This technique was developed by the Russian mathematician Andrey Markov, who published a pioneering paper in 1913 where he analyzed the letter sequence in one of Pushkin's novels [Markov, 1913; Hayes, 2013]. Markov chains have since been given countless applications.

In a *hidden Markov model* (HMM) the state sequence is unknown. For each time step the Markov chain transitions out of its current state and either enters a different state or loops back to the same state as before. Every time it enters a state, it emits an observation generated from a state-specific probability distribution. Based on these observations, we can try to determine the most likely sequence of states.

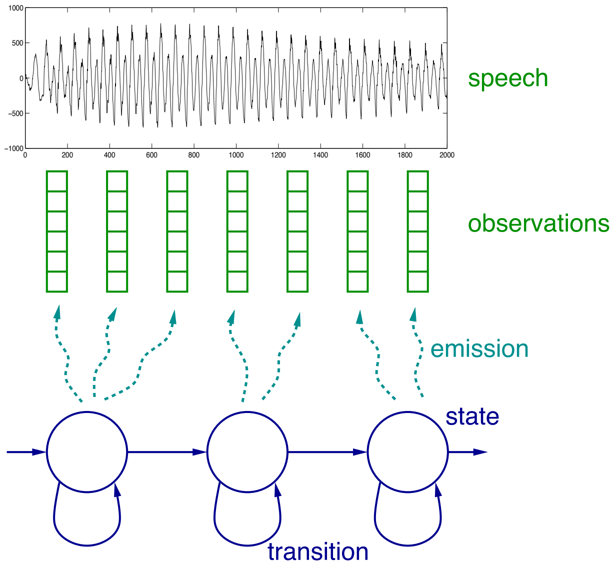


Figure 2.2: Hidden Markov model of speech. (Figure from Clark [2006].)

The use of HMMs in ASR dates back to the 1970s, and is generally attributed to the independent work of Baker [1975] and Jelinek [1975]. Although an HMM can be based on an ergodic Markov chain, where any state is reachable from any other state, the HMMs used in ASR almost invariably have a left-to-right topology, like the one shown in Figure 2.2, where you cannot return to a state after having transitioned to a new state. The HMMs used in ASR also tend to be first order, where the current state depends only on the previous state. Nothing precludes using a higher-order Markov chain (in an nth order Markov chain the current state depends on the n previous states), but this will entail substantially larger complexity and memory requirements. Some experiments have been done on second-order HMMs for ASR (see e.g., Watson and Tsoi [1992]; Mari et al. [1997]).

The following two conditional independence assumptions are made: In an nth order HMM (regardless of whether $n = 1$ or higher), the current state is assumed independent of all other states given the n previous states; and, the observation is assumed independent of all other observations and states given the current state. We also assume that the speech signal can be considered statistically stationary when it is in one particular state: the probability distribution governing how likely an observation vector is to be generated, is determined entirely by what state in the Markov chain we are in. Therefore each state must be chosen to represent a portion of the speech signal where the acoustics do not change too much. Since the phones represent our fundamental speech sounds (of which the phonemes are an abstraction), one possible choice is to let each phone have its own state. More commonly, though, each phone is represented by three states, to account for the fact that the beginning, middle and end of the utterance of a particular phone will tend to produce different acoustics.

The main components of an HMM are a set of states $\mathcal{S} = \{1, ..., S\}$; a set of probabilities of starting in any of those states, $\boldsymbol{\pi} = \{\pi_i\}$, $i \in \mathcal{S}$; a set of probabilities of moving between any pair of those states, $\mathbf{A} = \{a_{ij}\}$, $i, j \in \mathcal{S}$; and the mentioned state-specific emission probability distributions, $\mathbf{B} = \{\mathbf{b}_i\}$, $\mathbf{b}_i \in \mathbb{R}^m$, $i \in \mathcal{S}$, where m is the dimensionality of the emitted observation vectors.

For a few restricted applications, like command and control tasks with a restricted set of commands, it is feasible to represent each allowable phrase with a separate HMM. For tasks that are less constrained, but still have very small vocabularies, e.g., digit recognition, there may be one HMM per word. To evaluate the acoustic probability of a given hypothesis $\mathbf{w} = (w_1, ..., w_m)$, the separate word HMMs for $w_1, ..., w_m$ can then be concatenated, effectively constructing an HMM for the entire utterance. For most speech recognition tasks, though, the vocabulary will be too big for it to be feasible to use whole-word models (as there are typically not enough examples of most words in the training set to estimate the parameters of their HMMs reliably), and the words are broken down into subword units, usually phones, each of which is modeled with its own HMM.

Coarticulation effects are sometimes also included in the acoustic model, by using *context-dependent phones*.[4] In this case an `[aa]` preceded by an

[4]Context-dependent phones are sometimes referred to as *triphones* and, similarly, context-independent phones as *monophones*.

[x] and followed by a [y] is considered different from an [aa] preceded by an [x] and followed by a [z].

At every time step, the Markov chain transitions out of its current state and into a new one (possibly looping back to the same state as before). The HMMs used in ASR typically have a left-to-right topology, like the one shown in Figure 2.2, where you are allowed to stay in the same state, but not to go back to a state after you have moved on to the next state.

Also at each time step, the HMM emits an observation vector which represents the acoustic signal of the speech at this time. These observation vectors will generally not be easily modeled with a unimodal probability distribution, as speaker, accent and gender differences tend to create multiple modes in the data. Such multimodal distributions can be modeled as mixture distributions [McLachlan and Peel, 2000], where a linear combination of more basic probability distributions is used. In speech recognition applications, the basic distributions used are usually Gaussians, giving us what is referred to as a *Gaussian mixture model* (GMM).

A GMM is a superposition of K Gaussian densities of the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} c_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2.6}$$

where each Gaussian density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is called a component of the mixture and has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. The parameters c_k are called the *mixing coefficients*, and it is easily verified that for $p(\mathbf{x})$ to be a probability density function, these must satisfy the constraints

$$\sum_{k=1}^{K} c_k = 1, \quad 0 \leqslant c_k. \tag{2.7}$$

A GMM can be considered as random draws from a set of K Gaussian distributions; c_k is then the prior probability of making the draw from the kth Gaussian.

Figure 2.3 shows a one-dimensional GMM where $K = 3$. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the mixing coefficients, almost any continuous density can be approximated to arbitrary accuracy. For more details on GMMs, refer to Bishop [2006, Sections 2.3.9 and 9.2].

The Decoder

The decoder combines the knowledge sources to find the word sequence $\hat{\mathbf{w}}$ that is assigned the highest probability by our generative model. Letting $\mathbf{o}_t$

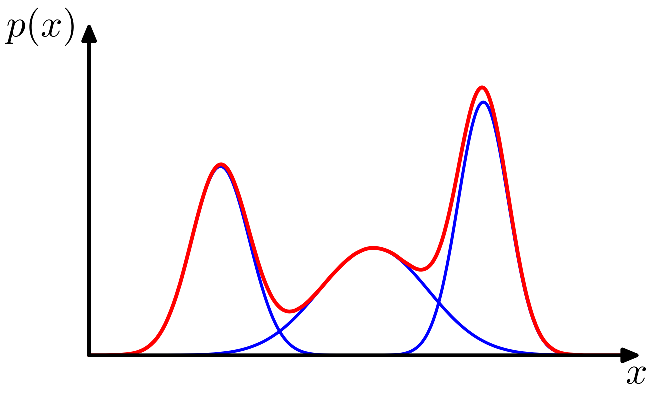


Figure 2.3: Example of a Gaussian mixture distribution in one dimension, showing three Gaussians (each scaled by a coefficient) in blue and their sum in red. (Figure from Bishop [2006].)

be the observation vector at time step t and $\mathbf{q} = q_1, ..., q_T$ be the sequence of hidden states in the HMM, and making some assumptions described below, we can rewrite the fundamental equation (2.3) as follows:

$$\begin{aligned}
\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} p(\mathbf{O}|\mathbf{w})P(\mathbf{w}) && (2.8)\\
&= \arg\max_{\mathbf{w}} \sum_{\mathbf{q}} p(\mathbf{O}|\mathbf{w},\mathbf{q})P(\mathbf{q}|\mathbf{w})P(\mathbf{w}) && (2.9)\\
&= \arg\max_{\mathbf{w}} \sum_{\mathbf{q}} p(\mathbf{O}|\mathbf{q})P(\mathbf{q}|\mathbf{w})P(\mathbf{w}) && (2.10)\\
&\approx \arg\max_{\mathbf{w}} \max_{\mathbf{q}} p(\mathbf{O}|\mathbf{q})P(\mathbf{q}|\mathbf{w})P(\mathbf{w}) && (2.11)\\
&= \arg\max_{\mathbf{w}} \max_{\mathbf{q}} P(\mathbf{q}|\mathbf{w})P(\mathbf{w}) \prod_{t=1}^{T} p(\mathbf{o}_t|q_t). && (2.12)
\end{aligned}$$

Equation (2.9) is just a rewriting of (2.8) where we sum over all the hidden state sequences $\mathbf{q}$ that are possible realizations of the hypothesis $\mathbf{w}$. Equation (2.10) follows from the assumption that the acoustics are independent of the words given the state sequence. In going from (2.10) to (2.11) we have made the so-called Viterbi approximation: that there is a single state sequence that is much more likely than all others, so that summing over $\mathbf{q}$ is approximately equivalent to maximizing over $\mathbf{q}$. And finally, Equation (2.12) follows from the assumption that given the current state q_t, the current observation $\mathbf{o}_t$ is independent of all other states and observations.

A hidden Markov model is a probabilistic *generative model*: it considers how likely the utterance is to have generated the observation vectors. This stands in contrast to *discriminative models*, which consider the probability of the utterance given the observation vectors.

For further details on the GMM-HMM framework, we refer the interested reader to Gales and Young [2007], Rabiner [1989], and Rabiner and Juang [1993].

The models described above are most often applied to the task of word recognition. However, there has also been a lot of work on *phone recognition*. Here the recognized string $\mathbf{w}$ will be a string of phones rather than a string of words. This effectively takes the pronunciation lexicon and the language model[5] out of the picture. It is therefore frequently used to investigate new kinds of acoustic models.

2.2 Multilayer Perceptrons and Tandem HMM Systems

Artificial neural networks, particularly the multilayer perceptron (MLP), have been widely used in speech recognition as part of models based on HMMs. Typically, MLP phone (or phone state) classifiers are used either in a so-called "hybrid approach" [Morgan and Bourlard, 1995; Bourlard and Morgan, 1994], in which HMM emission probabilities are replaced by scaled likelihoods computed from MLP phone posteriors; or in a "tandem approach" [Hermansky et al., 2000], in which the MLP log-posteriors serve as a replacement for (or an addition to) the raw acoustic observation vectors in a conventional GMM-HMM system. As mentioned, we present a tandem system for phone recognition in Section 4.3; we will therefore focus on tandem systems here, after giving a short presentation of the MLP.

Multilayer Perceptrons

The basic idea of an artificial neural network (ANN) is that inputs are fed into a network of nodes, each of which computes the value of an activation function with learned parameters, and this value is either passed on to the next layer in the network or given as an output of the network. This computational model is inspired by the brain, and the nodes are sometimes referred to as "artificial neurons".

MLPs are a kind of ANN which have a layered feedforward architecture, where each layer is fully connected to the next one, as shown in Figure 2.4.

[5]Although some language modeling can of course be done on the phonetic level.

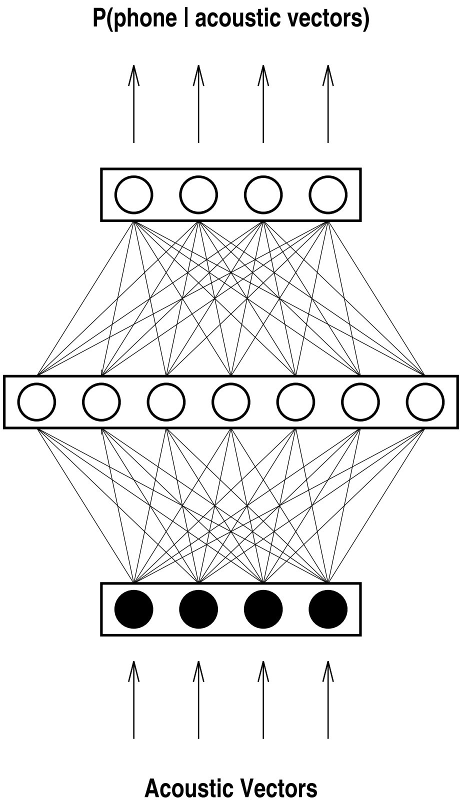


Figure 2.4: A multilayer perceptron for phonetic classification with one hidden layer. The filled circles represent the input nodes; the open circles represent nodes which compute a function of their inputs. (Figure from Morgan and Bourlard [1995].)

The input layer has one node for each input variable; there is then zero or more intermediate, or *hidden*, layers with freely chosen numbers of nodes; finally, the output layer has one node for each output variable. In the hidden and output layers each node evaluates the activation function for a weighted sum of its inputs. In a phone classification context, the input layer will have one node per dimension of the input vector. (The input vector, however, may be a concatenation of several acoustic observation vectors.) The output layer will have one node for each possible phonetic label, which will compute the posterior probability of the observation vector having the corresponding label.

The term "multilayer perceptron" is arguably a misnomer, since an MLP does not consist of a perceptron with multiple layers, but of multiple perceptrons organized into layers. Moreover, they are not perceptrons of the kind introduced by Rosenblatt [1962], which use a threshold activation function (with discontinuous nonlinearities), but can take any activation function, and in most current applications use a sigmoid (with continuous nonlinearities). When the most common sigmoid activation function,

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}} \,, \tag{2.13}$$

is used, an MLP could more correctly be characterized as multiple layers of logistic regression models. The logistic sigmoid (Figure 2.5b) can be viewed as a differentiable approximation to the threshold function (Figure 2.5a).

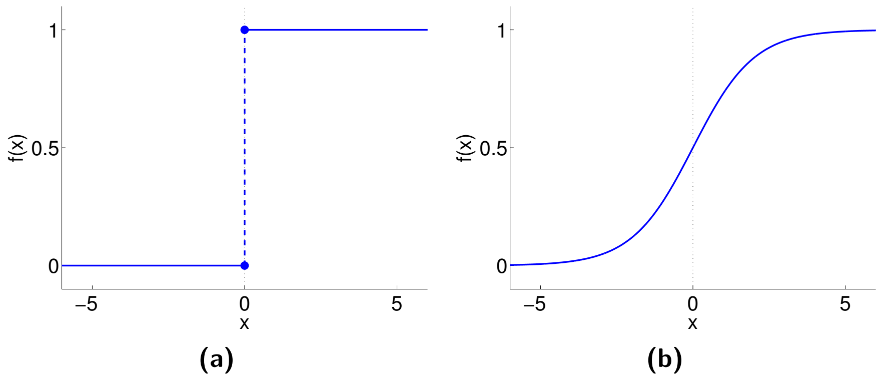


Figure 2.5: A discontinuous threshold activation function (a) and a continuous sigmoid activation function (b).

An alternative activation function which approximates the sigmoid function and is commonly used in the final layer, particularly for classification problems where there are more than two possible labels, is the *softmax function.*[6] For a layer with K nodes, this function is defined as

$$f(x_i) = \frac{\mathrm{e}^{x_i}}{\sum_{k=1}^{K} \mathrm{e}^{x_k}} . \tag{2.14}$$

This represents a smoothed version of a binary max function (because, if $x_k \gg x_j$ for all $j \neq k$, then $f(x_k) \approx 1$ and $f(x_j) \approx 0$), thence the name.

Consider an MLP using the softmax activation function being trained for a multiclass problem (e.g., phone classification). Let $\mathcal{T} = \{(\mathbf{o}_t, y_t)\}_{t=1}^{T}$ be a training set consisting of T ordered pairs, where $\mathbf{o}_t \in \mathbb{R}^m$ is an observation vector and $y_t \in \mathcal{Y}$ is its label, $\mathcal{Y}$ being the set of possible labels (e.g., the

[6] Also known as the *normalized exponential.*

phone set). Denote by $p_t(y)$ the empirical distribution of the classes y, given an input vector $\mathbf{o}_t$. Thus, $p_t(y) = 1$ if and only if $y = y_t$, and $p_t(y) = 0$ otherwise.

The number of output layer nodes is equal to the number of labels, $|\mathcal{Y}|$. Denote by $z_t(y)$, $y \in \mathcal{Y}$, the value of one output layer node before the application of the softmax function when the observation vector $\mathbf{o}_t$ is input to the network. $z_t(y)$, which is often referred to as the *linear output*, is a real number given by a weighted sum of the node's inputs from the previous layer in the MLP, determined by a weight vector, $\mathbf{w}_i$, which is specific to the node. Then the final output (after the softmax) of this node, $q_t(y)$, is given by

$$q_t(y) = \frac{\mathrm{e}^{z_t(y)}}{\sum_{y' \in \mathcal{Y}} \mathrm{e}^{z_t(y')}}. \tag{2.15}$$

The outputs $q_t(y)$ can be interpreted as an estimate of the posterior distribution over the classes $y \in \mathcal{Y}$ given the input vector $\mathbf{o}_t$.

Training consists of adjusting $\mathbf{W}$, the matrix of all the weight vectors $\mathbf{w}_i$, to minimize a cost function. One popular choice is the relative entropy (Kullback-Leibler divergence) between the empirical and estimated posterior distributions:

$$\mathcal{L}(\mathbf{W}) = \sum_{t=1}^{T} \sum_{y \in \mathcal{Y}} p_t(y) \ln \frac{p_t(y)}{q_t(y)}.^{7} \tag{2.16}$$

Another option is the mean square error criterion:

$$\mathcal{E}(\mathbf{W}) = \sum_{t=1}^{T} \|\mathbf{q}_t - \mathbf{p}_t\|^2, \tag{2.17}$$

where $\mathbf{q}_t$ and $\mathbf{p}_t$ are vectors of the estimated and empirical posterior probabilities of the classes—i.e., when $\mathcal{Y} = \{y_1^*, \ldots, y_K^*\}$ we have that

$$\mathbf{q}_t = \big(q_t(y_1^*), \ldots, q_t(y_K^*)\big)^{\mathrm{T}},$$
$$\mathbf{p}_t = \big(p_t(y_1^*), \ldots, p_t(y_K^*)\big)^{\mathrm{T}}.$$

The minimum of the chosen cost function is found through the technique of error back-propagation [Rumelhart et al., 1986], an iterative gradient descent procedure where the partial derivative of the cost function with respect to each weight element of $\mathbf{W}$ is used to shift that weight a little

[7]When $p_t(y) = 0$, $\mathcal{L}(\mathbf{W})$ is interpreted as 0, since $\lim_{x \to 0} x \ln x = 0$.

bit in the direction that reduces the value of the cost function. The size of this shift is governed by a parameter called the *learning rate*, which is a compromise between speed of learning and precision of result.

Training can be sped up by accumulating the error for a number of input observation vectors before back-propagating to update the weights. This is sometimes referred to as *bunch-mode training* [Bourlard and Morgan, 1998; Ellis, 2000] and can offer considerable speed-ups at a small cost in final precision.

Tandem Hidden Markov Models

In a tandem HMM system, the outputs of an MLP are used as an observation vector in the GMM-HMM model, instead of—or in addition to—acoustic vectors like MFCCs or PLPs.

For a given input vector, the MLP can output posterior probabilities of its belonging to each of the possible classes. For a tandem HMM, the MLP is typically trained to do frame classification of phones; i.e., for each frame of the audio signal the MLP estimates the probabilities of this frame being part of an utterance of every phone in our vocabulary. When the vocabulary consists of K phones this gives us a K-dimensional vector.

Because the distribution of the posterior probabilities is very skewed, which can be a problem for fitting the Gaussian mixture models, log probabilities are often used. Alternatively, one can use the linear outputs of the MLP, i.e., the output before the application of the sigmoid transformation. As we can see from (2.15), when using the softmax activation function, the natural logarithm of the probability estimate $q_t(y)$ is

$$\ln\big(q_t(y)\big) = z_t(y) - \ln\left(\sum_{y'\in\mathcal{Y}} \mathrm{e}^{z_t(y')}\right). \tag{2.18}$$

Since $\ln x$ grows very slowly when $x > 1$, this means that the log probabilities $\ln q_t(y)$ and the linear outputs $z_t(y)$ will in most cases be very close.

As mentioned, the GMM-HMM framework is a generative model, whereas discriminative models are often more desirable. Since an MLP classifier is discriminative, tandem HMM models represent one way of incorporating discriminative elements into the GMM-HMM framework.

The frame classification accuracy of MLPs is often improved when the input observation vectors are concatenated. E.g., instead of using one 39-dimensional vector, the observation vectors for the four preceding and four succeeding frames can be appended to the original vector, yielding one high-dimensional vector representing the acoustic signal over 9 frames. When such an MLP forms the basis for a tandem system, the MLP is used as a form of discriminative, nonlinear dimensionality reduction.

Tandem HMM systems have been widely used and have generally given good results. In the paper that introduced the tandem framework Hermansky et al. [2000] report a mean 35% reduction in word error rate relative to their GMM-HMM baseline on the Aurora digit recognition tasks with varying degrees of background noise (whereas their hybrid system achieved only 15%). Ellis et al. [2001] report error rate reductions of over 50% relative to the HMM baseline, while Sivadas et al. [2004] report a 15% relative reduction in error rate on task-independent data.

Lately, a new class of artificial neural networks has been gaining popularity: *deep neural networks* (DNNs) are neural networks with more hidden layers than is commonly used in MLPs where some pre-training is sometimes done [Bengio, 2009; Seide et al., 2011; Hinton et al., 2012]. These have also been used in tandem HMM systems [Vinyals and Ravuri, 2011; Zheng et al., 2013]. In this thesis, we focus on the classic MLP tandem framework.

2.3 Exemplar-Based Speech Recognition and Nearest Neighbor Methods

The ASR systems presented in the previous sections can all be regarded as examples of *global data modeling*, in the sense that they use all the available training data to build a single, global model before the test sample is seen. *Exemplar-based modeling* represents an alternative approach, where we judiciously select a subset of exemplars from the training data to build a local model specifically for every test sample. In this section we will give a brief presentation of the use of exemplar-based modeling in ASR, with particular emphasis on the nearest neighbor method—which is arguably the most popular and fundamental exemplar-based model, and plays a central role for the work in this thesis.

Deriving a single global model relies on the assumption that averaging across all the training examples results in a representation that is still reliable; this is questionable. For parametric models like GMM, there are also inherent assumptions about the data following specific probability distributions; these may be tenuous. Exemplar-based techniques, on the other hand, tend to forego such assumptions completely. One can say they build several local models, each of which does not attempt to encompass all of the training data; or one can say they do not build models at all.

Global models may be problematic when there is limited training data. If there are few examples in the training set for one of the classes, the model parameters estimated for this class can be unreliable. Because exemplar-based techniques build local models using only a few relevant training examples, they are not as vulnerable to data sparsity issues. On the other hand, very large amounts of training data may also pose problems for global models, since the model chosen will often be incapable of representing the fine detail in the distribution of the data [Sainath et al., 2012]. Since the local models of exemplar-based techniques do not try to encompass all of the data, they can be able to include details that would be averaged out in a global model.

In a recent review article, Sainath et al. [2012] mentioned four exemplar-based models that have found applications in ASR. *Sparse representations* [Wright et al., 2009; Sainath et al., 2010; Gemmeke et al., 2011] involve modeling the observed test vector as a linear combination of the vectors in the training set, or a subset of these. *Template matching* [Brunelli and Poggio, 1993; De Wachter et al., 2007; Demuynck et al., 2011a] compares the sequence of observed test vectors to a set of variable-length reference templates selected from the training data. The selection is usually done via a quick and approximate template matching or via a classical HMM system. Because the reference templates can be of different lengths, dynamic time warping [Sakoe and Chiba, 1978; Myers and Rabiner, 1981] is used to compare them with the test vectors. *Latent perceptual mapping* [Sundaram and Bellegarda, 2010, 2012] is an ASR technique derived from latent semantic analysis [Deerwester et al., 1990] in information retrieval. It uses vector quantization of the observation vectors to express them within a vector space spanned by a discrete set of acoustic units, and then employs a singular value decomposition of this space to reduce its dimensionality. In this way it attempts to derive prototypical acoustic units in a data-driven manner, and thus achieve a parsimonious template-like solution.

The fourth exemplar-based model mentioned by Sainath et al., *k-nearest neighbors*, which will be the focus of the rest of this section, is arguably the most fundamental of them. It is often used as a seed for both sparse representations and latent perceptual mapping, and as part of the decoding process of template matching systems.

k-Nearest Neighbors Classification

Nearest neighbor classification is among the simplest techniques in machine learning. Given a test observation vector that we want to classify, we find the observation vector in the training set that lies closest to it according to some distance metric—i.e., its nearest neighbor—and we assign the test vector to the same class as that of this training vector. This technique has no parameters and makes no assumptions about the underlying probability distributions. One needs only a representation of the observations in a vector space with a metric. The vectors are typically m-dimensional and real-valued, and the Euclidean distance is a common choice for the metric, but there are many other options for both. Nearest neighbor classifiers also involve no training; the training data is simply stored, and test points are classified by finding the nearest neighbors to it among the training points. For this reason, the training set is sometimes referred to as the *search set* and the test set as the *query set* in the context of k-NN classification.

k-nearest neighbor (k-NN) classification is a generalization of the nearest neighbor technique where we find the test vector's k nearest neighbors instead of just the single nearest one, and decide its class by majority vote among these neighbors. When $k = 1$ this, of course, reduces to single-nearest neighbor classification, which can be referred to as 1-NN classification. The basic k-NN classification algorithm is stated formally in Figure 2.6.

There are several well-known results on the theoretical performance of the k-NN classifier. The *risk* associated with a classifier is defined as the expected value of an appropriate loss function—in our case the loss function will simply count the number of errors. Cover and Hart [1967] showed that the asymptotic risk of the 1-NN classifier is upper bounded by $2R^*(1 - R^*)$, where R^* is the Bayes risk, which is the lowest risk achievable by any classifier. This entails that *any* other classifier can cut the asymptotic probability of error by at most one half. It can thence be claimed that half of the information contained in a training set of infinite size is contained in the single nearest neighbor. Loizou and Maybank [1987] showed that the k-NN classifier is asymptotically optimal if either very little or very much training data is available.

In addition to its theoretical strengths, the k-NN classifier has many practical advantages: there is no training step, there are very few parameters to tune, and the classifier is easily implemented. Due to its simplicity, it also lends itself more easily to detailed examinations of error patterns than is the case for less transparent classifiers like the MLP.

The k-NN classifier's main drawback is that it is relatively slow. It is an instance of what is sometimes referred to as "lazy learning", in that all the work is done at test time. A less famous result by Cover is that the rate of convergence to the $2R^*(1 - R^*)$ bound can be arbitrarily slow [Cover, 1968]. There are, however, several methods available to speed up k-NN search, such as k-d trees [Friedman et al., 1977], ball trees [Omohundro,

k-NN Classification Algorithm

- Given:
  - An observation space $\mathcal{X}$ and a label space $\mathcal{Y}$
  - A labeled training set $\mathcal{T} = (\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y})$, $i = 1, ..., N$
  - A metric $\mathcal{D}_x : (\mathcal{X} \times \mathcal{X}) \mapsto \mathbb{R}$
  - A natural number $k \leqslant N$
  - A test example $\mathbf{x}_0 \in \mathcal{X}$
- Let $i_1^*, ..., i_k^*$ be the indices of the k nearest neighbors of $\mathbf{x}_0$ in $\mathcal{X}$ with respect to $\mathcal{D}_x$ in order of increasing distance, i.e.:

$$\mathcal{D}_x(\mathbf{x}_0, \mathbf{x}_{i_1^*}) \leqslant ... \leqslant \mathcal{D}_x(\mathbf{x}_0, \mathbf{x}_{i_k^*}), \tag{2.19}$$

and

$$\forall i \notin \{i_1^*, ..., i_k^*\} : \mathcal{D}_x(\mathbf{x}_0, \mathbf{x}_i) \geqslant \mathcal{D}_x(\mathbf{x}_0, \mathbf{x}_{i_k^*}.). \tag{2.20}$$

- For each $y \in \mathcal{Y}$, let X_y be the set of the k nearest neighbors of $\mathbf{x}_0$ that belong to class y, i.e.:

$$X_y = \{\mathbf{x}_{i_j^*} : y_{i_j^*} = y, 1 \leqslant j \leqslant k\}. \tag{2.21}$$

- Predict the class of $\mathbf{x}_0$ by majority vote, i.e.:

$$\hat{y}_0 = \arg\max_{y \in \mathcal{Y}} |X_y|, \tag{2.22}$$

where $|X_y|$ denotes the number of elements in the set X_y.

Figure 2.6: Formal statement of the algorithm for k nearest neighbor classification.

1989; Liu et al., 2004], locality sensitive hashing [Indyk and Motwani, 1998; Andoni and Indyk, 2006] and the spatial approximation sample hierarchy [Houle and Sakuma, 2005].

Previous Work with Nearest Neighbors in ASR

The k-NN technique has been very popular in diverse pattern recognition applications for a number of years, but although very early speech recognition system used nearest neighbors, k-NN has seen relatively few applications in ASR until recently. The works of Lefèvre [2003] and Pellom et al. [2001] represent early attempts to improve a GMM-HMM ASR system with the help of k-NN classifiers. There are many other examples of recent work where k-NN plays a more or less major part [Glass, 2003; Gemmeke et al., 2009; Sainath et al., 2011; Singh-Miller and Collins, 2009]. We will here go into some detail on the work of three research groups, where k-NN plays a central role.

Frame-Based k-NN in an HMM Recognizer [Deselaers et al., 2007]
Deselaers, Heigold and Ney's 2007 paper represents an attempt to apply nearest neighbor classification of phone states in an ASR system. They took their motivation from the fact that modern ASR systems tend to use GMMs with a very high numbers of densities. If we increase the number of densities to the number of training vectors, we are effectively performing 1-NN classification. Deselaers et al. tried three different approaches for replacing GMMs with nearest neighbors in the modeling of the emission probability $p(\mathbf{o}|q)$ (i.e., the probability of the observation vector $\mathbf{o}$ being generated by state q).

Their first probability estimate was based on the squared Euclidean distance to the single nearest neighbor of class q. Let $\mathcal{Q}$ be the set of time indices of the observation vectors in the training set of class q, then

$$\hat{p}_{1\text{-NN}}(\mathbf{o}|q) = \exp\left(-\min_{i\in\mathcal{Q}} \|\mathbf{o}-\mathbf{o}_i\|^2\right), \tag{2.23}$$

where $\|\cdot\|$ is the Euclidean distance. Their second approach was to average this distance between the k nearest neighbors of class q. Let $\mathcal{K}_q$ be the set of time indices of the k observation vectors in the training set of class q that are closest to $\mathbf{o}$, then

$$\hat{p}_{k\text{-NN}}(\mathbf{o}|q) = \exp\left(-\frac{1}{k}\sum_{i\in\mathcal{K}_q} \|\mathbf{o}-\mathbf{o}_i\|^2\right). \tag{2.24}$$

This turned out to be detrimental to the results compared to the first approach. Their third probability estimate was a k-NN approximation to kernel densities [Silverman, 1986], and given as

$$\hat{p}_{\text{kern}}(\mathbf{o}|q) = \frac{1}{k} \exp\left(-\sum_{i \in \mathcal{K}_q} \frac{1}{2} \frac{\|\mathbf{o} - \mathbf{o}_i\|}{c} \right), \tag{2.25}$$

where c is a scaling factor. This third approach gave the best performance.

The results for the k-NN-based HMM system were on par with that of the GMM-HMM system for a corpus of German continuous digit strings. On a large-vocabulary English corpus of recordings from the European Parliament, the k-NN-based system outperformed the GMM-HMM when 3 hours or less of training data was available.

Segmental k-NN in an HMM Recognizer [Golipour and O'Shaughnessy, 2009, 2010, 2012] The work of Golipour and O'Shaughnessy started out from a k-NN classification of phones and went on to apply the classification results in HMM lattice rescoring as well as in a complete, non-parametric phone recognizer. In the classification, the phone boundaries were assumed known, and each phone segment represented by an observation vector of fixed dimensionality.[8] They divided the duration of each phone utterance into several parts, averaged the MFCC coefficients over each part and concatenated the resulting mean vectors. In their initial work [Golipour and O'Shaughnessy, 2009], they used a 42-dimensional observation vector consisting of 14-dimensional MFCCs averaged over three parts for each phone. In their later work [Golipour and O'Shaughnessy, 2010, 2012], they expanded their observation vectors to 253 dimensions, averaging 42-dimensional MFCCs over six separate parts for each phoneme and adding the duration of the phoneme as a final dimension. They also switched from a voting k-NN classifier (like the one described in Figure 2.6) to a *volumetric* k-NN classifier [Fukunaga, 1990], where one assigns the test exemplar $\mathbf{o}$ to the class for which the volume of the hypercube around $\mathbf{o}$ containing k neighbors of this class is the smallest. Their phone classification error on on the TIMIT corpus of continuous read speech[9] was substantially reduced.

[8]Note the difference between this and the phonetic *frame classification* experiments in this thesis, which arguably represent a harder task since the observation vectors in that case are extracted from a single frame, rather than from all of the frames within the phone segment, and thus contain less evidence.

[9]We describe the TIMIT speech corpus in Section 4.3.1

In their lattice rescoring work, Golipour and O'Shaughnessy [2010] used their k-NN phone classifier to classify the segments inside the phone lattices generated by a GMM-HMM phone recognizer. If the k-NN classification was different from the HMM's hypothesis, a cost proportional to the length of the phone was added to the HMM acoustic score. This approach improved the phone error rate on TIMIT of a GMM-HMM context-independent phone recognizer from 39.7% to 37.7%. However, the improvement for a similar context-dependent phone recognizer was far smaller, from 27.2% to 26.8%, due to the fact that the k-NN classifier used only the acoustic information of separate phones, while the recognition results in the context-dependent lattices were based on each phone and its preceding and succeeding phones.

In Golipour and O'Shaughnessy [2012], a phone recognition system that did away with the HMM framework entirely was presented. Instead of rescoring phone lattices from an HMM, they used their phone segmentation technique [Golipour and O'Shaughnessy, 2007] as a basis. They divided the boundaries from the segmentation into a set A with confident boundaries, and a set B with less confident boundaries. They computed the initial MFCC observation vectors based on the boundaries in A, and classified these with k-NN. A directed acyclic graph (DAG) similar to a phone lattice was constructed from this classification and the durational information from the segmentation. Each node in the graph represented a boundary in the set A, and each edge represented a phone hypothesis for which a score was computed. The constructed DAG was then modified: first, a *missed boundary compensation* step added nodes representing boundaries from the set B, and edges representing segments that split the existing ones; then, an *over-segmentation compensation* step iteratively added edges representing segments that merged existing ones. Whenever an edge was added, an observation vector was computed for the corresponding new segment, which was then classified by k-NN and a score was computed for the edge. Finally, Dijkstra's algorithm [Dijkstra, 1959] was used to find the shortest path in the DAG, yielding the hypothesized phone sequence. The resulting phone recognition accuracy on TIMIT was well below state-of-the-art results, but Golipour and O'Shaughnessy argued that their method had a substantially smaller search space and a very limited amount of score computations.

k-NN in a Template-Based ASR System [De Wachter et al., 2007; Demuynck et al., 2011a,b] De Wachter et al. [2007] presented a template-based large-vocabulary continuous speech recognition system that built on the Dynamic Time Warping algorithm (DTW) [Sakoe and Chiba, 1978]—a well-known technique for aligning and comparing a speech segment to a template of a different duration. To limit the search space of the DTW, a data-driven selection of candidates for DTW alignment was done. Initially, an approximate k-NN search (using the Roadmap algorithm [Povey and Woodland, 1999]) was done to find a small number of neighbors of the observation vector of the input frame. Then, a time filter algorithm found template candidates by examining how well the preceding and succeeding frames of each nearest neighbor match those of the input frame. When concatenating templates, one of the factors taken into account was the templates' meta-information, like the speaker's gender. On a corpus of continuous read speech with a 1000-word vocabulary [Price et al., 1988], the system achieved a word error rate almost on par with an HMM baseline.

Demuynck et al. [2011a; 2011b] presented several improvements to this system. The system was sensitive to outliers and errors in the training database; an averaging scheme was adopted to mitigate this so that the single best score was replaced by a weighted k-NN average. The DTW scoring was also improved by promoting acoustic continuity when selecting the k-NN templates. Along with the other improvements presented in Demuynck et al. [2011a], this gave a 15% relative decrease in word error rate on a corpus of continuous read speech from the Wall Street Journal with a 20 000-word vocabulary [Paul and Baker, 1992]. An advantage of exemplar-based systems over those that build a global statistical model is that one can easily derive a wealth of meta-information about each exemplar. Such meta-information from the nearest neighbors was integrated in the recognition system using the SCARF toolkit [Zweig and Nguyen, 2010] for segmental conditional random fields [Zweig and Nguyen, 2009],[10] reported in Demuynck et al. [2011b]. For each of the nearest neighbor templates, the authors considered: its position in the training database, the phone label, the phone duration, the speaker ID, the word the phone originated from, and its position in the word. SCARF was used to rank features computed from this meta-information, and to combine them with the recognition score from the template-based recognition system. This gave a further 7% relative reduction in the word error rate.

2.4 Graphical Models in Speech Recognition

Unlike exemplar-based techniques, graphical models are examples of global data modeling. The work that we will present in Sections 4.1 and 4.2, though, is concerned with incorporating the results of a k-NN classification

[10]We present conditional random fields in Section 2.4.2.

in a conditional random field, which is one kind of graphical model. This is one reason why we give a brief presentation of graphical models here. Another reason is that graphical models feature prominently in articulatory speech technology, as we shall go on to see in Section 2.5.

A graphical model (GM) is a graphical abstraction of a statistical model where important aspects of the model are represented using the two-dimensional visual formalism of a graph. The semantics of the GM determines how the graph's vertices and edges encode properties of the statistical model. These properties are usually factorization constraints. Many probability distributions will fit any given GM; the GM can thus be regarded as a filter which accepts some distributions and rejects others. GMs provide us with a simple and visually intuitive way of expressing complicated statistical ideas. There are two main kinds of GMs: *directed graphical models*, of which Bayesian networks are the dominant kind; and *undirected graphical models*, also called Markov networks or Markov random fields. In this section, we will give a brief presentation of the variants of these two that are most commonly used in ASR: dynamic Bayesian networks and conditional random fields.

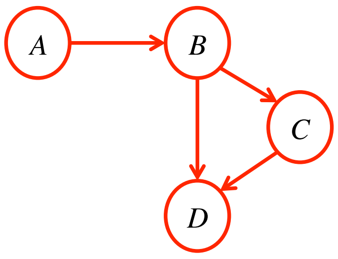


Figure 2.7: A Bayesian network for four random variables, which shows that the variable C is independent of the variable A conditioned on the variable B—i.e., that $C \perp\!\!\!\perp A \mid B$. Similarly, this network shows that $D \perp\!\!\!\perp A \mid B$.

2.4.1 Dynamic Bayesian Networks

A *Bayesian network* (BN) is a directed acyclic graph that encodes the independence properties of a joint probability distribution. Each node in the graph represents a variable, and a directed edge from one node (the parent) to another (the child) signifies a conditional dependency. If we have four random variables A, B, C and D, then we know that their joint density $p(a, b, c, d)$ can be factorized as

$$p(a, b, c, d) = p(a) \cdot p(b \mid a) \cdot p(c \mid a, b) \cdot p(d \mid a, b, c),$$

regardless of their distribution. If we know, however that their distribution is as described by the BN in Figure 2.7, then the factorization simplifies to

$$p(a, b, c, d) = p(a) \cdot p(b \mid a) \cdot p(c \mid b) \cdot p(d \mid b, c),$$

since the graph shows that C is independent of A conditioned on B, and D is independent of A conditioned on B and C.

Although the edges in a BN show all the conditional dependencies, it is not entire straightforward to ascertain whether a particular conditional independence statement holds for the distributions expressed by the graph. The *Markov blanket* [Pearl, 1988] of a node A in a graphical model is the minimal set of nodes ∂A such that any other node in the BN is conditionally independent of A given ∂A. For a BN, the Markov blanket of A is composed of A's parents, its children and its children's other parents.

A *Dynamic Bayesian network* (DBN) is a BN with a repeated template structure that can capture the temporal evolution of the phenomenon being modeled. For temporal processes such as speech, this is essential. The template can be repeated for each frame of the speech signal, and variables in a frame can depend on variables in previous frames. A hidden Markov model is one of the simplest possible DBNs—the repeated template in an HMM consists of two variables: the hidden state and the acoustic observation vector. However, the expressive power of DBNs far exceed those of the HMM [Bilmes, 2006].

There are further advantages to working with more general graphical models than HMMs. There can be exploitable computational advantages to working with graphical models that explicitly represent factorization properties as factorization is the key to tractable probabilistic inference. A graphical model like a DBN can also express constraints that the model must obey and thus reduce the amount of parameter freedom. Furthermore, a DBN can convey structural information about the underlying problem in a visual and intuitive way. Its explicit representation, where separate random variables can exist for such purposes as word identification, word position, phone identity and indication of a phone transition, stand in contrast to the implicit representation of HMMs, where the structure of the speech recognition model is more opaque [Bilmes and Bartels, 2005].

2.4.2 Conditional Random Fields

A conditional random field is a special case of a Markov random field (MRF) [Kindermann and Snell, 1980]. An MRF is similar to a Bayesian network in that it is a graph where the nodes represent variables and the edges represent dependencies. The edges, however, are undirected, and conditional independence is determined by simple graph separation. For disjoint sets of nodes $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$, we have that $\mathcal{A}$ is conditionally independent of $\mathcal{B}$ given $\mathcal{C}$ if and only if no node in $\mathcal{B}$ can be reached from any node in $\mathcal{A}$ without passing through a node in $\mathcal{C}$. The Markov blanket of a node in an MRF is simply the set of its adjacent nodes.

A *clique* is a subset of the nodes in a graph that is fully connected, i.e., there exists a link between any pair of nodes in the subset. A *maximal clique* is a clique such that no more nodes from the graph can be included in it without it ceasing to be a clique. In Figure 2.8a, the pairs $\{x_1, x_2\}$, $\{x_2, x_3\}$, and $\{x_1, x_3\}$ are all cliques, but $\{x_1, x_2, x_3\}$ is the only maximal clique. The factors in the decomposition of the joint probability distribution expressed by an MRF can be defined as functions of the variables in its maximal cliques. For a given MRF, let $\mathbf{x}$ be the set of its variables, $p(\mathbf{x})$ their joint distribution, $\mathcal{C}$ the set of maximal cliques, and if $c \in \mathcal{C}$ then let $\mathbf{x}_c$ be the set of variables in the maximal clique c. The joint distribution can then be written as a product of *potential functions* $\psi_c(\mathbf{x}_c)$

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c), \tag{2.26}$$

where Z, often called the *partition function*, is a normalization factor. If $\mathbf{x}$ is comprised of discrete variables, Z is given by

$$Z = \sum_{\mathbf{x}} \prod_{c} \psi_c(\mathbf{x}_c). \tag{2.27}$$

The factors in Bayesian networks represented the conditional distribution of the corresponding variable conditioned on the state of its parents, thus no normalization factor was necessary. The potential functions in an MRF are not required to have any probabilistic interpretation. The only restriction placed on them is that they are to be strictly positive.

The factorization of an MRF is still ambiguous—several factorizations of the joint distribution are possible given an MRF. *Factor graphs* remove this ambiguity and thus give a more detailed picture of the underlying distribution. A factor graphs adds a new kind of node (depicted by small squares) for each factor in the joint distribution. Instead of being connected to each other, the variable nodes are now connected to every node representing a factor that depends on them. This is illustrated in Figure 2.8 where two different factor graphs are shown for a simple MRF.

A *conditional random field* (CRF) [Lafferty et al., 2001; Sutton and McCallum, 2011] is an MRF over two disjoint sets of variables $\mathbf{x}$ and $\mathbf{y}$ that

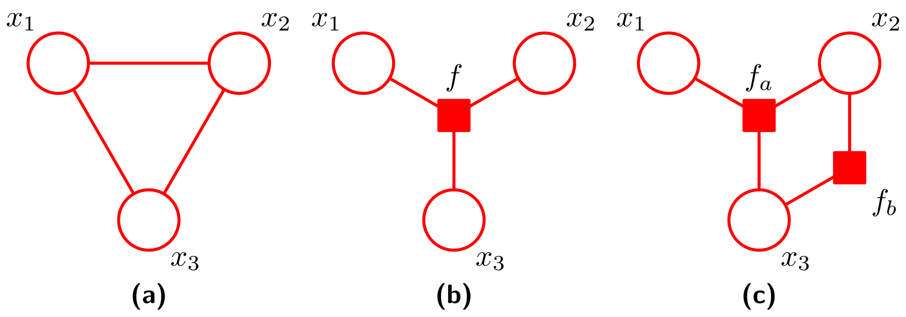


Figure 2.8: One Markov random field and two factor graphs representing the same distribution. (a) An MRF with a single clique potential $\psi(x_1, x_2, x_3)$. (b) A factor graph with one factor $f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$. (c) A factor graph whose factors satisfy $f_a(x_1, x_2, x_3) f_b(x_2, x_3) = \psi(x_1, x_2, x_3)$. (Figure from Bishop [2006].)

models the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$, where $\mathbf{x}$ consists of observed variables and $\mathbf{y}$, often referred to as the *output variables*, can consist of any random variables. This implies that a CRF is inherently a discriminative model. By modeling the conditional distribution directly, the CRF can remain agnostic about the form of $p(\mathbf{x})$. In contrast, a generative model, being a full probabilistic model of all variables, must model the full joint distribution $p(\mathbf{x}, \mathbf{y})$. If we want to change a factor graph representing the joint distribution $p(\mathbf{x}, \mathbf{y})$ to represent the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$, we can eliminate any factors that depend only on $\mathbf{x}$—they are constant with respect to $\mathbf{y}$ and are therefore irrelevant to $p(\mathbf{y} \mid \mathbf{x})$. Since modeling dependencies between the input variables $\mathbf{x}$ is often difficult to do while maintaining tractability, this arguably makes discriminative models such as CRFs better suited to include rich, overlapping features than generative models like DBNs [Sutton and McCallum, 2011].

2.5 Articulatory Speech Technology

The phonetic GMM-HMM paradigm can be said to be underpinned by a *linear phonology*, where the representation of an utterance is always a single string of symbols, and pronunciation variation is modeled using rules for the substitution, insertion and deletion of these. *Articulatory phonology*, proposed by Browman and Goldstein [1986; 1992], represents an alternative. Here, the basic units are articulatory *gestures*. These gestures can be thought of as instructions to the vocal tract to produce a certain degree of constriction at a given location with a given set of articulators—e.g.,

tract variable		articulators involved
LP	lip protrusion	upper & lower lips, jaw
LA	lip aperture	upper & lower lips, jaw
TTCL	tongue tip constrict location	tongue tip, body, jaw
TTCD	tongue tip constrict degree	tongue tip, body, jaw
TBCL	tongue body constrict location	tongue body, jaw
TBCD	tongue body constrict degree	tongue body, jaw
VEL	velic aperture	velum
GLO	glottal aperture	glottis

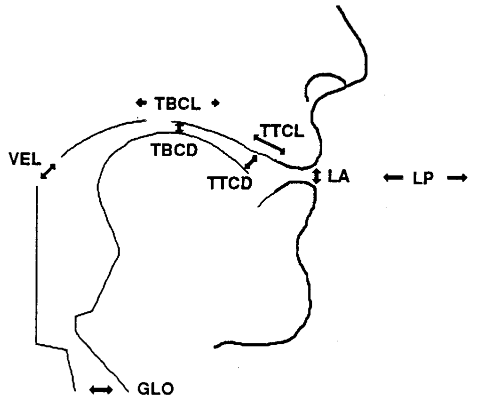


Figure 2.9: The articulatory tract variables and their relation to the physical articulators, as defined by Browman and Goldstein [1989].

"make a narrow lip opening." These gestures are expressed in terms of set of continuous-valued *tract variables*, shown in Figure 2.9.

There has been a gradual increase of work inspired by articulatory phonology in the speech technology community, and the work in this thesis forms part of it. We present experiments with classifying a set of articulatory features in Chapter 3, and the two automatic transcription systems we present in Chapter 4 are both articulatory in nature. In this section we will give a brief presentation of some previous work within this area, focusing first on the classification of articulatory features, then on articulatory ASR systems.

2.5.1 Acoustic-Articulatory Inversion and Articulatory Feature Classification

Acoustic-articulatory inversion (also called speech inversion or just articulatory inversion) gets its name from seeking to invert the process of speech production: given a speech signal, one tries to recover the sequence of articulatory configurations that produced it. This task has been addressed for many reasons, including on its own as a scientific question and as an intermediate step in articulatory approaches to automatic speech recognition. The work in this field falls within two broad categories: those that represent the articulatory configurations with continuous variables (which can be measurements from an instrument like an electromagnetic articulograph) and those that represent them with discrete variables (which can be binary or multi-valued). We will refer to the former variables as *articulatory parameters* and the latter as *articulatory features*. The term *inversion* is most commonly applied only to articulatory parameters. For articulatory features, one tends to speak of *classification* or *recognition*; if the features are binary, *detection* is also a commonly used term.

Work on recovering articulatory parameters from the acoustic signal dates back to the 1970s (see e.g., Wakita [1979] and the references therein). Examples of recent efforts include the work of Qin and Carreira-Perpiñán [2009; 2010], Richmond [2006; 2007], McGowan and Berger [2009], Mitra et al. [2011; 2014], Ghosh and Narayanan [2011], and Ananthakrishnan and Engwall [2011a; 2011b].

A different approach is to parameterize the articulatory configurations with discrete rather than continuous variables—using what is typically referred to as *articulatory features* (AFs). As there are no large speech corpora available that have been annotated on the articulatory level, the ground-truth labeling for AFs is usually derived from a phonetic transcription, by mapping each phone to corresponding AFs. This method is prone to errors, particularly around phone boundaries.

Classification of one particular set of AFs will be one of the main themes of this thesis. In the following, we will review some related work.

Many different sets of articulatory features have been used in the literature. **King and Taylor [2000]** report frame classification results for a set of 13 binary AFs based on the distinctive features from *Sound Pattern of English* (SPE) by Chomsky and Halle [1968]. The classification was done with recurrent neural networks. These differ from feedforward neural networks, like the MLPs presented in Section 2.2, in that connections between the nodes form a directed cycle. These recurrent connections have a time-delaying effect which gives the network some "memory" from one frame to the next. This lets the classifier take some previous context into account. A single network was trained to recognize all the features simultaneously. The experiments were performed on the TIMIT corpus with 13-dimensional MFCCs as observation vectors The mean frame error rate for these AFs was

8%. In the same paper, King and Taylor also report frame classification results for a multi-valued AF set with eight features based on IPA categories (centrality, continuant, front-back, manner, phonation, place, roundness, and tenseness) which have cardinalities between 2 and 10. The experimental setup was the same, except that for the classification of these features, a separate neural network was trained for each AF. The mean frame error rate, at 14%, was worse for these features—which is natural given the higher cardinalities. That the number of frames where all AFs are correctly classified were about the same for the two AF sets (52% and 53%) is an indication that the performance of the neural network classifiers was quite similar for them.

Kirchhoff [1999; 2002] used a set of five AFs (voicing, manner, place, front-back, rounding) with cardinalities between 3 and 10 which were classified using MLPs and GMM-HMMs; MLPs gave a slightly superior performance. A separate neural network was trained for each AF. The OGI Numbers corpus [Cole et al., 1995] of read numbers with detailed phone-level transcription was used, and classification was done for both clean and noisy speech. On clean speech, the mean frame error rate for the AFs was 19.0%.

Metze and Waibel [2002] used the 76 linguistically motivated questions that were used during construction of the decision tree for context-dependent phone modeling as their set of binary AFs. Rather than train a separate classifier like an MLP, detectors for these articulatory features were built in the same way as acoustic models for existing speech recognizers, using a GMM-HMM system. In joint work with Stüker and Schultz [Stüker et al., 2003a], they did similar experiments across several languages—demonstrating how articulatory modeling can be somewhat language independent. In later work by Metze [2007], the number of AFs was reduced to a set of 68 binary features.

The Johns Hopkins University (JHU) Summer Workshop of 2006 had a research group for "Articulatory Feature-Based Methods for Acoustic and Audio-Visual Speech Recognition" [Livescu et al., 2007a,b; Frankel et al., 2007a] where two separate AF sets were used. Only one of these was classified from audio, however. This set included seven traditional articulatory features (place, degree, nasality, rounding, glottal state, height, and frontness) with cardinalities between 3 and 10, in addition to a separate feature for vowel classification with cardinality 23. Frame classification was done on 2000 hours of conversational speech data in a cross-validation setup, with 39-dimensional MFCCs concatenated for 9 frames, giving 351-dimensional observation vectors for each frame. MLPs were used as classifiers, trained separately for each of the 8 articulatory features. The mean frame error rate was 19.5%.

While both of the works described above use only neural network classifiers, **Frankel, Wester and King** also use dynamic Bayesian networks for AF classification and recognition. Their set of six AFs (manner, place, voicing, rounding, front-back , and static) with cardinalities between 3 and 8 is inspired by both the theory of distinctive features of Chomsky and Halle [1968] and by the articulatory phonology of Browman and Goldstein [1992]. Like Kirchhoff et al., Frankel et al. used the OGI Numbers corpus [Cole et al., 1995]. Their initial frame classification experiments were with recurrent neural networks, yielding a mean frame error rate of 14% for the six AFs. In Frankel et al. [2004], they compared these results with those of a DBN for frame classification. They started out with a DBN model where the six AFs are independent of each other and compared this to a DBN model with nine manually added dependencies between them. The DBN model with dependencies performed better than the independent DBN model, but worse than the neural network classifier. Their later work [Frankel et al., 2007b] expanded on this methodology in several ways. They alleviated the problems inherent in training on phone-derived AF labels by including embedded training, and they expanded their work to include AF recognition in addition to AF frame classification. Their best results, 11% mean error rate for frame classification and 12% for recognition, are achieved by combining the neural networks with the DBNs.

Siniscalchi et al. [2007; 2012; 2013a] have done a series of experiments on classifying similar sets of around 20 binary AFs based on SPE [Chomsky and Halle, 1968], using a variety of classifiers. In Siniscalchi et al. [2007], MLPs with one hidden layer with 500 nodes were used for the frame classification. In a later paper [Siniscalchi et al., 2012], it was demonstrated that a similar system was relatively robust across six European and Asian languages. The MLPs are expanded to a hierarchical structure in Siniscalchi et al. [2013b]. Here, observation vectors are created for 155 ms windows to the left and to the right of the current frame; the output of these two lower MLPs are then sent to a third MLP that acts as a merger. The mean frame error rate for the AFs was 5.2% on a corpus of continuous read speech from the Wall Street Journal with a 5000-word vocabulary [Paul and Baker, 1992]. When deep neural networks were used instead of single-layer MLPs in a recent paper [Siniscalchi et al., 2013a], this error rate was reduced to 4.5%.

2.5.2 Articulatory Speech Recognition

We will now go on to consider how articulatory features have been used to improve automatic speech recognition systems. Although some attempts have been made at using continuous articulatory parameters in ASR systems (see e.g. Wrench and Richmond [2000]), it is chiefly discrete articulatory features that have been put to such use. We will consider three broad classes of models: tandem systems, DBN systems and CRF systems.

In Chapter 4, we present work on incorporating articulatory k-NN features in a tandem system for phone recognition and in a CRF for forced transcription of AFs. Similar k-NN features could fit into most, if not all, of the systems presented in the following.

Models based on MLPs and HMMs

Articulatory information has been used to augment HMM recognizers in several different systems. One of these is that of **Çetin et al.** [**2007**], who used the MLP-based AF classifiers from the JHU 2006 workshop Frankel et al. [2007a], described above, to build an articulatory tandem system. One MLP was trained for each of the eight AFs and their outputs were concatenated and log-transformed. A PCA transformation was applied, both to reduce the dimensionality (from 64 to 26) and also to decorrelate the features, and thus attempt to eliminate the redundancies in the features which may have been caused by their AFs not being independent of one another. The resulting vectors were concatenated with 39-dimensional PLPs, yielding a 65-dimensional observation vector. Word recognition experiments were conducted on a small corpus of telephone conversations with a 500-word vocabulary. The articulatory tandem system improved slightly upon a baseline phonetic tandem system, lowering the word error rate from 63% to 62.3%. Further gains were obtained by training the MLPs for AF classification on a larger data set; and when a factored observation modeling was added, so that the HMM modeled the PLPs and the tandem features separately, a word error rate of 59.1% was achieved.

Siniscalchi et al. [**2007; 2013b**] combine neural networks with HMMs in a different way. Siniscalchi et al. [2007] outlines a three-block architecture for phone recognition, with a set of binary AF classifiers followed by an *event merger* that combines the frame-level AF posteriors to phone posteriors, before an *evidence verifier* decodes the phone sequence. As mentioned in Section 2.5.1, the AF classification was done with one MLP for each AF that had a hidden layer with 500 nodes. The event merger was implemented as a single MLP with 800 nodes in the one hidden layer. The evidence verifier was a decoding network consisting of HMMs modeling context-independent phones with three states. A phone error rate of 27.7% was achieved on the TIMIT corpus of continuous read speech with a set of 39 phones. In a later paper by the same authors [Siniscalchi et al., 2013b], the architecture was modified using hierarchical MLPs and weighted finite-state machines (WFSMs) [Mohri et al., 2002]. We saw in Section 2.5.1 how the AF classification was done with hierarchical MLPs in this work; the posterior probabilities for each binary AF output by these classifiers were then concatenated with an MFCC vector that was input to an MLP for phone classification (corresponding to the event merger in [Siniscalchi et al., 2007]). The resulting phone posteriors were then combined with phone posteriors from a regular MLP phone classifier and input to the evidence verifier which was implemented with WFSMs. The resulting word error rate was 6% on a corpus of continuous read speech from the Wall Street Journal with a 5000-word vocabulary [Paul and Baker, 1992].

Kirchhoff et al. [2002] based a hybrid HMM system for word recognition on the MLP classification of five multi-valued AFs mentioned in Section 2.5.1. They input the posterior probabilities output by their five MLPs for AF classification into a higher-level MLP for phone classification which was used in the hybrid HMM. The baseline was a hybrid HMM with a single MLP phone classifier using acoustic observation vectors as its input. The sum of cardinalities for the AF set, and thus the number of posterior probabilities estimated by their MLPs, was 28. To reduce this number to match the dimensionality of the acoustic observation vectors used by their baseline system, they eliminated the 10 posteriors that had the least impact on the phone classification on the higher-level MLP (measured by relative entropy between the phone class distributions). The articulatory system was on par with the baseline in clean conditions and showed a distinct advantage in noisy conditions. When both systems were combined by linearly merging the outputs of the MLPs for phone classification, the word error rates were significantly reduced in all conditions.

Metze [2007] combined his frame classification results for 68 binary AFs, mentioned in Section 2.5.1, with standard phone-based acoustic models in an HMM for word recognition. The acoustic models for the AFs were integrated in the HMM with those of the phones using a multi-stream model [Bourlard et al., 1996]. Substantial improvements over a standard GMM-HMM system were attained on a corpus of spontaneous conversations containing a large portion of partial words and other spontaneous effects as well as a high proportion of words in a language foreign to the speakers. These improvements can be taken as an indication that this ASR

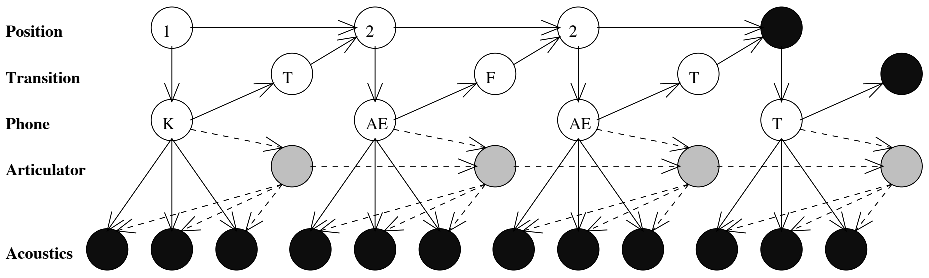


Figure 2.10: A dynamic Bayesian network for isolated word recognition [Stephenson et al., 2000] modeling the word *cat* in four time steps.

system allowed the recognizer to adapt to the articulatory characteristics of individual speakers or speaking styles better than the baseline system.

Dynamic Bayesian Network Models

We have previously seen how Frankel et al. [2004; 2007b] used dynamic Bayesian networks for classification and recognition of articulatory features (cf. page 40). The use of DBNs for speech recognition was pioneered by Zweig [1998], but his experiments with models that included articulatory modeling were limited. **Stephenson et al. [2000]**, though, presented a DBN model for isolated word recognition which made use of articulatory parameters. A database with time-aligned audio and and the position of gold pellets placed on the articulators measured with X-ray microbeams [Westbury, 1994] was used. The continuous articulatory measurements were quantized using K-means clustering—i.e., the vectors of articulatory parameter values were divided in K clusters such that each vector belonged to the cluster with the nearest mean vector—thus creating a single discrete articulatory feature. The best performance was achieved when the cardinality, K, was equal to 4.

Figure 2.10 shows the DBN model. The variable `Phone` gives the phone identity of the current frame and `Position` gives its index in the current word. `Transition` is a Boolean variable which equals 1 when the current frame is the last of the current phone and there is a transition to a new phone in the next frame. The `Articulator` variable gives the value of the single AF. The black nodes represent observed variables, which except for in the last frame is only `Acoustics`. The acoustic observation vectors (39-dimensional MFCCs) were also quantized using K-means, with $K = 256$. Thus, all variables in this DBN are discrete. When the `Articulator` variable is removed from the DBN in Figure 2.10, the model is equivalent to

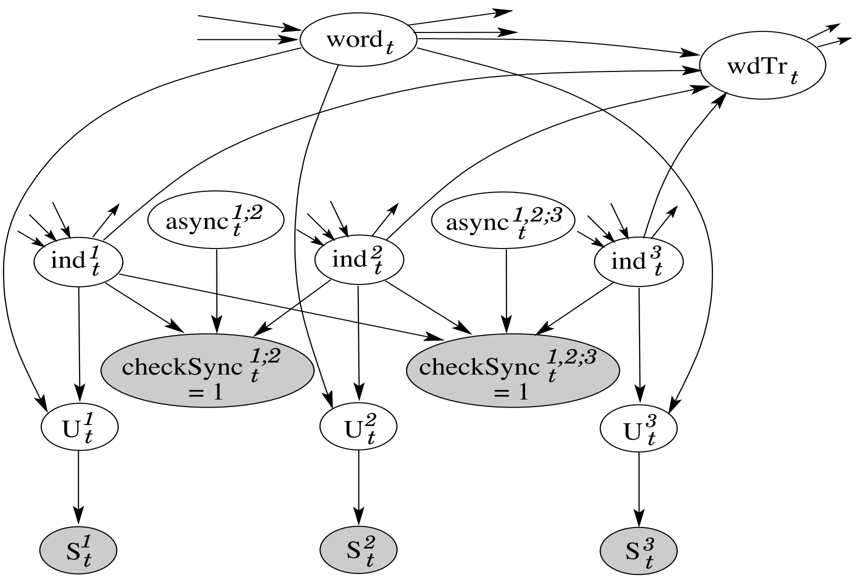


Figure 2.11: The pronunciation model DBN of Livescu and Glass [2004], with three articulatory features and two asynchrony constraints.

an HMM. This formed the baseline system, which the DBN with articulatory information improved upon with a 10% reduction in WER.

Note that although the DBN of Stephenson et al. does include some articulatory modeling, the modeling of pronunciation variation is very limited. Due to the fact that all the articulatory information is encoded in a single AF, there can be no asynchrony between them of the kind we used in Section 1.1 to explain the *don't* → `[d ow_n t]` variation (cf. Figure 1.1). **Livescu and Glass [2004]** presented a more sophisticated dynamic Bayesian network for modeling pronunciation variation based on articulatory features. They used a multi-valued AF set with eight features that corresponds closely to the original tract variables defined by Browman and Goldstein [Browman and Goldstein, 1992]. It has a close relationship to physical measurements, but remains discrete and subject independent. This is the articulatory feature set we will use for our experiments in the following chapter, and it is presented in detail in Section 3.1.

The pronunciation model DBN of Livescu and Glass [2004] is shown for one frame in Figure 2.11. For simplicity, only three articulatory features are included; these are denoted simply as `1`, `2` and `3`. The variables S_t^j, where $j \in \{1, 2, 3\}$, denote the surface values of these AFs in frame t; the variables U_t^j denote their underlying values. The AFs have separate indices, $\texttt{ind}_t^j$, that track the progression of their values through the current word. The underlying value of the corresponding AF, U_t^j, is deterministically determined by $\texttt{ind}_t^j$ along with the current word, $\texttt{word}_t$. The surface value, S_t^j, can deviate from U_t^j, and by learning the distributions $p(S_t^j \mid U_t^j)$, one can learn the probabilities of different substitutions. Asynchrony between the AFs is allowed—e.g., the AF 1 can progress faster through its values for the current word than the AF 2. The $\texttt{async}_t^{A;B}$ and $\texttt{checkSync}_t^{A;B}$ variables enforce constraints on this asynchrony.

This pronunciation model was used on a lexical access task on a corpus of telephone conversations. An end-to-end recognizer could in principle be built from this system, by adding acoustic observation variables as children of the S_t^j variables (which would then be hidden), but to facilitate quick experimentation and to isolate the performance of the pronunciation model, these surface AF values were considered as observed variables. The lexical access task was to determine which word is being spoken based on the surface AF values. A 40% relative reduction in word error rate was reported compared to a system using a large set of phonological rules.

This approach is described in more detail in Livescu [2005], and has been developed further at the JHU 2006 workshop [Livescu et al., 2007a,b] and by Jyothi et al. [2011].

Conditional Random Field Models

With increased attention around the benefits of discriminative models, CRFs are growing increasingly popular in speech technology applications [Fosler-Lussier et al., 2013]. We have already seen how Demuynck et al. [2011b] incorporated meta-information from a k-NN classifier in a CRF for word recognition (cf. page 32); here we will consider previous work on using CRFs for articulatory modeling of speech.

Morris and Fosler-Lussier [2006; 2008] built a CRF system for phone recognition with input features based on frame classification of AFs. Their AF set had eight multi-valued features based on the attributes of the International Phonetic Alphabet,[11] and a separate MLP classifier was trained for each of these. Rather than feeding the posterior probabilities of all AF values for all frames to the system, like one would in a tandem system, Morris and Fosler-Lussier made one feature for each possible pairing of AF values and phone labels using feature functions.[12] For example, the

[11]The articulatory features (and cardinalities) were: *sonority* (5), *voice* (3), *manner* (8), *place* (9), *height* (6), *front* (5), *round* (5), and *tense* (3).

[12]This kind of features is commonly used in CRFs when the labels are discrete—cf. Sutton [2011, pp. 293–4].

following feature function linked the phone label [b] to the output of the MLP classifier for the AF for voicing:

$$f_{\texttt{[b]},\text{voiced}}(\mathbf{O}, \mathbf{y}, t) = \begin{cases} \text{MLP}_{\text{VOICE=voiced}}(\mathbf{o}_t), & \text{if } y_t = \texttt{[b]} \\ 0, & \text{otherwise,} \end{cases} \tag{2.28}$$

where $\mathbf{O} = (\mathbf{o}_1, ..., \mathbf{o}_n)$ is the sequence of acoustic observation vectors; $\mathbf{y} = y_1, ..., y_n$ is the sequence of labels; and $\text{MLP}_{\text{VOICE=voiced}}(\mathbf{o}_t)$ designates the posterior probability of voicing in frame t as estimated by the AF classifier. By learning weights for these features, the CRF could assign importance to each of the AF values as indicators of each of the phones. In considering all possible pairings of AF values with phone labels, rather just the canonical AF values for each phone, the learning of negative weights was also encouraged (e.g., for $f_{\texttt{[p]},\text{voiced}}$, as [p] is an unvoiced consonant). The CRF compared favorably to a tandem HMM system with similar input features.

The experiments of **Chen and Wang** [**2009**] with a CRF for phone recognition indicate that AF asynchrony might be a bigger challenge for articulatory ASR systems than imperfect AF classification. They used recurrent neural networks to classify a set of eleven binary AFs based on Government Phonology [Harris, 1994] for the TIMIT corpus of read speech. The neural network outputs were thresholded, yielding binary classification results vectors. Compared to training the CRF directly on these classification results vectors (which included both AF asynchrony and classification errors) or on vectors of ground-truth AF labels[13] (which included neither), they found that training on forced-aligned AF labels (which included AF asynchrony but no classification errors) gave a better performance.

Chen and Wang also tried to isolate the effect of AF asynchrony by comparing the performance of the CRF on the test set when ground-truth AF labels were used as input and when forced-aligned AF labels were used. Similarly, to isolate the effect of the AF classification errors, they considered the CRF's test set performance when forced-aligned AF labels were used compared to when classification results vectors were used. The drop in phone accuracy was far bigger (28% relative) in the former case than in the latter (11%), so the AF asynchrony seemed to be considerably more problematic for their CRF system than the AF classification errors. It should be noted, however, that an articulatory word pronunciation model that bypasses phones (like that of Livescu and Glass, shown in Figure 2.11) might not find AF asynchrony to be a challenge in the same way that Chen and Wang found it to be for phone recognition.

[13]The ground-truth AF labels were derived from the phonetic labeling.

Another consequence of the findings of Chen and Wang is that phone-derived AF labels are likely to be unreliable close to phone boundaries. **Prabhavalkar et al.** [**2011**] addressed this problem through a CRF for *forced transcription* of articulatory features. Unlike in *forced alignment*, where one knows the sequence of labels and only needs to find the start and end times of each, in forced transcription you allow for multiple possible pronunciations. Thus, the task was to obtain an articulatory labeling given the acoustics and the word labeling. This task is separated from that of word recognition in that the words are treated as observed rather than hidden variables; and the AF labels, which would otherwise have been obtained as an intermediate step, are here the main goal. It should be noted that this model, unlike those of Morris and Fosler-Lussier or Chen and Wang, did not use phones in its modeling of pronunciation; in this sense it is arguably one step further removed from the "beads-on-a-string" model of pronunciation criticized by Ostendorf [1999]. We have chosen to use this system as a back-end for our features based on k-NN frame classification. Therefore, we defer further presentation of it to Sections 4.2.1 and 4.2.2.

2.6 Statistical Considerations

In this section we will consider the evaluation of speech recognition systems, and how to assess the statistical significance of differences between systems. We will begin by describing the evaluation metrics used in this thesis, before we go on to consider statistical tests for frame classification and continuous speech recognition. In closing, we will deduce a confidence interval for frame classification error rates.

2.6.1 Evaluation

For frame classification, evaluation is relatively straightforward. The classification of each frame is an independent event, and is either correct or incorrect. In this case, the error rate

$$\hat{e} = \frac{M}{n}, \tag{2.29}$$

where M is the number of errors and n is the total number of frames, is a maximum likelihood estimator of the probability of misclassifying a frame, e.

For continuous speech recognition tasks, the most popular performance measures are word error rate (WER) or phone error rate (PER), depending on whether the recognition unit is words or phones. Both of these measures can be defined as

$$W = \frac{\sum_i D_i}{\sum_i n_i}, \tag{2.30}$$

where n_i is the number of words/phones in sentence i and D_i is the edit distance between the reference transcription of sentence i and the recognizer output for this sentence. The edit distance, or Levenshtein distance, is equal to the minimum number of insertions, substitutions and deletions necessary to transform the one sentence into the other; it can be computed efficiently using dynamic programming algorithms. For WER, the sentence is considered as a string of words; for PER, as a string of phones.

WER and PER are rates representing the number of errors per spoken word/phone. Because of the possibility of insertions, they can exceed 100%. Thus, they cannot be regarded as probability estimates, unlike the classification error rate, and do not lend themselves as readily to statistical analysis.

2.6.2 McNemar's Test

In assessing the effectiveness of a statistical test comparing the performance of two systems for classification or recognition, there are particularly three probabilities that are of interest. A *Type I error* involves incorrectly detecting a difference between two systems when no difference exists. Minimizing the probability of such an error is usually the main task of a statistical test. A *Type II error* involves failing to detect a difference between two systems that does exist. This is typically considered less serious than a Type I error, but is still something to be avoided, so we want the probability of a Type II test to be small as well. The *power* of a statistical test is given by $1 - P(\text{Type II error})$.

For frame classification there are several available tests for statistical significance. Dietterich [1998] presented a thorough comparison of five possible choices, using simulation experiments to assess their performance on three data sets. A simple *difference-of-proportions test*, based on measuring the difference between the error rates of two classifiers on the same test set, had very incorrect independence assumptions and was shown to fail in cases where the error rates are close to 50%. The *resampled paired t test*, which has been a very popular choice in the machine learning literature, was found severely lacking in having an unacceptably high probability of Type I error. A *10-fold cross-validated paired t test* also had a relatively high probability of Type I error, but it also had a high power, and can hence be recommended for cases where avoiding Type II errors is particularly important (which is not the case for our experiments). The fourth test in Dietterich's comparison—*McNemar's test*—was shown to have a low probability of Type I error and also relatively high power. A test proposed by Dietterich, the *5×2cv paired t test*, is a variant of the cross-validated paired t test involving 5 repetitions of 2-fold cross-validation. It was shown to have slightly higher power than McNemar's test in some cases, but also a slightly higher probability of Type I error. In addition, it requires far more computation and limits the training set size to half of the available data.

McNemar's test thus seems like the best choice for our task. We go on to present this test in detail.

Let two systems for classification, S_1 and S_2, classify the same set of n labeled data points, yielding error rates $\hat{e}_1 = \frac{M_1}{n}$ and $\hat{e}_2 = \frac{M_2}{n}$, respectively. If the errors are independent events, the random variables M_j, denoting the numbers of errors made by system j, follow binomial distributions $\mathcal{B}(n, e_j)$. e_j denotes the probability that S_j will misclassify a data point, so to assess whether the difference between S_1 and S_2 is statistically significant, we want to test the hypothesis that $e_1 = e_2$.

If we could assume that M_1 and M_2 were independent variables, we could test this hypothesis using only the observed numbers of errors m_1 and m_2 along with the number of data points, n. This, however, will not be the case when systems S_1 and S_2 are tested on the same data set, because if they are at all similar, they are bound to have errors in common. We could try to estimate the correlation between $\hat{e}_1$ and $\hat{e}_2$, but this would require using up some of the n data points, rendering $\hat{e}_1$ and $\hat{e}_2$ as less reliable estimators of the true probabilities of misclassification, e_1 and e_2, by increasing their variances.

McNemar's test [Gillick and Cox, 1989; McNemar, 1947] offers a solution to this problem by considering only the K data points where the classification decisions of the two systems differ. The joint performance of systems S_1 and S_2 can be summarized in a contingency table as follows:

		S_2	
		Correct	**Incorrect**
S_1	**Correct**	A	B
	Incorrect	C	D

where:

$$
\begin{aligned}
A &= \text{the number of data points classified correctly by both } S_1 \text{ and } S_2,\\
B &= \text{the number of data points classified correctly by } S_1 \text{ and incorrectly by } S_2,\\
C &= \text{the number of data points classified incorrectly by } S_1 \text{ and correctly by } S_2,\\
D &= \text{the number of data points classified incorrectly by both } S_1 \text{ and } S_2.
\end{aligned}
$$

Note that $n = A + B + C + D$ and that $K = B + C$. If we now define $e_A = P(\text{a data point is classified correctly by both } S_1 \text{ and } S_2)$ and, analogously, e_B, e_C and e_D, we can see that $e_1 = e_C + e_D$ and $e_2 = e_B + e_D$. Thus, our null hypothesis, $e_1 = e_2$, is equivalent to $e_C = e_B$.

We go on to define

$$e_{C|K} = \frac{e_C}{e_B + e_C}, \tag{2.31}$$

which represents the conditional probability that S_1 will misclassify a data point given that it is misclassified by either S_1 or S_2. This gives us another equivalent formulation of the null hypothesis: $e_{C|K} = \frac{1}{2}$.

If we condition on $K = k$, still assuming that the errors are independent events, then C follows a binomial distribution: $C \sim \mathcal{B}(k, e_{C|K})$. After observing the values taken by C and D, we can then compute the probability of our null hypothesis. This can be done using the binomial distribution directly or, more conveniently, if $k > 50$ and C is not too close to k or 0, a normal approximation to the binomial may be used. C, then, should be approximately normally distributed: $C \stackrel{\cdot}{\sim} N(\mu, \sigma^2)$. We know that $\mu = E(C) = k \cdot e_{C|K}$ and that $\sigma^2 = \mathrm{Var}(C) = k \cdot e^2_{C|K}$. Thus, under the null hypothesis, $C \stackrel{\cdot}{\sim} N(\frac{k}{2}, \frac{k}{4})$ and

$$W = \frac{|C - \frac{k}{2}| - \frac{1}{2}}{\sqrt{\frac{k}{4}}} \stackrel{\cdot}{\sim} N(0, 1), \tag{2.32}$$

where the $-\frac{1}{2}$ in the numerator is a continuity correction factor. This is the formulation of McNemar's test given by Gillick and Cox [1989]. McNemar's original test [McNemar, 1947] with Edwards's correction for continuity [Edwards, 1948] is given as

$$W^* = \frac{(|B - C| - 1)^2}{B + C} \stackrel{\cdot}{\sim} \chi^2(1). \tag{2.33}$$

We can simplify (2.32) by substituting $k = B + C$:

$$W = \frac{|\frac{C-B}{2}| - \frac{1}{2}}{\sqrt{\frac{B+C}{4}}} = \frac{|B - C| - 1}{\sqrt{B + C}} \stackrel{\cdot}{\sim} N(0, 1). \tag{2.34}$$

Since we know that the square of a standard normal random variable follows a chi-squared distribution with one degree of freedom, we see that (2.32) and (2.33) are equivalent.[14] We compute the p-value as $p = 2P(Z \geqslant w)$, where $Z \sim N(0,1)$ and w is the realized value of the test statistic W in (2.34). The p-value equals the probability of the observed data (or data showing a more extreme departure from the null hypothesis) when the null hypothesis is true; i.e., in this case: the probability of S_1 and S_2 performing at least as differently as they do if one is no better than the other. We reject the null hypothesis if p is below our chosen significance level (e.g., 5%), in which case we conclude that there is a statistically significant difference between systems S_1 and S_2.

It should be noted that the p-value given by McNemar's test says nothing of *how different* the performance of the two systems is; it only addresses how probable it is that there is *some* difference between them, not how big of a difference this is.

2.6.3 The Matched Pairs Sentence-Segment Word Error Test

Because of language modeling, an error made by a continuous speech recognizer can influence the recognition of the rest of the utterance. Thus, the errors are not independent events in this case. To apply McNemar's test, we need to divide the speech recognizer output into segments in such a way that the errors in one segment are statistically independent of the errors in any other segment. A sentence is one natural candidate for such a segment. Another way of dividing the output into segments is to require that each segment where one or more errors occur (a "bad" segment), must be bounded by two segments where no errors occur for some minimal number of recognition units T ("good" segments). For a word recognition system with a trigram language model, setting T equal to two words is sufficient. The segments for two different systems can easily be aligned with a dynamic programming algorithm [Hunt, 1988]. This method will lead to a statistical test with higher power than using sentences since it yields more segments.

We can then run McNemar's test, as described in the previous section, on the resulting segments. We only need to consider the "bad" segments, and each segment must be considered as either correctly or incorrectly recognized. However, if we have two systems where one makes twice as many errors as the other, but these errors are spread across the same number of segments, this test will not be able to distinguish between these two systems.

[14] In the rare case that $B = C$, these two tests will give slightly different p-values because of the continuity correction factor. However, both p-values will be so close to 1 in this case that it is of no consequence.

The Matched Pairs Sentence-Segment Word Error (MAPSSWE) Test suggested by Gillick and Cox [1989] considers also the number of errors in each segment. For two speech recognition systems S_1 and S_2, let $M_1^{(i)}$ be the number of errors made by S_1 on the ith segment and $M_2^{(i)}$ be the number of errors made by S_2. Let $R_i = M_1^{(i)} - M_2^{(i)}$ be the difference in number of errors on segment i. Determining whether the performance of S_1 is equal to that of S_2 amounts to determining whether $\mu_R = E(R_i) = 0$. Let this be our null hypothesis.

We estimate μ_R by

$$\hat{\mu}_R = \bar{R} = \frac{1}{n}\sum_{i=1}^{n} R_i. \tag{2.35}$$

Since the $\hat{\mu}_R$ is the arithmetic mean of n random variables, we know that it is approximately normally distributed for sufficiently large values of n.[15] We can estimate its variance as

$$\mathrm{Var}(\hat{\mu}_R) = \frac{1}{n}\mathrm{Var}(R_i) \approx \frac{1}{n}S_R^2 = \frac{1}{n(n-1)}\sum_{i=1}^{n}(R_i - \bar{R})^2, \tag{2.36}$$

where S_R^2 is the sample variance of the R_is. Under the null hypothesis, $E(\hat{\mu}_R) = \mu_R = 0$, and thus the following test statistic should be approximately standard normal:

$$V = \frac{\hat{\mu}_R}{\sqrt{\mathrm{Var}(\hat{\mu}_R)}} = \sqrt{n} \cdot \frac{\hat{\mu}_R}{S_R} \mathrel{\dot{\sim}} N(0,1). \tag{2.37}$$

The p-value is computed in the same manner as above: $p = 2P(Z \geqslant v)$, where $Z \sim N(0,1)$ and v is the realized value of the test statistic V in (2.37). We reject our null hypothesis if p is below our chosen significance level.

2.6.4 Confidence Intervals

When errors are independent events, as is the case in frame classification, we can in addition to finding an estimate $\hat{e}$ for the error rate by Equation (2.29), find a confidence interval for the true error rate, e—i.e., an interval within which e will be found with a given probability.

[15]The normal approximation will be good for $n \geqslant 30$ as long as the distribution of the random variables is not exceptionally skewed [Walpole et al., 2012, p. 234]. The presumption of normality becomes more accurate as n grows larger.

Under the assumption of independent events, the number of errors will follow a binomial distribution: $M \sim \mathcal{B}(n, e)$. As before, we can use the normal approximation to the binomial when $n > 50$ and M is not too close to 0 or n,[16] so that

$$M \stackrel{\cdot}{\sim} N\big(E(M), \mathrm{Var}(M)\big) = N\big(ne, ne(1-e)\big). \tag{2.38}$$

Thus, we can construct a statistic, U, that is approximately standard normal:

$$U = \frac{M - ne}{\sqrt{ne(1-e)}} \stackrel{\cdot}{\sim} N(0,1). \tag{2.39}$$

Hence, we get the confidence interval

$$P\left(-z_{\alpha/2} - \frac{1}{2} < \frac{M - ne}{\sqrt{ne(1-e)}} < z_{\alpha/2} + \frac{1}{2}\right) \approx 1 - \alpha, \tag{2.40}$$

where α is our chosen significance level, $z_{\alpha/2}$ is the critical value of the standard normal distribution such that $P(Z > z_{\alpha/2}) = \frac{\alpha}{2}$ when $Z \sim N(0,1)$, and the added and subtracted halves are continuity correction factors.

We want to solve (2.40) to find functions $f_1(M)$ and $f_2(M)$ such that

$$P\big(f_1(M) < e < f_2(M)\big) \approx 1 - \alpha. \tag{2.41}$$

Harborg [1990] gives the solutions to this equation as

$$f_1(M) = \frac{M - \frac{1}{2} + \frac{1}{2}z_{\alpha/2}^2 - z_{\alpha/2}\sqrt{\frac{1}{4}z_{\alpha/2}^2 + \frac{1}{n}(M - \frac{1}{2})(n - M + \frac{1}{2})}}{n + z_{\alpha/2}^2}, \tag{2.42}$$

and

$$f_2(M) = \frac{M + \frac{1}{2} + \frac{1}{2}z_{\alpha/2}^2 + z_{\alpha/2}\sqrt{\frac{1}{4}z_{\alpha/2}^2 + \frac{1}{n}(M + \frac{1}{2})(n - M - \frac{1}{2})}}{n + z_{\alpha/2}^2}. \tag{2.43}$$

For fixed values of n and α, Equations (2.41), (2.42) and (2.43) give us a confidence interval which depends only on the number of errors, M.

In this way, confidence intervals can be computed for frame classification error rates. This method can not be used to compute confidence intervals

[16] A more precise requirement, involving the unknown parameter e, is that we should have both $ne > 5$ and $n(1-e) > 5$ for the normal approximation to the binomial to be valid.

for phone or word error rates, however, since errors are typically not independent events in continuous speech recognition tasks.

Bisani and Ney [2004] propose a *bootstrap* method for computing a confidence interval that will be valid for WER and PER. Bootstrapping [Efron and Tibshirani, 1993] is a statistical resampling method based on creating replications of a statistic by random sampling from the data set with replacement. Bisani and Ney divide the test set into segments such that the errors in one segment are independent from the errors in all the others (typically this will be a sentence, but in some cases other units may be more suitable—e.g., the set of all utterances of one speaker in the case of speaker-dependent ASR). They then randomly select s sentences, with replacement, where s is the number of sentences in the test set, thus generating a bootstrap sample. By computing the WER for a large number of these bootstrap samples, a confidence interval for the true error rate can be estimated. A confidence interval for the difference in error rates between two systems can be found by computing the difference in error rate for the two systems on a number of identical bootstrap samples. These bootstrapped confidence intervals are computationally costly to produce; and since the focus of thesis is not so much the error rates themselves but rather how different systems compare to each other, we will not use them here.

Chapter 3

Frame Classification by Nearest Neighbors

In this chapter we address the problem of classifying articulatory features (AFs) from audio. Given an observation vector representing 10 ms of the speech signal, we try to assign a value to each of the features listed in Table 3.1, effectively performing eight separate classification tasks. This might be seen as an instance of acoustic-articulatory inversion, in that we try to infer the articulatory configuration of the speaker based only on the acoustic signal. Inferring the articulatory configuration from speech audio is of interest as a scientific question in and of itself, but our main interest in classifying these AFs is as an intermediate step in an articulatory speech recognition system.

The classification of articulatory phonology-based features from acoustics has not been extensively studied. In this chapter, we study the problem of classifying the feature values in a frame of speech, both jointly and independently, using k-nearest neighbor (k-NN) classifiers and multilayer perceptrons (MLPs). MLPs are commonly used for phonetic classification and have also been used for classification of several sets of AFs (as we saw in Section 2.5.1), but to our knowledge not for articulatory phonology-based features. Although our main focus is the classification of AFs, we also do phonetic frame classification of the same data, in order to give an idea of how our methods perform on a more familiar task.[1]

[1]It should be noted that the task of *frame classification of phones* is different from *phone recognition*, which we shall return to in Chapter 4. In frame classification, the decision is based solely on the current observation vector.

Table 3.1: Articulatory Phonology-Based Feature Set

Feature	Name	Values
Lip constriction location	LIP-LOC	protruded (0), labial (1), dental (2)
Lip opening degree	LIP-OPEN	closed (0), critical (1), narrow (2), wide (3)
Tongue tip constriction location	TT-LOC	inter-dental (0), alveolar (1), palato-alveolar (2), retroflex (3)
Tongue tip opening degree	TT-OPEN	closed (0), critical (1), narrow (2), mid-narrow (3), mid (4), wide (5)
Tongue body constriction location	TB-LOC	palatal (0), velar (1), uvular (2), pharyngeal (3)
Tongue body opening degree	TB-OPEN	closed (0), critical (1), narrow (2), mid-narrow (3), mid (4), wide (5)
Velum opening degree	VEL	closed (0), open (1)
Glottis opening degree	GLOT	closed (0), critical (1), wide (2)

3.1 The Articulatory Feature Set

Table 3.1 shows the set of articulatory features introduced by Livescu [2005] and used for the experiments in this thesis. These AFs are based on the vocal tract variables of articulatory phonology [Browman and Goldstein, 1992], which we described at the beginning of Section 2.5, but unlike the vocal tract variables, our AFs are discrete rather than continuous. They refer to the locations and degrees of constriction of the major articulators in the vocal tract, as illustrated in Figure 3.1. This articulatory feature set has been proposed as alternative subword units (as opposed to phones) for automatic speech recognition in the work of Deng et al. [1997], Livescu [2005] and Hu et al. [2010].

As we saw in Section 2.5.1, several different AF sets have been used in the literature, both binary and multi valued. The feature set we consider here has certain appealing qualities compared to others, such as greater independence of the individual features and a more direct correspondence with measurable vocal tract properties. It is also able to explain many types of pronunciation variation that are hard to account for with AF sets based on the categories used in the International Phonetic Alphabet (IPA) [Albright, 1958], which are commonly used (see, e.g., Metze [2005]; Bromberg et al. [2007]; Morris and Fosler-Lussier [2008]). For the pronunciation variant *don't* → `[d ow_n t]`, which we considered in the introduction (cf. Figure 1.1), IPA-style features can easily account for the nasalization of the vowel, but not the `[n]` deletion. As we saw in our discussion on page 3, the `[n]` deletion can be explained by asynchrony between GLOT and VEL on the one hand and the tongue features on the other; IPA-style features, being strongly tied to a phonetic representation, are usually unable to model this form of asynchrony. Similarly, stop insertion (e.g., *sense* → `[s eh n t s]`) and reduction of consonants to glides or vowel-like sounds (e.g., `/b/` → `[w]`) are hard to explain with IPA-style features but easily accounted for with the feature set used here. In the lexical access experiments of Livescu [2005, Ch. 4], where a word is identified from its AF values, this feature set was more successful than one based on IPA categories.

The ground-truth values for each of the AFs are set by a deterministic mapping from the annotated phones, laid out in detail in Table A.4 in the appendix. The same mapping was used in the lexical access experiments of Livescu [2005]. Stops, affricates, and diphthongs are split into two segments each with separate AF configurations. In such cases, $^2/_3$ of the annotated segment's duration is assigned to the first configuration and the latter $^1/_3$ to the second. Mapping from phones to AFs is not optimal, but we choose this option due to the lack of corpora annotated at the articulatory feature level.

3.2 Data Description

The experiments presented in this chapter are done on a portion of the Switchboard Transcription Project (STP) data [Greenberg, 1996; Greenberg et al., 1996]. This is a subset of the Switchboard Telephone Speech corpus [Godfrey et al., 1992], which consists of spontaneous telephone conversations between speakers of American English. The STP data has been manually transcribed at a detailed phonetic level, including diacritics indicating nasalization, frication (of an otherwise unfricated sound), glottalization ("creaky voice") and other phonetic properties. We use only the portion of STP that has been manually aligned at the phonetic level (the remainder of STP was manually aligned only at the syllable level). We choose this corpus because of its detailed labeling and conversational nature; we expect the biggest advantages of using articulatory models over phonetic models to be seen for conversational, spontaneous speech rather than more formal or read speech.

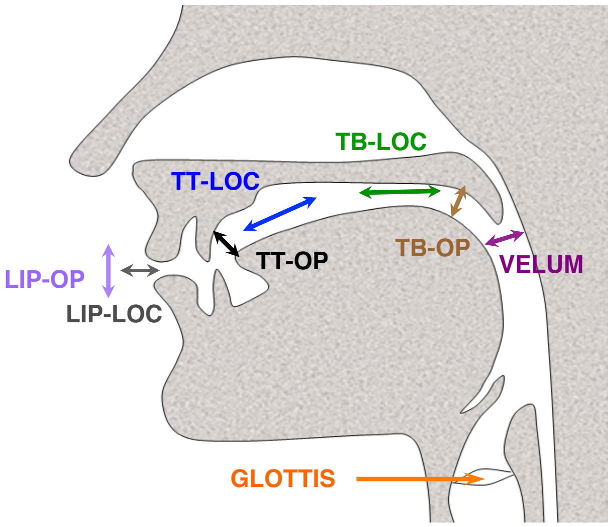


Figure 3.1: An illustration in the mid-sagittal plane of the articulatory feature set given in Table 3.1. (Figure from Livescu [2005].)

We follow Livescu [2005] in disregarding all diacritics except for nasalization, due to the lack of consistency between transcribers. This reduces the number of phone types in our corpus from 414 to 98.

The acoustic observation vectors consist of 12 mel-frequency cepstral coefficients (MFCCs) plus log energy concatenated with their first and second derivatives—thus having 39 dimensions. The MFCCs are computed for 20 ms windows with a 10 ms frame shift. They have been extracted with the freely available Rastamat Matlab toolbox [Ellis, 2005], using settings to reproduce the MFCC computation of the Hidden Markov Model Toolkit [Young et al., 2009; Ellis, 2013].

We have experimented with concatenations of multiple consecutive frames around the current frame to form longer-context feature vectors. Here we report on experiments using either single-frame windows or 9-frame windows. Considering that classifiers can struggle with inputs of too high dimensionality, and since the derivatives are usually included with MFCCs to capture context, we did not consider it useful to include these derivatives for all nine of the concatenated frames. The derivative is computed over a window of five frames (the current frame as well as the two preceding and the two succeeding ones), so we decided to keep the first and second derivatives of the first two and the last two of the nine concatenated frames, omitting them for the five frames in the middle. In other words, we omit the derivatives only when they are computed over a window of frames that is fully contained in the concatenated vector. This gives the 9-frame concatenated vectors a dimensionality of $39 \cdot 4 + 13 \cdot 5 = 221$.

The frames are assigned ground-truth phonetic labels based on the time stamps in the transcriptions. Silence frames are excluded, so that silence does not dominate the classification task by virtue of being by far the most frequent label. After excluding silence frames, our corpus consists of 219 251 frames, or 36.5 minutes, of speech. Note that, since silence is excluded, the number of minutes might seem misleadingly low.

All of our experiments are done on the training subset of the STP transcriptions. Due to the small size of this data we have chosen to follow a round-robin five-fold cross-validation scheme for all of our experiments. We divide our data in five parts, use three of them as a training set, one as a development set and one as a test set. We report the mean performance over the five test folds.

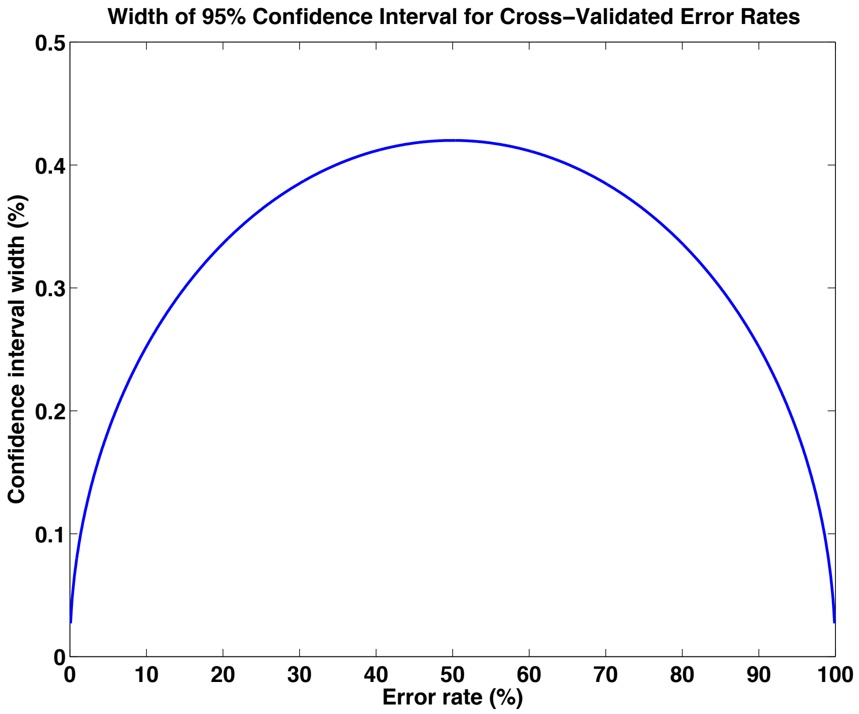


Figure 3.2: The width of a 95% confidence interval for the true probability of misclassification, p, is plotted against the magnitude of the error rate, $\hat{e}$, for the experiments in this chapter. The width ranges from 0.03% (for $\hat{e} = 0.1\%$ or $\hat{e} = 99.9\%$) to 0.42% (for $\hat{e} = 50\%$).

Confidence Intervals As noted in Section 2.6.4, for fixed values of the number of utterances and the confidence level α, the width of the $(1-\alpha)\%$ confidence interval for the error rate depends only on the number of errors. We can thus tell in advance what the confidence interval will be for any given error rate. Figure 3.2 shows how the width of a 95% confidence interval will vary with the error rate for our experiments. Because of our round-robin cross-validation setup, we are testing on all of the 219 251 frames. As the figure shows, this makes the confidence intervals very narrow for all error rates. The largest confidence interval occurs for an error rate of 50% and is no more than 0.42% wide—i.e., from 49.79% to 50.21%.[2]

3.3 Baselines

Our main baseline is a multilayer perceptron classifier. However, we have also included a chance classifier which is instructive for comparing the classification tasks.

Chance Classifier

The eight AFs differ greatly in how hard they are to classify. This is partly due to their different cardinalities (it is obviously easier to choose one out of two classes than one out of six), but also due to the differing distributions. For some of the AFs, one value is much more common than the others. The dominance of this one value can make some of the classification tasks somewhat easier than their cardinalities would suggest.

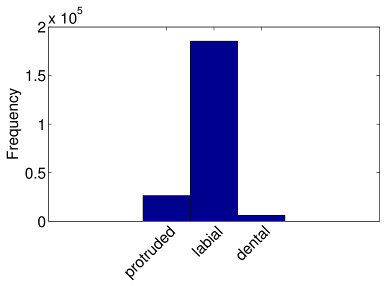
Our chance classifier always chooses this most common value, ignoring the observation vector. We notice from Table 3.2 that this always performs better than pure chance, in some cases markedly better, e.g., for LIP-OPEN. Figure 3.3 shows histograms of the feature values of the eight AFs across all frames in our data,[3] and we can see that LIP-OPEN is the AF where the most frequent feature value is the most dominant.

Multilayer Perceptrons

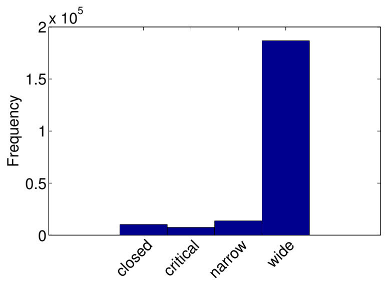
We trained the MLP classifiers using back-propagation with stochastic gradient descent, implemented in the QuickNet toolkit [Johnson et al., 2004]. The number of units in the hidden layer was tuned on a grid of 9 values

[2]For comparison, the division of the STP data used by Livescu [2005], with a 15 462-frame test set, would result in a confidence interval 1.58% wide in this case.

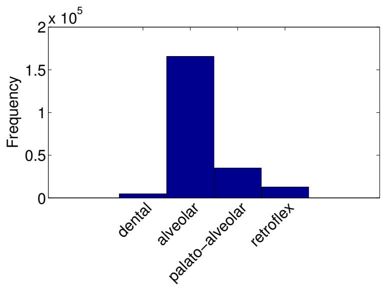
[3]The histogram for phone frequencies in the STP corpus is shown in Figure B.1 in the appendix.



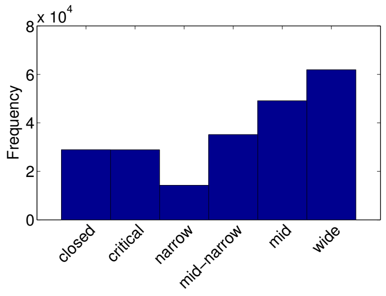
(a) LIP-LOC



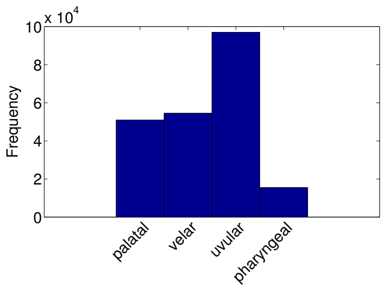
(b) LIP-OPEN



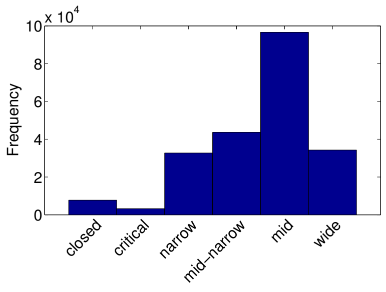
(c) TT-LOC



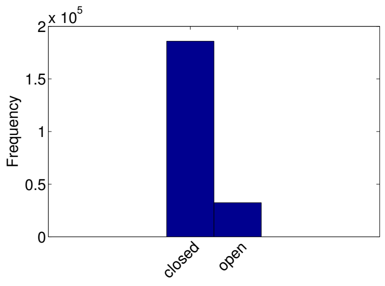
(d) TT-OPEN



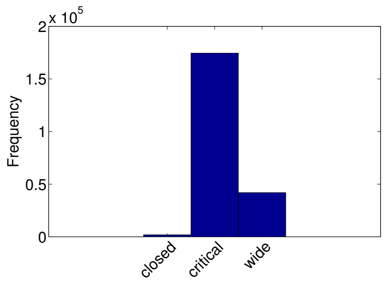
(e) TB-LOC



(f) TB-OPEN



(g) VEL



(h) GLOT

Figure 3.3: Histograms of the frequencies in our STP data set of each value of the eight articulatory features.

Table 3.2: Chance Classifier Performance
Error rates when always choosing the most common label, given in percentages.

	Cardinality	Error Rate (%)
AF Mean	-	33.9
LIP-LOC	3	15.1
LIP-OPEN	4	14.4
TT-LOC	4	24.2
TT-OPEN	6	71.7
TB-LOC	4	55.4
TB-OPEN	6	55.7
VEL	2	14.7
GLOT	3	20.3
Phones	98	95.3

from 250 to 10 000. Bunch size (i.e., the number of input observation vectors for which the error is accumulated before back-propagating to update the weights) was tuned on a grid from 8 to 128 in powers of 2. To avoid overfitting, we adopted an adaptive learning rate schedule and early stopping. The learning rate was tuned on a grid that went from 0.004 to 0.032 in powers of 2. A separate MLP was trained for each AF.

The performance of the MLP classifiers is given in Table 3.3. 9-frame vectors consistently outperform 1-frame vectors. We also see that the MLP is only marginally better than the chance classifier when the latter has an error rate under 30%, which is the case for five of the AFs, with the notable exception of GLOT. Figure 3.3 shows us that the AFs in question (LIP-LOC, LIP-OPEN, TT-LOC and VEL) all have one dominant value. For these AFs, the absolute improvements in error rate of the MLP over the chance classifier range from 1.27% to 3.54%, so the MLP classifier is unable to perform substantially better than picking the most frequent class for these particular tasks.

3.4 k-Nearest Neighbor Classification

Our k-NN classifiers are implemented with the LMNN Matlab toolkit [Weinberger, 2007]. This makes use of ball trees [Andoni and Indyk, 2006] to speed

Table 3.3: MLP Performance
Error rates (in %) for the multilayer perceptron classifier with 1-frame and 9-frame observation vectors, in percentages.

	1-frame	**9-frame**
AF Mean	24.1	22.6
LIP-LOC	14.3	13.8
LIP-OPEN	14.0	12.9
TT-LOC	21.9	20.8
TT-OPEN	44.5	41.6
TB-LOC	37.4	35.4
TB-OPEN	39.4	36.6
VEL	11.8	11.2
GLOT	9.5	8.6
Phones	67.1	63.9

up the neighbor search. Most of the computation time is taken by finding the neighbors; the actual classification takes very little CPU time. We use the standard Euclidean distance metric.

In our cross-validation setup (cf. Section 3.2) we classify the development fold with most or all odd numbers of nearest neighbors up to a maximum of 1001 (or lower if there is clearly no improvement in performance), and use the optimum of these to classify the test fold. Tie breaking (i.e., when two classes get the same number of votes from the nearest neighbors) is done by going to a lower k. We use only odd numbers to reduce the chance of ties.

Table 3.4 shows the results for standard k-NN classification, with observation vectors consisting either of a single frame of the speech signal or of nine concatenated frames. We see consistently better performance when using the concatenated 9-frame observation vectors; hence we see that the classifier benefits from including more context. The baseline MLP results have been included for ease of comparison, and we see that the k-NN classifier does worse. For the results with 9-frame observation vectors, McNemar's test shows that the differences between the k-NN and MLP results are significant for all the nine classification tasks ($p < 0.0005$).

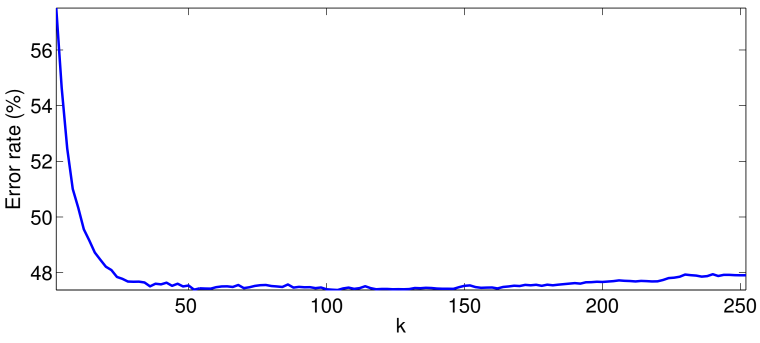
The optimal values of k ranged from 7 to 103 for the AF classifications and from 33 to 97 for the phone classifications. There were no clear differences between the optimal ks for 1-frame and 9-frame features, and the

Table 3.4: k-NN Performance
Error rates (in %) for the k-NN classifier with 1-frame and 9-frame observation vectors, in percentages. The results for the MLP baseline are included for ease of comparison.

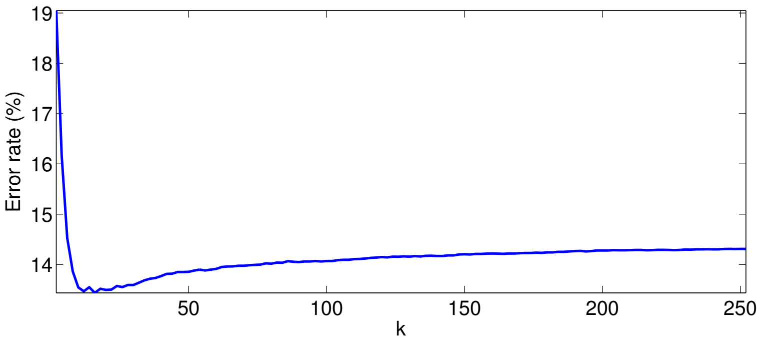
	1-frame		**9-frame**	
	k-NN	MLP	k-NN	MLP
AF Mean	28.6	*24.1*	27.5	*22.6*
LIP-LOC	14.6	*14.3*	14.1	*13.8*
LIP-OPEN	14.1	*14.0*	13.8	*12.9*
TT-LOC	23.3	*21.9*	22.2	*20.8*
TT-OPEN	55.7	*44.5*	53.0	*41.6*
TB-LOC	45.0	*37.4*	43.2	*35.4*
TB-OPEN	47.5	*39.4*	46.4	*36.6*
VEL	13.4	*11.8*	13.4	*11.2*
GLOT	15.1	*9.5*	14.3	*8.6*
Phones	78.6	*67.1*	76.5	*63.9*

behavior for increasing values of k was similar across the AFs. As shown in Figure 3.4, the error rate curves tend to reach an “elbow point” around $k = 30$. For fold 5 of the 1-frame classification of TB-OPEN shown in Figure 3.4a, which has $k = 103$ as its optimum, the change is small above $k = 30$, but the error keeps sinking up to $k = 103$. For fold 3 of the 9-frame classification of LIP-OPEN shown in Figure 3.4b, we see the most typical behavior: a small increase in error rate after an optimum around $k = 30$. For fold 3 of the 9-frame classification of TT-OPEN shown in Figure 3.4c, we see the same pattern, but a far steeper increase in the error rate after the optimum.

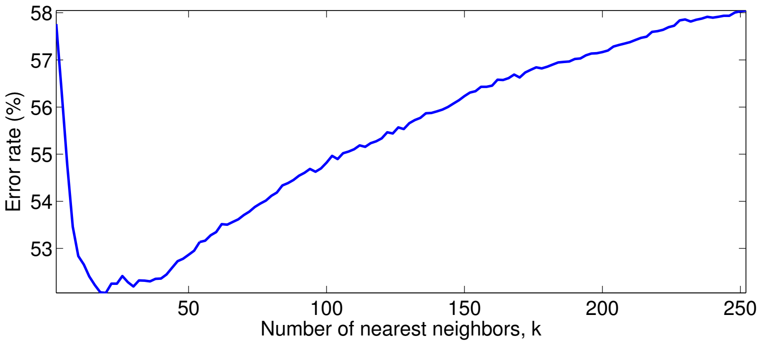
GLOT consistently has markedly lower optimum values of k than the other AFs. The mean error rate of the ten folds (five for 1-frame, five for 9-frame) is 11.2%; for the others the means range from 20.6% to 63.6%. It is not clear why this would be the case. GLOT does not have the lowest cardinality, nor is the chance classifier error rate for GLOT particularly low or high. We shall see, however, that GLOT tends to behave slightly differently from the other AFs in several respects.



(a) TB-OPEN, fold 5, 1-frame features



(b) LIP-OPEN, fold 3, 9-frame features



(c) TT-OPEN, fold 3, 9-frame features

Figure 3.4: Plots of the error rate against the number k of nearest neighbors used by the k-NN classifier for one fold of three different AFs.

Individual vs. Joint Classification

So far we have classified the eight AFs separately; another option is to classify them jointly, treating each configuration of all eight AFs as one class. The former approach is more appropriate under the assumption that there are no dependencies between the AFs, while the latter allows for arbitrary dependencies.

While the AFs are "physiologically" largely independent (there are few constraints on the state of one articulator given those of other articulators), it is reasonable to think there might be dependencies between them, in the sense of the value of one AF impacting the probability distributions of one or more of the other AFs. We classify the AFs jointly to examine this possibility.

For individual classification, the number of classes for each task equals the cardinality of the AF in question. For joint classification, the number of classes is higher. There are 41 472 possible such configurations (the product of the cardinalities of the eight AFs), but only 64 configurations occur in the data and we limit ourselves to this number of classes. We evaluate the joint classification in the same way as the individual classification, considering the error rate for each AF separately, by mapping the joint AF labels to individual AF labels.

Table 3.5 shows the results for joint classification, alongside the corresponding results for individual classification. We see that joint classification consistently hurts performance except for the classification of GLOT, which again is the outlier.

Normalization of Observation Vectors

The fact that the dimensions of the MFCCs vary in both range and mean may impact the Euclidean distance between the observation vectors. We therefore try a global normalization, giving all dimensions of the observation vectors zero mean and unit standard deviation. For each of the cross-validation's five partitions into training, development and test sets, we compute the mean vector of the three training folds, $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{o}_i$, and the vector of the sample standard deviations of each dimension, $\mathbf{s} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{o}_i - \hat{\boldsymbol{\mu}})^2}$. Then, $\hat{\boldsymbol{\mu}}$ is subtracted from all the observation vectors and the resulting vectors are divided, element-wise, by $\mathbf{s}$.

Table 3.6 shows the results for k-NN classification with the resulting normalized observation vectors. We see a substantial improvement over the results for non-normalized ("raw") observations. For 9-frame vectors, McNemar's test shows all the differences to be significant at the 0.001 level;

Table 3.5: Joint k-NN Performance
Error rates (in %) for the joint k-NN classifier. The results for the individual k-NN classifier are included for ease of comparison.

	1-frame		**9-frame**	
	Joint	*Individual*	Joint	*Individual*
AF Mean	31.3	*28.6*	30.1	*27.6*
LIP-LOC	16.6	*14.6*	16.5	*14.1*
LIP-OPEN	15.9	*14.1*	15.8	*13.8*
TT-LOC	26.7	*23.3*	24.6	*22.2*
TT-OPEN	56.9	*55.7*	54.1	*53.0*
TB-LOC	49.6	*45.0*	47.9	*43.2*
TB-OPEN	52.6	*47.5*	50.9	*46.4*
VEL	16.8	*13.4*	17.0	*13.4*
GLOT	15.0	*15.1*	13.9	*14.3*

for 1-frame vectors, the significance is as strong as it gets, with all the p-values below the numeric accuracy (i.e., $p < 10^{-15}$).

A result of this more marked improvement for 1-frame vectors is that concatenating frames does not help us in this case. While the concatenation slightly improves performance for three of the eight AFs (and for phones), it deteriorates performance for four of the others.

The impact of different ks is about the same in this case as in the non-normalized one. However, the range in optimal ks is higher for the 9-frame case here, from 7 to 189, versus 21 to 87 in the single-frame case.

Comparison of Different Acoustic Observation Vectors

As mentioned in Section 2.1, there are two prevalent ways of computing the acoustic observation vectors that make up the observation vectors representing the speech signal: mel-frequency cepstral coefficients (MFCCs) and perceptual linear predictive coefficients (PLPs). So far we have reported results with MFCCs; we will here make a comparison to the performance with PLPs.

Our PLP vectors are computed with the Rastamat Matlab toolbox [Ellis, 2005], using settings to reproduce the PLP computation of the `feacalc` tool released by the International Computer Science Institute [ICSI, 2012; Ellis, 2013]. The order of the PLP modeling is set to 12. We use 13 cepstral

Table 3.6: Normalized k-NN Performance
Error rates (in %) for the k-NN classifier with normalized observation vectors, in percentages. The results for "raw"—i.e., non-normalized—features are included for ease of comparison.

	1-frame		**9-frame**	
	Normalized	*Raw*	Normalized	*Raw*
AF Mean	24.3	*28.6*	24.5	*27.6*
LIP-LOC	14.0	*14.6*	13.9	*14.1*
LIP-OPEN	13.8	*14.1*	13.6	*13.8*
TT-LOC	21.4	*23.3*	21.4	*22.2*
TT-OPEN	45.9	*55.7*	45.7	*53.0*
TB-LOC	37.7	*45.0*	37.9	*43.2*
TB-OPEN	39.9	*47.5*	40.6	*46.4*
VEL	11.5	*13.4*	12.1	*13.4*
GLOT	10.2	*15.1*	10.3	*14.3*
Phones	70.7	*78.6*	70.6	*76.5*

coefficients plus their first and second derivatives, computed for 20 ms windows with a 10 ms frame shift, yielding 39-dimensional observation vectors. We have seen previously in this section that 9-frame vectors generally give better performance than single-frame vectors; for these experiments 9-frame vectors will be used. We omit the derivatives for the five middle frames for the PLPs like we did for the MFCCs, with a resulting dimensionality of 221. The computation of the MFCC vectors was described in Section 3.2.

Table 3.7 shows the k-NN frame classification error rates for MFCCs and PLPs, as well as the difference between them, when the observation vectors have not been normalized. We see that the PLP vectors give a better performance in all but one case, and that the differences are relatively substantial.

When the global normalization described above is applied to the observation vectors, however, the picture changes. Table 3.8 shows the results for this case, and we see that although the normalization improves the performances for both MFCCs and PLPs in all cases, the benefits for the MFCCs are higher in all but one case, usually by a factor of two or more. With normalized observation vectors, the k-NN error rates are lower when using MFCCs in the clear majority of cases. The differences, however, are mostly

Table 3.7: MFCCs v PLPs, Raw Observation Vectors
k-NN classification error rates (in %) for raw 9-frame acoustic observation vectors.

			Difference	
	MFCCs	**PLPs**	Absolute	Relative
AF Mean	27.55	26.20	1.35	4.90
LIP-LOC	14.10	13.97	0.13	0.91
LIP-OPEN	13.77	13.68	0.09	0.68
TT-LOC	22.28	22.52	−0.24	−1.08
TT-OPEN	53.09	50.28	2.81	5.30
TB-LOC	43.30	40.77	2.53	5.85
TB-OPEN	46.30	43.81	2.49	5.39
VEL	13.36	12.19	1.16	8.71
GLOT	14.23	12.41	1.82	12.80
Phones	76.52	73.19	3.33	4.35

insubstantial; we see that the relative changes are mostly well under 1%. Although McNemar's test shows the differences to be significant at the 0.005 level for five out of the eight AFs as well as for the phones, we do not think these differences are large enough for us to draw any conclusions as to one acoustic feature extraction being better than the other for this application.

The rest of the experiments in this chapter will use MFCCs.

3.5 Dimensionality Reductions

The basic assumption behind nearest neighbor classification schemes is that examples that are close in the feature space are likely to belong to the same class. How well this works depends on the distance metric, $\mathcal{D}$.

Typical choices for $\mathcal{D}$ are the Euclidean distance and its generalizations, the Mahalanobis distances. A Mahalanobis distance between two d-dimensional vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ is given by

$$\mathcal{D}_M(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^{\mathrm{T}} M (\mathbf{x}_1 - \mathbf{x}_2)}, \tag{3.1}$$

where M is a $d \times d$ symmetric positive-semidefinite matrix. If M equals the identity matrix, this reduces to the Euclidean distance. The most common

Table 3.8: MFCCs v PLPs, Normalized Observation Vectors
k-NN classification error rates (in %) for normalized 9-frame acoustic observation vectors.

			Difference	
	MFCCs	**PLPs**	Absolute	Relative
AF Mean	24.45	24.71	−0.25	−1.04
LIP-LOC	13.91	13.88	0.03	0.23
LIP-OPEN	13.61	13.66	−0.05	−0.34
TT-LOC	21.35	21.54	−0.18	−0.85
TT-OPEN	45.76	45.88	−0.12	−0.27
TB-LOC	37.93	38.10	−0.17	−0.45
TB-OPEN	40.64	41.11	−0.48	−1.17
VEL	12.08	11.81	0.27	2.25
GLOT	10.34	11.68	−1.34	−12.92
Phones	70.63	70.57	0.06	0.08

Mahalanobis distance lets M be the inverse covariance matrix of all the vectors in the training set

Linear feature transforms, such as principal components analysis (PCA) and linear discriminant analysis (LDA), also induce Mahalanobis distances. Consider a linear transform L from $\mathbb{R}^d$ to another (possibly lower-dimensional) space $\mathbb{R}^l$. The Euclidean distance between two vectors mapped to this space, $\mathcal{D}(L\mathbf{x}_1, L\mathbf{x}_2)$, is equivalent to a Mahalanobis distance with $M = L^{\mathrm{T}}L$ in the original space:

$$\begin{aligned}&(L\mathbf{x}_1 - L\mathbf{x}_2)^{\mathrm{T}}(L\mathbf{x}_1 - L\mathbf{x}_2)\\&= \big(L(\mathbf{x}_1 - \mathbf{x}_2)\big)^{\mathrm{T}}L(\mathbf{x}_1 - \mathbf{x}_2)\\&= (\mathbf{x}_1 - \mathbf{x}_2)^{\mathrm{T}}L^{\mathrm{T}}L(\mathbf{x}_1 - \mathbf{x}_2).\end{aligned} \tag{3.2}$$

PCA and LDA can thus be viewed as a form of distance learning, since estimating such a linear transform is equivalent to learning a Mahalanobis distance from the data. We would therefore expect these transformations to be helpful for improving the k-NN performance.

When the linear transformation matrix L is not square, but has fewer rows than columns, it works as a dimensionality reduction. We have seen that our k-NN classifier performs better with 9-frame than with single-frame windows. However, 221 is a fairly high number of dimensions. The “curse of dimensionality” is a well-known problem in machine learning in general [Bellman and Dreyfus, 1962; Friedman, 1997] and for k-NN classifiers in particular [Indyk and Motwani, 1998], so we would also expect that reducing the dimensionality of our observation vectors would be beneficial to performance.

In this section we will consider dimensionality reduction by PCA and LDA, as well as by a newer technique, Locality Preserving Projections.

3.5.1 Principal Component Analysis

Principal component analysis [Pearson, 1901; Jolliffe, 2002] computes the linear transformation L that projects the observation vectors in the training set into a variance-maximizing subspace. The sum of the variances of the observation vectors $\{\mathbf{x}_i\}_{i=1}^{n}$ of the training set is equal to the trace of their covariance matrix, C:

$$C = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathrm{T}}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}), \tag{3.3}$$

where $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$, denotes the sample mean. The covariance matrix of the transformed vectors $\{L\mathbf{x}_i\}_{i=1}^{n}$ is

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}(L\mathbf{x}_i - L\hat{\boldsymbol{\mu}})^{\mathrm{T}}(L\mathbf{x}_i - L\hat{\boldsymbol{\mu}}) \\
&= \frac{1}{n}L^{\mathrm{T}}\left(\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathrm{T}}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})\right)L \\
&= L^{\mathrm{T}}CL.
\end{aligned} \tag{3.4}$$

We want to choose the linear transformation matrix L to maximize the variances of the transformed vectors (i.e., the trace of $L^{\mathrm{T}}CL$) subject to the constraint that L is a projection matrix. Thus, the optimization problem is given by

$$L = \arg\max_{\tilde{L}} \mathrm{tr}(\tilde{L}^{\mathrm{T}}C\tilde{L}), \qquad \text{subject to:} \quad LL^{\mathrm{T}} = I, \tag{3.5}$$

where $\mathrm{tr}(A) = \sum_i a_{ii}$, is the trace of the matrix A. This optimization problem is solved by letting the rows of L be the leading eigenvectors of C.

PCA is an unsupervised transformation, so it does not use the class labels of the training set to derive informative linear projections. But

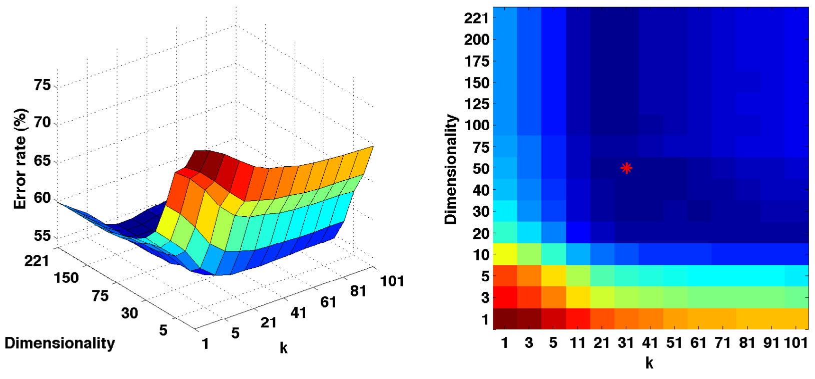
Table 3.9: k-NN Performance with PCA
Error rates (in %) for the k-NN classifier with acoustic observation vectors whose dimensionalities have been reduced by principal component analysis. The error rates for the regular k-NN classifier (with normalized features) are included for ease of comparison.

	PCA	***Regular***
AF Mean	27.3	*24.5*
LIP-LOC	14.1	*13.9*
LIP-OPEN	13.8	*13.6*
TT-LOC	22.2	*21.4*
TT-OPEN	52.9	*45.7*
TB-LOC	43.1	*37.9*
TB-OPEN	45.8	*40.6*
VEL	13.2	*12.1*
GLOT	13.2	*10.3*
Phones	76.4	*70.6*

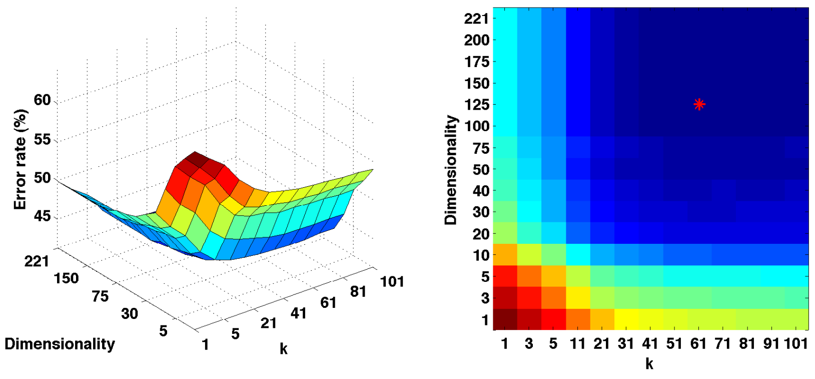
like other eigenvector-based methods (such as singular value decomposition [Hestenes, 1958; Golub and Van Loan, 2012] and latent semantic analysis [Deerwester et al., 1990]) PCA can achieve a “de-noising” effect by projecting out the components of the bottom eigenvectors, which has been noted to often reduce the k-NN error rate [Weinberger, 2007]. PCA is sometimes, particularly in the signal processing literature, referred to as the Karhunen-Loève transform.

We use a freely available Matlab toolkit [van der Maaten, 2010] for PCA. The dimensionality was tuned on a grid from 1 up to the full dimensionality of 221; the optima ranged from 20 to 200. We tried k values up to 501, but performance never improved for $k > 100$; the optima ranged from 11 to 91. Figure 3.5 shows how the parameters impact the error rate for on one fold for three AFs. Although there is variation in how the parameters impact classifier performance for the different AFs, the general trend is that the error rate levels out when either k or the dimensionality or both reach a level of around 30.

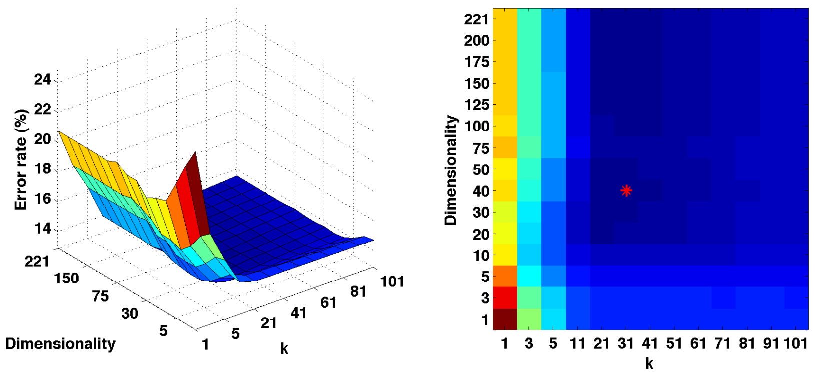
Table 3.9 shows the results. We see that the results are only slightly better than those for unnormalized raw features, and substantially worse than those for normalized raw features. The observation vectors are cus-



(a) TT-OPEN, fold 5



(b) TB-LOC, fold 1



(c) VEL, fold 4

Figure 3.5: Surface and heat plots of the error rate against different settings for the dimensionality of the PCA-transformed observation vectors and the number k of nearest neighbors used by the k-NN classifier for one fold of three different articulatory features. The minimum error rate is marked with red asterisk in the heat plot.

tomarily always centered (by subtracting the sample mean) before doing PCA. We also tried the second part of our global normalization (dividing by the sample standard variance) before doing PCA, but this only hurt performance.[4]

3.5.2 Linear Discriminant Analysis

Linear discriminant analysis [Fisher, 1936] is a supervised method that tries to project the observation vectors into the subspace that best discriminate among the classes. To this end, we want as much variance as possible *between* the classes and as little variance as possible *within* each class. If we, without loss of generality, assume that the data is globally centered, i.e. that the sample mean $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i = 0$, we can express the between-class covariance matrix as

$$C_b = \frac{1}{m}\sum_{c=1}^{m} \hat{\boldsymbol{\mu}}_c \hat{\boldsymbol{\mu}}_c^{\mathrm{T}}, \tag{3.6}$$

where m is the number of classes and $\hat{\boldsymbol{\mu}}_c$ is the sample mean of the observation vectors belonging to class c. The within-class covariance matrix can be expressed as

$$C_w = \frac{1}{n}\sum_{c=1}^{m}\sum_{\mathbf{x}_i \in c} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)^{\mathrm{T}} \tag{3.7}$$

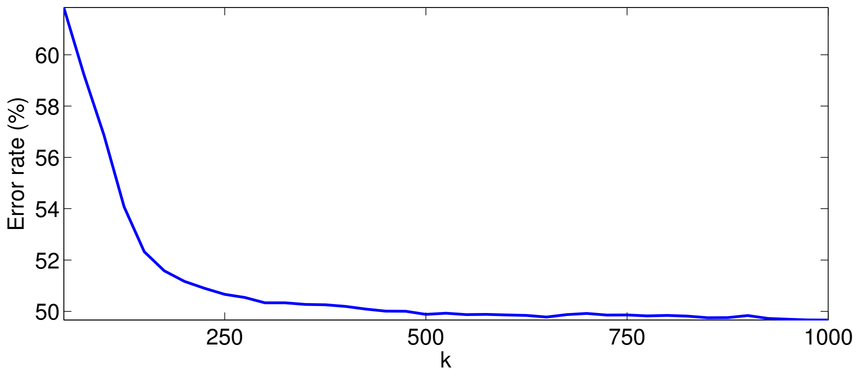
where n is the number of elements in the training set. We want to find the linear transformation matrix L that maximizes the amount of between-class variance relative to the amount of within-class variance, again subject to the constraint that L be a projection matrix. This results in the following optimization problem:

$$L = \arg\max_{\tilde{L}} \operatorname{tr}\left(\frac{\tilde{L}^{\mathrm{T}} C_b \tilde{L}}{\tilde{L}^{\mathrm{T}} C_w \tilde{L}}\right), \qquad \text{subject to:} \quad LL^{\mathrm{T}} = I, \tag{3.8}$$

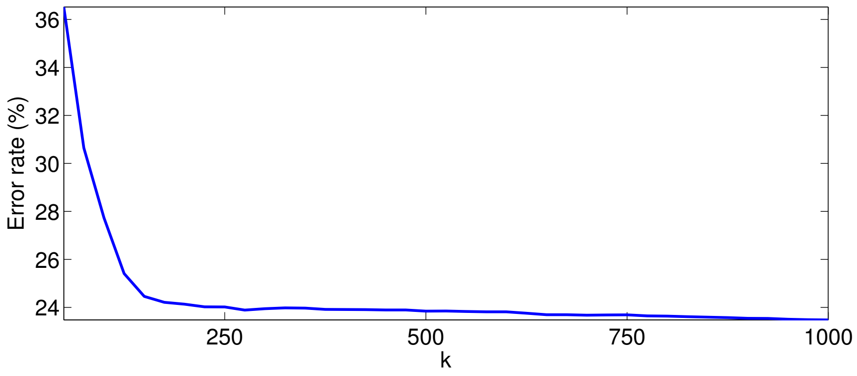
A solution to this optimization problem is to let the rows of L be the leading eigenvectors of $C_w^{-1}C_b$.

We use the same Matlab toolkit [van der Maaten, 2010] for LDA as we did for PCA. The maximum dimensionality of LDA is the cardinality of the classification task minus one. For all the articulatory features the maximum dimensionality is the optimal one across all five folds of the cross-validation.

[4]The increase in error rates for the AFs ranged from 0.21% to 2.05%, absolute.



(a) TT-OPEN, fold 5.



(b) TT-LOC, fold 4.

Figure 3.6: Plots of the error rate against the number k of nearest neighbors used by the k-NN classifier with LDA-transformed observation vectors for one fold of two different AFs. For both of these folds the optimal k is 1001, but the decrease above $k \approx 100$ is marginal.

The optimal k ranges from 71 all the way to 1001, but the improvement for $k > 100$ was insubstantial. Figure 3.6a shows the error rate of TT-OPEN in fold 5 plotted against k. This was the error rate that sank most slowly, and even here the reduction in error rate is fairly marginal above $k = 120$. The other folds with especially high optima for k look more like Figure 3.6b, which shows the fold 4 error of TT-LOC, and where little is gained above $k = 71$. For phones, the optimal dimensionality ranges from 25 to 35, while the optimal k is between 77 and 99 for all five folds.

Table 3.10 shows the results. As we can see they are only marginally better than those for PCA, despite LDA being a supervised method and PCA being unsupervised. The results are substantially worse than for nor-

Table 3.10: k-NN Performance with LDA
Error rates (in %) for the k-NN classifier with acoustic observation vectors whose dimensionalities have been reduced by linear discriminant analysis. The error rates for the regular k-NN classifier (with normalized features) are included for ease of comparison.

	LDA	***Regular***
AF Mean	27.1	*24.5*
LIP-LOC	14.9	*13.9*
LIP-OPEN	14.2	*13.6*
TT-LOC	22.9	*21.4*
TT-OPEN	50.5	*45.7*
TB-LOC	42.3	*37.9*
TB-OPEN	47.1	*40.6*
VEL	13.8	*12.1*
GLOT	11.3	*10.3*
Phones	69.1	*70.6*

malized raw features. Normalizing the features before doing LDA makes no difference to the results.

3.5.3 Locality Preserving Projections

Introduced by He and Niyogi [2003], locality preserving projections (LPP) is, like PCA and LDA, a linear transformation, but it is designed to be more capable of discovering the nonlinear structure of the data manifold. It attempts to preserve the proximity structure of the data points through the use of an adjacency graph of neighboring points. LPP can be considered a linear approximation to Laplacian eigenmaps [Belkin and Niyogi, 2003]. Promising results have been published for ASR applications [Tang and Rose, 2008]. We consider here the variant of LPP with a k-NN adjacency graph and heat kernel weights.

Given a training set $\mathbf{x}_1, ..., \mathbf{x}_n$ in $\mathbb{R}^d$ we want to find a transformation matrix L that maps the training set to vectors $\mathbf{y}_1, ..., \mathbf{y}_n$ in a lower-dimensional space $\mathbb{R}^l$. The first step of LPP is to construct the adjacency graph G with m nodes. We put an edge between nodes i and j if, for a chosen number $k \in \mathbb{N}$, $\mathbf{x}_i$ is among the k nearest neighbors of $\mathbf{x}_j$, or if $\mathbf{x}_j$ is among the k nearest neighbors of $\mathbf{x}_i$. We then let W be the sparse symmetric matrix where the element W_{ij} is zero if there is no edge joining i and j in the graph G. If there is an edge joining i and j in G, we let

$$W_{ij} = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}_j)}{t}\right), \tag{3.9}$$

where $t \in \mathbb{R}$ is a chosen parameter. W constitutes a weighting of the graph G.

We want our transformation L to be such that if points $\mathbf{x}_i$ and $\mathbf{x}_j$ are close, then so are the transformed points $\mathbf{y}_i$ and $\mathbf{y}_j$. Thus, a good transformation would minimize the following objective function

$$\sum_{ij} (\mathbf{y}_i - \mathbf{y}_j)^{\mathrm{T}}(\mathbf{y}_i - \mathbf{y}_j) \cdot W_{ij}. \tag{3.10}$$

This objective function will incur a heavy penalty if neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ are mapped far apart. A solution to the resulting optimization problem is to compute the eigenvectors $\mathbf{a}_i$ and eigenvalues λ_i of the generalized eigenvector problem

$$XAX^{\mathrm{T}}\mathbf{a}_i = \lambda_i XDX^{\mathrm{T}}\mathbf{a}_i, \tag{3.11}$$

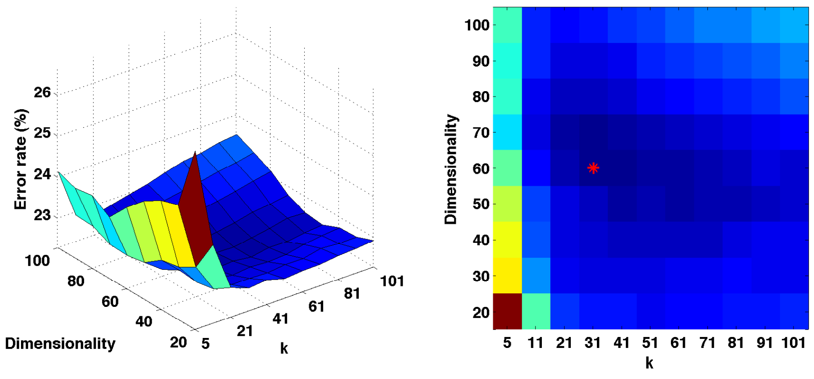
where D is a diagonal matrix of column (or, equivalently, row) sums of W, i.e., $D_{ii} = \sum_j W_{ji} = \sum_j W_{ij}$; $A = D - W$ is the Laplacian matrix; and the ith column of the matrix X is $\mathbf{x}_i$. Then, let the rows of the transformation matrix L be the l leading eigenvectors $\mathbf{a}_i$.

In our experiments, we find the optimal dimensionalities to range from 40 to 80, while the optima for k range from 5 to 101. Figure 3.7 shows how the parameters impact the error rate for three representative AFs on one fold. As ususal the performance evens out at around $k = 30$; the error tends to increase for higher ks, but usually not substantially. We notice that the impact of the dimensionality is unlike what we saw for PCA, however. The error rates will sink until the dimensionality reaches its optimum, but then it will tend to rise, sometimes dramatically, for higher dimensionalities.

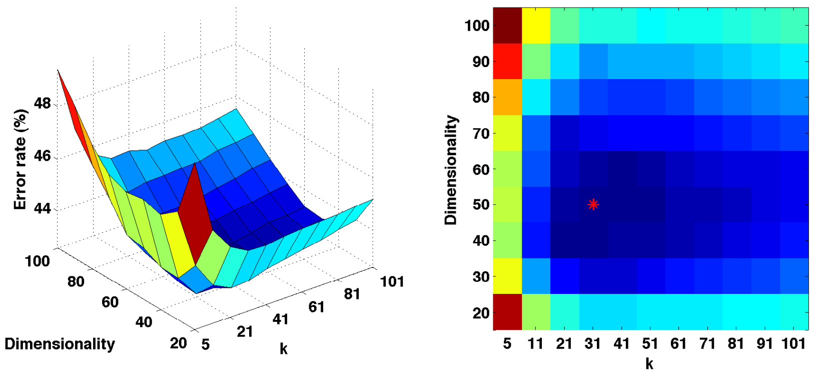
Table 3.11 shows the results with the optimal parameter settings. There is a slight improvement with raw observation vectors, but a slight deterioration when observation vectors are normalized. Nevertheless, unlike PCA and LDA, LPP gives better results with normalized than raw observation vectors.

3.6 Canonical Correlation Analysis

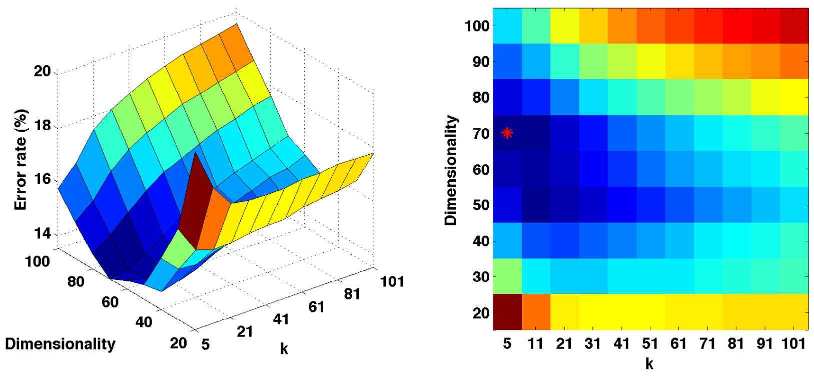
Canonical correlation analysis (CCA) [Hotelling, 1936; Borga, 1998] is a technique that given a set of paired vectors $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$



(a) TT-LOC, fold 4



(b) TB-OPEN, fold 1



(c) GLOT, fold 4

Figure 3.7: Surface and heat plots of the error rate against different settings for the dimensionality of the LPP-transformed observation vectors and the number k of nearest neighbors used by the k-NN classifier for one fold of three different articulatory features. The minimum error rate is marked with red asterisk in the heat plot.

Table 3.11: k-NN Performance with LPP
Error rates (in %) for the k-NN classifier with raw and normalized acoustic observation vectors whose dimensionalities have been reduced by locality preserving projections. The error rates for the regular k-NN classifier without LPP are included for ease of comparison.

	Raw		**Normalized**	
	LPP	*Regular*	LPP	*Regular*
AF Mean	26.4	*27.5*	25.5	*24.5*
LIP-LOC	14.1	*14.1*	14.3	*13.9*
LIP-OPEN	13.9	*13.8*	13.9	*13.6*
TT-LOC	22.5	*22.2*	22.4	*21.4*
TT-OPEN	50.0	*53.0*	48.2	*45.7*
TB-LOC	40.7	*43.2*	39.7	*37.9*
TB-OPEN	43.0	*46.4*	42.3	*40.6*
VEL	13.9	*13.4*	12.8	*12.1*
GLOT	13.4	*14.3*	9.9	*10.3*
Phones	72.9	*76.5*	72.9	*70.6*

finds the directions that maximize the correlation between them. This is commonly used in situations where one has two views of a data set, and wishes to find a common representation for these two views.

Let $X = \{\mathbf{x}_i\}$ and $Y = \{\mathbf{y}_i\}$. CCA finds pairs of directions $(\mathbf{v}_k, \mathbf{w}_k)$ such that the projections of X and Y onto those directions, $\mathbf{v}_k^{\mathrm{T}} X$ and $\mathbf{w}_k^{\mathrm{T}} Y$, are maximally correlated. $\mathbf{v}_k^{\mathrm{T}} X$ and $\mathbf{w}_k^{\mathrm{T}} Y$ are called the *canonical variables.* Thus, the first pair of directions is given by

$$\arg\max_{v,w} \rho(\mathbf{v}^{\mathrm{T}} X, \mathbf{w}^{\mathrm{T}} Y), \tag{3.12}$$

where $\rho(\mathbf{x}, \mathbf{y}) = \frac{\mathrm{Cov}(\mathbf{x},\mathbf{y})}{\sqrt{\mathrm{Var}(\mathbf{x})\,\mathrm{Var}(\mathbf{y})}}$ is the correlation between $\mathbf{x}$ and $\mathbf{y}$.

To solve this optimization problem, let C_{xx} and C_{yy} be the auto-covariance matrices of X and Y respectively, and let $C_{xy} = C_{yx}^{\mathrm{T}}$ be the cross-covariance matrix. In general the vectors $\mathbf{v}_k$ are the eigenvectors of $C_{xx}^{-1} C_{xy} C_{yy}^{-1} C_{yx}$, and the vectors $\mathbf{w}_k$ are the eigenvectors of $C_{yy}^{-1} C_{yx} C_{xx}^{-1} C_{xy}$. We only need to solve one of these eigenvector problems, since $\mathbf{w}_k = C_{yy}^{-1} C_{yx} \mathbf{v}_k$. The rank of these eigenvalue problems determines the number of pairs of directions

$(\mathbf{v}_k, \mathbf{w}_k)$ we can find. If the matrices C_{xx}, C_{yy} and C_{xy} are full rank, this number will equal the lower of the dimensionalities of $\mathbf{x}$ and $\mathbf{y}$.

In our experiments, we use the acoustic observation vector as the first view. As the second view we use an 8-dimensional vector of the articulatory labels for this observation vector. Thus, we effectively use CCA as a supervised dimensionality reduction, with the vectors $\mathbf{v}_k$ as the rows of our transformation matrix, L.

The resulting transformation is in many ways similar to LDA, but uses the labels for all eight AFs rather than just one and allows for a slightly higher dimensionality (8 for this way of using CCA, versus from 1 to 5 for LDA). As demonstrated by De la Torre [2008], LDA can be regarded as a special case of CCA. Consider a classification task of cardinality m, where observation vectors $\mathbf{x}_1, ..., \mathbf{x}_n$ belong to classes $1, ..., m$. If the first view of CCA consists of the observation vectors $\mathbf{x}_i$, while the second view consists of m-dimensional binary vectors $\mathbf{y}_i$, where the jth element of $\mathbf{y}_i$ equals one if the label of $\mathbf{x}_i$ equals j, and zero otherwise. In this case CCA is equivalent to LDA.

The results are shown in the first column of Table 3.12. We see that the performance is slightly better than it is for LDA, but still worse than for normalized raw features. We have tried adding regularization to our experiments, following the method of De Bie and De Moor [2003], but did not see any improvement in results.

Multi-Task Linear Discriminant Analysis We have seen that CCA used as a transform performs better than the related LDA transformation, perhaps partly due to its allowing for a higher dimensionality. Our experiments with PCA and LPP indicate, however, that the optimal dimensionality for our classification tasks is higher still.

The application of CCA described above allowed for only 8 dimensions. This was determined by how we defined the second view of the data. In order to increase the dimensionality, we make the second view a binary vector with one element for each possible value of each of the eight AFs. The dimensionality of this binary vector is equal to the sum of the cardinalities of the AFs, i.e., 32. However, since exactly one of the elements corresponding to the values of each AF must be equal to 1, there are just $32 - 8 = 24$ degrees of freedom to this vector. The resulting eigenvector problem is therefore of rank 24, which is thus the dimensionality of the transformed features—triple that of regular CCA. To the best of our knowledge, CCA has not been applied in this way before. As we saw above, CCA is equivalent to LDA when the second view is a binary vector indicating the label of a

Table 3.12: k-NN Performance with CCA Variants
Error rates (in %) for the k-NN classifier with acoustic observation vectors whose dimensionalities have been reduced using canonical correlation analysis. MT-LDA denotes multi-task linear discriminant analysis, and MT-LDA+$\perp$ denotes MT-LDA vectors concatenated with residue vectors from the orthogonal space transformed with LDA. The error rates for the regular k-NN classifier (with normalized features) are included for ease of comparison.

	CCA	**MT-LDA**	**MT-LDA+$\perp$**	***Regular***
AF Mean	26.16	24.66	24.40	*24.45*
LIP-LOC	14.4	14.1	14.1	*13.9*
LIP-OPEN	14.2	13.9	13.8	*13.6*
TT-LOC	22.4	22.0	21.8	*21.4*
TT-OPEN	49.4	46.1	45.6	*45.7*
TB-LOC	41.7	38.4	37.8	*37.9*
TB-OPEN	43.8	40.7	40.6	*40.6*
VEL	12.9	12.2	12.0	*12.1*
GLOT	10.5	9.9	9.5	*10.3*
Phones	76.7	71.1	69.0	*70.6*

single classification task. Our application of CCA here is similar to this, except that we use the labels for all our eight classification tasks, thus creating a multi-task variant of LDA that we will refer to as multi-task linear discriminant analysis (MT-LDA).

The results are shown in the second column of Table 3.12. We see a consistent improvement over regular CCA ($p < 10^{-7}$ for all AFs).

Adding Orthogonal Information We can find the vector space that is orthogonal to the space spanned by the CCA (or MT-LDA) directions by computing the null space of the transformation matrix, L. If we project the MFCC observation vectors onto this space we get a sort of residue, which is not expressed by the CCA-transformed observation vectors. Since these “residue vectors” contain information that is not in the CCA vectors, concatenating the CCA vectors with elements from the residue seems like a promising approach.[5]

[5]This approach is inspired by work on private learning in multi-view settings [Ek et al., 2008; Salzmann et al., 2010].

The residue vectors have the same dimensionality as the original 9-frame MFCC vectors, 221. Concatenating this to a CCA or MT-LDA vector of dimensionality 8 or 24 would give too much weight to the residue. It is therefore natural to reduce the dimensionality of the residue vector before concatenating.

We have experimented with concatenating PCA- and LDA-transformed residue vectors to both CCA- and MT-LDA-transformed observation vectors. The third column of Table 3.12 shows the best results achieved, for MT-LDA vectors concatenated with residue vectors whose dimensionalities are reduced with LDA. The dimensionalities of the LDA-transformed residue vectors are the maximum for each AF (i.e., from 1 to 5), concatenated to the MT-LDA vectors this gives observation vectors of dimensionality between 25 and 29. We see a small but consistent gain over the regular MT-LDA results ($p < 10^{-4}$ for six of the eight AFs), indicating that the residue vectors do indeed contain some useful information for the classification.

3.7 Using MLP Posteriors for k-NN Classification

As an alternative to using linear transformations on the observation vectors, we consider the nonlinear mapping given by class posteriors from MLPs trained to classify AFs. This approach is inspired, in part, by the tandem approach to ASR (described in Section 2.2) where the log-posteriors of MLPs trained for phone classification were used in addition to MFCCs in a conventional GMM-HMM system. Note that the use of such posterior representations can be thought of as a nonlinear dimensionality reduction technique, wherein the MFCCs are mapped to the space of MLP posteriors. In contrast to other work using phone-based MLPs, we are addressing multiple classification tasks and will have multiple MLPs. For the classification of an articulatory feature, we have a choice of either using only the MLP posteriors (MLPPs) for that feature or concatenating posteriors from multiple MLPs. Using concatenated MLPPs to classify each AF may be preferred, as it allows us to implicitly take into account the dependencies between AFs. Because of these dependencies, it may be helpful to know, when classifying one AF, the "opinions" of MLPs for other AFs.

For our k-NN classification experiments using these MLP posteriors, we considered both of these setups: using only the MLPPs corresponding to a single AF for the classification of that AF, and using the concatenation of all of the MLPPs for each AF. We also tried concatenating either of these with the MFCC vectors (either single-frame and 9-frame windows) to create a "tandem-like" feature vector. In addition, we consider using

Table 3.13: MLPP k-NN Performance
Error rates (in %) for the k-NN classifier using posterior class probabilities from the MLP as its features. The error rates for the MLP baseline are included for ease of comparison.

	MLPP k-NN	*MLP*
AF Mean	21.8	*22.6*
LIP-LOC	13.5	*13.8*
LIP-OPEN	12.7	*12.9*
TT-LOC	19.8	*20.8*
TT-OPEN	40.1	*41.6*
TB-LOC	33.8	*35.4*
TB-OPEN	35.5	*36.6*
VEL	10.7	*11.2*
GLOT	8.6	*8.6*
Phones	67.1	*63.9*

either raw posterior probabilities, log-posteriors or the linear outputs of the MLPs in these setups. These features were tried in both normalized and non-normalized variants.

The best performing method was that using the non-normalized concatenation of the raw posterior probabilities for each AF, and not adding the MFCC vector. The dimensionality of these feature vectors is 32. Table 3.13 shows the results. We see that this method does slightly better than the MLP baseline for all the AFs,[6] for seven of them the p-values are below 10^{-5}, but interestingly the performance for phones is below that of the MLP.

Figure 3.8 shows confusion matrices for all of the AFs, using the MLPP k-NN classifiers. Although we see that most of the classifications fall along the diagonal as they should, there is a tendency for the classes with the highest prior probabilities to dominate. We see these as dark columns in the confusion matrices. This is perhaps most pronounced for LIP-OPEN, which was to be expected since this AF has the lowest error rate for the chance classifier, despite there being three AFs with lower cardinalities.

[6]The error rates for GLOT are 8.59% with MLPP k-NN and 8.64% with the MLP when using two decimal places ($p = 0.08$).

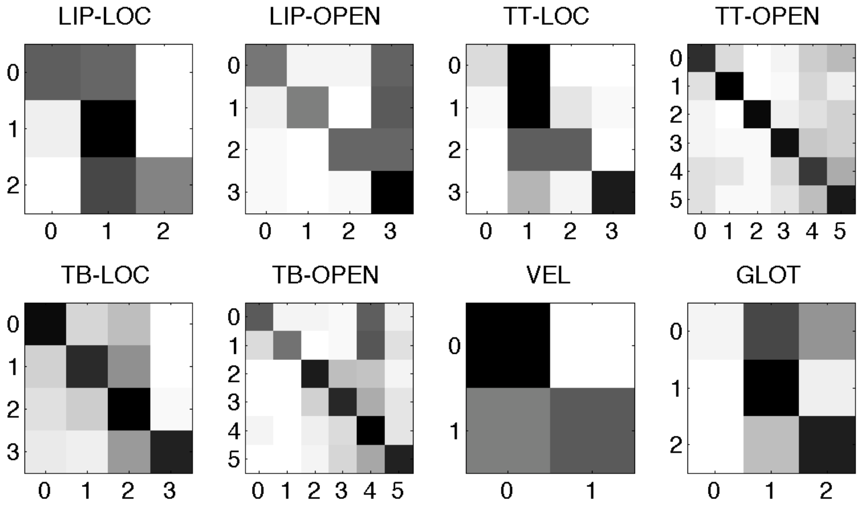


Figure 3.8: Grayscale heat plots of the confusion matrices for k-NN AF classification using MLPP inputs. Rows corresponds to true classes and columns to predicted classes; darker cells indicate higher values.Thus, a dark diagonal means a largely correct classification of all classes.

Table 3.14: STP Frame Classification Results Summary
All error rates (in %) presented in this chapter.

	LIP-LOC	LIP-OPEN	TT-LOC	TT-OPEN	TB-LOC	TB-OPEN	VEL	GLOT	AF Mean	Phones
Chance	15.1	14.4	24.2	71.7	55.4	55.7	14.7	20.3	33.9	95.3
MLP										
1-frame	14.3	14.0	21.9	44.5	37.4	39.4	11.8	9.5	24.1	67.1
9-frame	13.8	12.9	20.8	41.6	35.4	36.6	11.2	8.6	22.6	63.9
***k*-NN**										
Raw, 1-frame	14.6	14.1	23.3	55.7	45.0	47.5	13.4	15.1	28.6	78.6
Raw, 9-frame	14.1	13.8	22.2	53.0	43.2	46.4	13.4	14.3	27.5	76.5
Joint, 1-frame	16.6	15.9	26.7	56.9	49.6	52.6	16.8	15.0	31.3	-
Joint, 9-frame	16.5	15.8	24.6	54.1	47.9	50.9	17.0	13.9	30.1	-
Normalized, 1-frame	14.0	13.8	21.4	45.9	37.7	39.9	11.5	10.2	24.3	70.7
Normalized, 9-frame	13.9	13.6	21.4	45.7	37.9	40.6	12.1	10.3	24.5	70.6
PCA	14.1	13.8	22.2	52.9	43.1	45.8	13.2	13.2	27.3	76.4
LDA	14.9	14.2	22.9	50.5	42.3	47.1	13.8	11.3	27.1	69.1
LPP	14.1	13.9	22.5	50.0	40.7	43.0	13.9	13.4	26.4	72.9
LPP, Normalized	14.3	13.9	22.4	48.2	39.7	42.3	12.8	9.9	25.5	72.9
CCA	14.4	14.2	22.4	49.4	41.7	43.8	12.9	10.5	26.2	76.7
MT-LDA	14.1	13.9	22.0	46.1	38.4	40.7	12.2	9.9	24.7	71.1
MT-LDA+$\perp$	14.1	13.8	21.8	45.6	37.8	40.6	12.0	9.5	24.4	69.0
MLPP	13.5	12.7	19.8	40.1	33.8	35.5	10.7	8.6	21.8	67.1

3.8 Discussion

Table 3.14 shows all the results presented in this chapter. We see that the MLP classifier outperforms the k-NN classifiers regardless of which linear transformations is used on the features for the latter. The k-NN classifier using MLP posteriors, however, outperforms the MLP classifier for all of the AFs.

Joint k-NN classification yields a substantially worse performance than individual k-NN classification. This clear discrepancy could be taken as an indication that there are no dependencies between the separate AFs. This is contradicted, however, by the fact that the MLPP features that yielded the best performance were the ones where the class posteriors for all the AFs were concatenated. If there really was little or no dependency between the AFs, it should work better to include just the posteriors for the AF that was being classified. One could argue that the only reason the concatenated MLPP features worked better was their higher dimensionality, but we also tried to concatenate the posteriors for the AF being classified with single- or 9-frame MFCC features without seeing any improvement in results. There do seem to be dependencies between the AFs that classification can benefit from, but straightforward joint classification was not the way to exploit them, perhaps because the dependencies are only among subsets of the AFs.

Using 9-frame concatenated observation vectors rather than single-frame features improves results for both the MLP and k-NN classifiers. For k-NN this is the case for both raw and transformed features (for the transformed features we reported results only for the 9-frame case). There is one exception: For normalized features without any transformation, single-frame features outperform 9-frame features for half of the AFs. The concatenation of frames adds context, but it also adds some confusability, by sometimes including frames from neighboring phones. It is therefore not unreasonable that concatenating frames can cause performance to deteriorate in some cases. Why this would be the case for normalized features and not for raw features, however, is not clear.

All of the linear transformations improve the results over using raw features, but the gains are quite small, and applying a global normalization to the features improves results more. It is surprising that well-established methods like PCA and LDA yield such poor results for this task. For LDA's part this seems to have to do with the low dimensionality it imposes for the AFs, due to their low numbers of classes. k-NN with LDA-transformed features has the highest relative deterioration to k-NN on raw 9-frame features on LIP-LOC, which has a cardinality of 3, and thence a maximum LDA dimensionality of 2. There is a weak tendency for LDA to do less bad on the AFs with higher cardinalities. For GLOT, however, which also has a cardinality of 3, LDA features do substantially better than raw features. We also notice that for phones, where the maximum dimensionality of LDA is 98, and the optimum dimensionalities found in cross-validation tuning were lower than that, k-NN with LDA-transformed features does a lot better than k-NN with raw features.

As we saw in Figure 3.5, the dimensionality of the PCA-transformed features did not have great effect above a certain threshold. Even though the optimum dimensionality could be as low as 20 (out of the 221 possible), the difference in error rate as the dimensionality increased was not so great. This is an indication that the "de-noising" effect of PCA, which can be a great benefit of reducing dimensionalities with eigenvector-based methods, is not helpful to us for our classification task. Although the usually steep fall in the error rate as the dimensionality increases up to around 10 or 20 does indicate that the top eigenvectors capture the most important information for the classification, the fact that we see no rise in the error rate on the other end of the dimensionality axis suggests that PCA does not manage to relegate to the bottom eigenvectors the information that is superfluous to the classification.

Using CCA as a dimensionality reduction gave better results than PCA and LDA. As mentioned the resulting transformation from using CCA as a dimensionality reduction is quite similar to LDA, except for allowing for a higher dimensionality. The improvement over LDA is a sign that a higher dimensionality than those allowed by LDA is important for our task.

The improvement by our new method MT-LDA over standard CCA is another indication of the importance of dimensionality. MT-LDA does not add any more information than CCA, but by reparametrizing the vector of AF values to a binary vector it allows for a threefold increase of the dimensionality, from 8 to 24. This gives us a mean relative decrease in error rate by 4.8% for the AFs.

Adding LDA-transformed features from the orthogonal space of MT-LDA gave a further 1.2% relative decrease in the error rate compared to regular CCA. This gain was small, but consistent across all eight AFs. We conjecture that adding the few dimensions allowed by LDA (2 to 6) to the 32-dimensional vector gave adequate weight to the extra information contained in the orthogonal space, without introducing too much noise relative to the information captured by the CCA transformation.

Given that the supervised methods LDA and CCA both do better than the unsupervised PCA, and considering the importance that dimensionalities seem to play, LPP seemed like a promising transformation—by virtue both of being supervised and of allowing for all dimensionalities. The performance was only marginally better than that of PCA and LDA, however, and somewhat worse than for any of the CCA-based methods.

MLPP k-NN was the only method that outperformed the MLP baseline, attaining a mean decrease in error rate of 3% for the AFs, relative to the MLPs. The fact that it was also the only truly nonlinear of our transformations raises the question if nonlinear transformations might not be better suited to our task. This is an open question. Van der Maaten et al. [2009] argue that although nonlinear transformations are often shown to perform better on artificial data sets, they are also often unable to outperform linear techniques like PCA on real-world data. The disappointing performance of LPP, which although linear, tries to approximate nonlinear transformations, does not add to the promise of nonlinear mappings for our task. Since both the learning and application of nonlinear transformations tends to be more involved than linear ones, it could also be argued that they to some extent undermine the simplicity and the lack of training time of the k-NN approach. Nevertheless, nonlinear transformations would be an interesting avenue for future work.

Future work

An important question given our task is: How can you use the fact that you are doing multiple related classification tasks to find good features for all of them? Several of our more successful feature transformations (like CCA, MT-LDA and MLPP) implicitly make use of information across the classification tasks. We saw, however, that joint AF classification did not perform nearly as well as individual AF classification. The AFs arguably cluster naturally in three groups: the ones pertaining to the lips, the ones pertaining to the tongue, and the two pertaining to the glottis and velum, and some previous work [Prabhavalkar et al., 2011; Jyothi et al., 2011] use this grouping. Doing the classification jointly for each of these AF clusters could be a promising middle way between joint and individual classification.

We saw in the confusion matrices of Figure 3.8 that there is a tendency for the class priors to dominate. While this is natural, it may be problematic and should be particularly so for large values of k. One possible way to mitigate this effect is to weight neighbors differently depending on their distances from the test vector, a possible improvement for future work. (We have, however, experimented with distance weighting when applying the results of the k-NN frame classification in systems for automatic transcription—cf. Section 4.1.)

One step further from changing the distance weighting would be to replace the distance metric altogether. Some work in k-NN classification for speech has seen promising results when using the cosine distance rather than the Euclidean distance [Asaei et al., 2010b]. Investigating how a change in distance metric would affect our experiments is a promising avenue for further research.

As mentioned, the relatively poor performance of the linear transformations compared to the nonlinear MLPPs indicates that other nonlinear transformations might hold promise. Two natural extensions of our work would be to investigate the kernelized variants of LDA and CCA. Kernel discriminant analysis (KDA) [Mika et al., 1999; Baudat and Anouar, 2000] has shown some promise, but it will pose the same restrictions on the dimensionalities of the features as LDA. Kernel canonical correlation analysis (KCCA) [Akaho, 2001; Hardoon et al., 2007] might therefore be a more promising technique. It is not obvious, however, how kernelization would affect the performance of MT-LDA, with its binary input vectors, so dimensionality might turn out to be an issue for KCCA as well.

Our MLPP k-NN method involved using the posterior probabilities output by the final layer of an MLP trained for our classification task as input features for a k-NN classifier, inspired by tandem features for HMMs. In work with *bottleneck features*, the outputs of a hidden layer in a neural network are used as features instead of the posteriors. In early work (e.g., Grézl et al. [2007]), bottleneck features were typically generated by MLPs with three hidden layers, where the middle layer was the "bottleneck". More recently, deep neural networks have often been used [Yu and Seltzer, 2011; Tüske et al., 2013; Gehring et al., 2013]. Since one can choose the number of hidden nodes in the bottleneck layer, and thus the dimensionality of the resulting features, the technique of extracting bottleneck features is a more flexible transformation than our MLPP method. Since MLPP k-NN performed so well for our task, and since the optimal dimensionalities in our experiments often seemed to be higher than the MLPP's 32 dimensions, bottleneck features seem to hold some promise for our task.

Chapter 4

Nearest Neighbor Features in Transcription Systems

In this chapter we will investigate how we can use the frame classification results from Chapter 3 to create observation vectors for two systems for automatic transcription, effectively using the k-nearest neighbor classifiers as part of the front-end for these systems. The first system is a conditional random field for forced transcription of articulatory features; the second is a tandem hidden Markov model for phone recognition. In both of these cases the k-NN classifiers are replacing MLP classifiers. Before we look at these two applications in detail, we shall consider how we can create feature vectors from our k-NN classifiers that can replace the posterior probabilities output by MLPs.

4.1 k-NN Feature Functions

A neural network frame classifier like the MLPs described in Section 3.3 can output, for each frame of the data set, a posterior probability of it belonging to each possible class. Such posterior probabilities can in turn be used as input to other systems, e.g. tandem HMMs, as we saw in Section 2.2.

A k-NN classifier will usually just output a decision, i.e., which class it considers it most likely that the query frame belongs to. However, we also have access to the nearest neighbors found by the classifier, and to their distances to the query frame. In this section, we will consider three ways of using this information to create "posterior-like" features.

Proportion of Nearest Neighbors (`knn1`) We are in a sense trying to approximate the probability of the query frame belonging to each class. The most straightforward approach is to use the proportion of the k nearest neighbors belonging to the class as a proxy for this probability.

To express this formally, we introduce some notation: Let $\mathbf{x}_t$ be the acoustic observation vector for the query frame with time index t, and let $\eta(k, \mathbf{x}_t)$ be a function that returns the index of the kth nearest neighbor of $\mathbf{x}_t$ in the training set. Let AF^i be the ith articulatory feature (i.e., for the feature set shown in Table 3.1, AF^1 is LIP-OPEN, AF^2 is TT-LOC, etc.), and let a^i_j be the jth value that AF^i can take. Finally, let AF^i_m be the value of AF^i for the frame in the training set with index m.

Then, the first feature function, giving an estimate of the probability $P(\mathrm{AF}^i_t = a^i_j)$, can be expressed as

$$f_{\mathtt{knn1}}\big(a^i_j, \mathbf{x}_t\big) = \frac{1}{k} \sum_{l=1}^{k} \delta\big(\mathrm{AF}^i_{\eta(l,\mathbf{x}_t)} = a^i_j\big), \tag{4.1}$$

where $\delta(\cdot)$ is the indicator function, which equals 1 if its argument is true and 0 otherwise. Here, each of the k nearest neighbors gets one vote.

Rank-Weighted Proportion of Nearest Neighbors (`knn2`) It is a natural assumption that the closest neighbors of $\mathbf{x}_t$ are more reliable predictors of the class of $\mathbf{x}_t$ than the more distant neighbors. We can therefore try to weight the influence of the neighbors by giving more votes to the ones that are closer to the query vector. The simplest way of doing this is to give k votes to the closest neighbor, $k - 1$ votes to the second closest neighbor and so on down to the most distant of the k nearest neighbors, which gets one vote.

It would be natural to normalize by the total number of votes, which is the arithmetic series $1+2+\ldots+k = \frac{k(k+1)}{2}$, in order to get the k-NN features to sum to 1. However, our experiments have shown, somewhat surprisingly, that normalizing by k works better in both of the transcription systems we will present in the following sections.

We can thus express the second feature function as

$$f_{\mathtt{knn2}}\big(a^i_j, \mathbf{x}_t\big) = \frac{1}{k} \sum_{l=1}^{k} \delta\big(\mathrm{AF}^i_{\eta(l,\mathbf{x}_t)} = a^i_j\big)(k + 1 - l). \tag{4.2}$$

Distance-Weighted Proportion of Nearest Neighbors (`knn3`) Another way to give more influence to the closer neighbors is to weight them according to their distance from the query vector rather than their rank among the neighbors. Since we want to assign less weight to the more distant neighbors, we sum the inverse distances of the neighbors belonging to the considered class (i.e., all the frames where AF^i has value j) and normalize by the sum of the inverse distances for all the neighbors. Thus, we express the third feature function as

$$f_{\text{knn3}}\big(a_j^i, \mathbf{x}_t\big) = \frac{1}{\sum_{l=1}^{k} \mathcal{D}^{-1}(\mathbf{x}_t, \mathbf{x}_{\eta(l,\mathbf{x}_t)})} \sum_{l=1}^{k} \delta\big(\mathrm{AF}^i_{\eta(l,\mathbf{x}_t)} = a_j^i\big) \mathcal{D}^{-1}(\mathbf{x}_t, \mathbf{x}_{\eta(l,\mathbf{x}_t)}), \tag{4.3}$$

where $\mathcal{D}^{-1}(\mathbf{x}_1, \mathbf{x}_2)$ is the inverse Euclidean distance between vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. This can be considered a variant of rank-weighting, where the weight is dependent on the inverse distance, which is monotonic with rank.

Constructing feature vectors We create observation vectors for the transcription systems we will consider in the following sections, by concatenating values for one of the feature functions above for all of the possible values for each AF. So, for a given acoustic observation vector $\mathbf{x}_t$, we construct a new observation vector $\mathbf{x}_t^*$ by computing, for example, $f_{\text{knn3}}\big(a_j^i, \mathbf{x}_t\big)$ for all possible combinations of i and j, and concatenating these computed values to form $\mathbf{x}_t^*$.

4.2 Forced Transcription on Switchboard

The first system we shall insert the k-NN features described in the previous section into is a conditional random field (CRF) for forced transcription of articulatory features. We take the system of Prabhavalkar et al. [2011] as our starting point, and try to replace the features based on MLP classifiers in that system with features based on our k-NN classifiers. We will see that the performance with k-NN features improves slightly upon the results with MLP features.

We shall first give some explanation of the task of AF forced transcription, before we present the experimental setup in Sections 4.2.2, 4.2.3 and 4.2.4, and then move on to the results of these experiments in Sections 4.2.5 and 4.2.6.

4.2.1 Forced Transcription of Articulatory Features

In Chapter 3 we obtained articulatory labels for our data set by mapping from phones to AFs. Although the detailed phonetic labeling of the STP corpus makes this slightly more palatable than it would otherwise be, it is plainly suboptimal—particularly since we expect the AFs to change their values not only at the phone boundaries. There is, however, a lack of data transcribed at the articulatory level. Although some attempts at manual transcription have been made [Livescu et al., 2007c], this is too labor intensive to be feasible for a larger data set. Some work has been done on inferring subject-independent AFs from physical measurements [Ghosh and Narayanan, 2011], but this work has so far not produced any speech corpus.

Forced transcription is commonly applied to obtain phonetic transcriptions and, as we saw on page 47, Prabhavalkar et al. [2011] showed one way of employing this technique to obtain an articulatory transcription.

The task of AF forced transcription is to obtain an articulatory labeling given the acoustics and the word labeling—i.e., given a word transcript of an utterance and a series of consecutive acoustic observation vectors, one for each time frame of the utterance, we want to predict the most likely values of the AFs at each time frame. In the system we will consider, each utterance is a single word. Since there is a large number of speech corpora labeled at the word level, a good system for AF forced transcription would be very useful.

Note that this is a very different task from the one examined in Chapter 3. There, we tried to infer the values of the AFs for a single frame of the audio signal given the observation vector for that frame only. Here we are given a sequence of observation vectors as well as which word they represent.

An advantage of using k-NN-based features in such a system, rather than features from MLPs, is that it allows for easy experimentation with forced transcription of different sets of AFs.

4.2.2 The Conditional Random Field Model

Figure 4.1 shows the model presented in Prabhavalkar et al. [2011]. The observed variable `Word` gives the identity of the word that is being uttered in the current frame. The other observed variable, $\mathbf{x}$, is the observation vector, which is what we are changing in our experiments.

In this model it is assumed that the four AFs relating to the tongue (TT-LOC, TT-OPEN, TB-LOC and TB-OPEN) are completely synchronized. It is also assumed that the two features relating to the lips (LIP-LOC and LIP-OPEN) are synchronized and that VEL and GLOT are synchronized. Thus, the model effectively has only three articulatory features, shown as $\texttt{AF}^1$, $\texttt{AF}^2$ and $\texttt{AF}^3$ in Figure 4.1. We shall refer to these as L (Lips), T (Tongue) and G

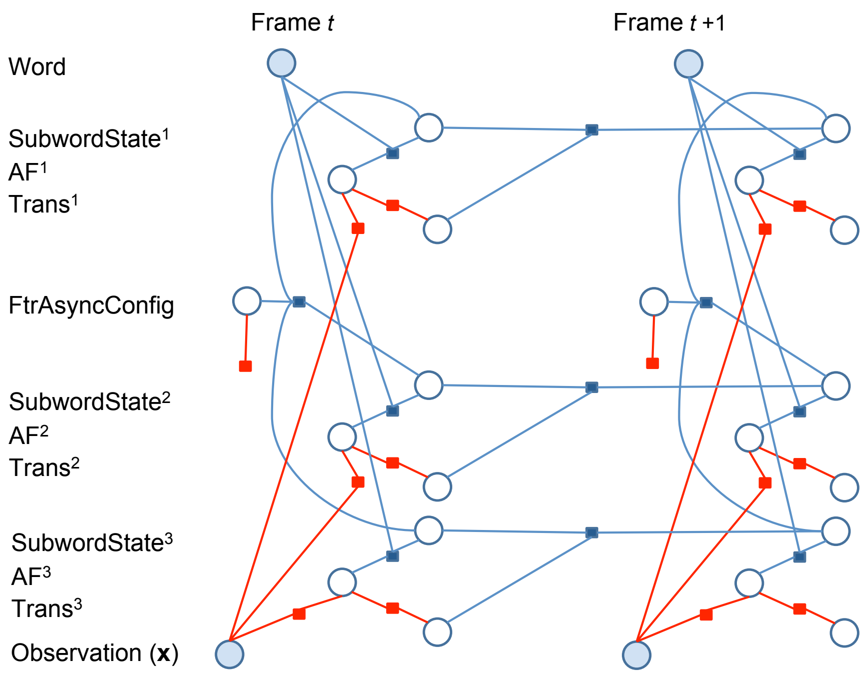


Figure 4.1: Factor graph for the conditional random field model used for forced transcription of the AFs [Prabhavalkar et al., 2011]. A template for three AFs, unrolled for two frames, is shown. The shaded circles represent observed variables while the white circles represent hidden variables. The red and blue square nodes represent factors: non-negative functions over the set of variables connected to them.

(Glottis/Velum). Each value of T is a combination of values of TT-LOC, TT-OPEN, TB-LOC and TB-OPEN—e.g., alveolar/closed/uvular/wide. Without any constraints, the cardinality of T would be $4 \cdot 6 \cdot 4 \cdot 6 = 576$. However, we restrict ourselves to configurations that are seen in the data, and we also allow a maximum asynchrony of one subword state between any pair of L, T and G. Considering this constraint, the cardinalities of L, T and G are 8, 25 and 4, respectively.

We see that each articulatory feature has three variables associated with it: $\texttt{AF}^i$, $\texttt{SubwordState}^i$ and $\texttt{Trans}^i$. $\texttt{AF}^i$ gives the value of the ith articulatory feature for the current frame. $\texttt{SubwordState}^i$ is a counter variable for which subword state of the current word $\texttt{AF}^i$ is currently in; in our case, subword states correspond to phones. $\texttt{Trans}^i$ is a binary variable which is 1 if the current frame is the last one in the current subword state for $\texttt{AF}^i$ and 0 otherwise. Thus, when $\texttt{Trans}^i = 1$ and $\texttt{SubwordState}^i = 3$ for frame t, then $\texttt{SubwordState}^i = 4$ for frame $t + 1$.

The model allows for asynchrony between the AFs, to allow for some types of pronunciation variation. For example, we can have `SubwordState`^{2} > `SubwordState`^{3} for a frame, meaning that the tongue features "get ahead of" the glottis and velum by moving on to the value for the next phone earlier—as we saw was the case for the pronunciation variant `[d ow n t]` → `[d ow_n t]` (cf. Figure 1.1). However, inference becomes intractable without any constraints on this asynchrony. The mentioned limiting of the asynchrony between any pair of L, T and G to one subword state is enforced by the factor connecting the variable `FtrAsyncConfig` to the three `SubwordState`^{i} variables.

The blue squares in Figure 4.1 represent *deterministic factors*. These are binary functions whose purpose is to ensure that invalid values of the connected hidden variables are assigned zero probability. In addition to the one already mentioned that enforces the asynchrony constraints, there are three deterministic factors for each frame and three for each transition (one for each of the AFs); the former ensures that `AF`^{1} has the correct value for the current position of the current word, and the latter ensures that the subword state indices increment by at most 1 from one frame to the next.

The *trainable factors* in the CRF are represented by red squares in the figure. There are seven trainable factors in each frame template: two for each of the AFs (one relating to the transitioning between AF values, and one modeling the relationship between AF values and observation vectors) plus one for the asynchrony between them. Each of these trainable factors has a vector of weights associated with it, which is learned during the training of the CRF.

For details about the implementation of the CRF and its inference, we refer to the original paper on the forced transcription model [Prabhavalkar et al., 2011].

4.2.3 Data Description

We use the Switchboard Transcription Project (STP) corpus described in Section 3.2. For these experiments we follow Prabhavalkar et al. [2011] in using identical data sets to those used in Livescu's lexical access experiments [Livescu, 2005, Ch. 4]. This setup differs from the one we used for our frame classification experiments (cf. Section 3.2) in that some words are excluded, and in that we do not use a cross-validation scheme.

The vocabulary is restricted to the 3500 most common words in the training subset of the STP data, excluding partial words, words whose transcriptions contain non-speech noise and words for which the baseform pronunciation was missing. Words whose baseform pronunciations are four phones or shorter (where stops, affricates and diphthongs are considered two phones each) are also excluded. This length restriction is intended to exclude words that are so short that most of their pronunciation variation is caused by neighboring words.

The data is split into a training set, a development set and a test set. The training set consists of 2941 word tokens encompassing 89 748 frames (15.0 minutes of speech); the development set has 165 word tokens and 5365 frames (0.9 minutes); and the test set consists of 236 word tokens and 7037 frames (1.2 minutes).

The acoustic observation vectors in these experiments are 39-dimensional perceptual linear predictive coefficients (PLPs). These are computed in the same way as described on page 67, except that when they are concatenated in 9-frame windows, all the derivatives are included in the concatenation, so the resulting observation vectors have dimensionality 351.

4.2.4 Experimental Setup

Our experiments with the CRF for forced transcription focus on the observation vectors ($\mathbf{x}$ in Figure 4.1). The system is unchanged except for these input features, which are generated by the frame classification outputs of either MLPs or k-NN classifiers.

MLP Baseline For our baseline, we follow Prabhavalkar et al. [2011] in using observation vectors consisting of log-transformed posterior probabilities from MLP frame classifiers. There are four MLPs, three for classifying each of the three articulatory features L, T and G, and one for classifying phones. The observation vectors have one dimension for each of the possible values of each of these, $8 + 25 + 4 + 98 = 135$ dimensions in total. The MLPs all have a single hidden layer and a softmax activation function on the output layer, with the number of nodes in the hidden layer determined by tuning on the development set.

k-NN Alternative When replacing the MLPs with k-NN classifiers, we use the feature functions presented in Section 4.1 to construct posterior-like k-NN features. In order to match the conditions for the MLPs, the k-NNs classify, in addition to phones, the L, T and G labels directly (even though the computational overhead for classifying all the eight AFs used in Chapter 3 would be very small for k-NNs). We use the same PLP vectors as input to the MLP classifiers. We normalize these by subtracting the global mean and dividing by the sample standard deviation, but do not apply any

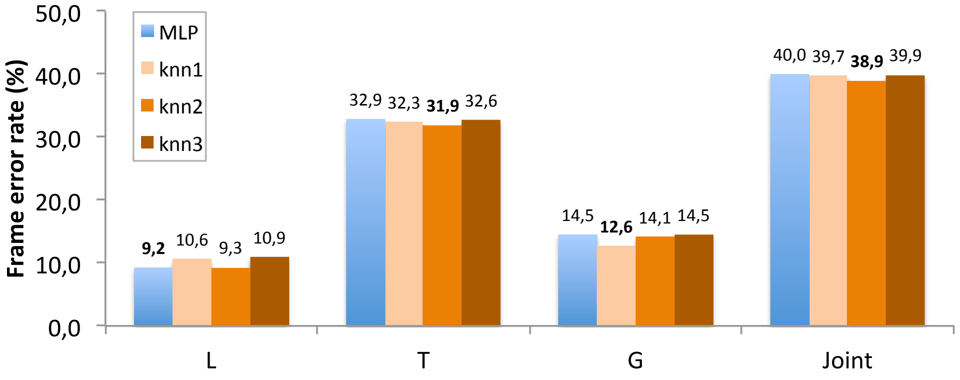


Figure 4.2: Frame error rates (in %) of the CRF for forced transcription of AFs with MLP-based and k-NN-based input features. The AFs are *Lips* (L), *Tongue* (T) and *Glottis/Velum* (G); *Joint* gives the percentage of frames where at least one of the three AFs is incorrectly transcribed.

other transformations to them. The number of nearest neighbors k is tuned on the development set; we found the optimal values to vary from 80 to 200.

4.2.5 Results

Figure 4.2 shows the frame error rates of the forced transcription CRF when classifying each of the articulatory feature streams L, T and G separately, and when classifying all three jointly. Results are shown for the MLP features used in the original system (with 9-frame PLP vectors) as well as for the three different k-NN features presented in Section 4.1: `knn1`, `knn2` and `knn3`. The reported results with k-NN features are all based on 1-frame PLP vectors, as they did consistently better for these experiments (9-frame vectors gave a 2% absolute increase in error rates on average).

We see that the performance of the different k-NN feature functions varies somewhat between the tasks, with `knn1` being the best performer for G and `knn2` giving better performance for L and T as well as for Joint. We also see that the performance of the `knn2` features is slightly better than that of the MLP features for all tasks except for L, for which the performances are on par.

Table 4.1 compares the performance of the CRF for the MLP features and the best k-NN features, and gives the the p-values of a MAPSSWE test for each pair.[1] We see that the improvement of the k-NN features for the

[1]Because the forced transcription CRF models dependencies between frames, the independence assumptions of McNemar's test do not hold here.

Table 4.1: MLP Features vs. k-NN Features in CRF
Comparison of the performance of the CRF for forced transcription with MLP features and k-NN features. The best k-NN features are chosen for each task: `knn1` for G, `knn2` for the others.

AF	**Frame Error Rate (%)**			**MAPSSWE**
	MLP feats	k-NN feats	Difference	p-**value**
L	9.2	9.3	−0.1	0.8
T	32.9	31.9	1.0	0.07
G	14.5	12.6	1.9	0.009
Joint	40.0	38.9	1.1	0.06

articulatory feature G is strongly statistically significant. For the other two tasks where the k-NN features give better results than the MLP features, the improvements are almost but not quite significant at the 0.05 level. For the one task where the MLP features give the best performance, the p-value is so high that the difference can be considered spurious.

4.2.6 Discussion

To begin with, we should note that the same problem applies to these results as to the AF results reported in Chapter 3: Our ground-truth AF labels, being derived from the phonetic labeling, are inherently unreliable, particularly around phone boundaries. This adds a measure of uncertainty to the results. However, we can reasonably assume that this will impact the CRFs with different observation vectors randomly, and not give systematic advantages to any of them.

The improvement of the k-NN features over the MLP features is small. However, the main motivation for using k-NNs rather than MLPs here is to improve the flexibility of the system without deteriorating its results. Any improvement can thus be considered a bonus.

As for the difference between the performances of the various k-NN features, we see that rank-weighting improves the results (i.e., `knn2` outperforms `knn1`) for three out of the four tasks. This improvement is to be expected. Why it does not occur for the G task is not clear. It might have to do with the fact that G has the lowest cardinality of the three AFs we consider in these experiments (4, as opposed to 8 for L and 25 for T). It makes some sense, intuitively, that rank-weighting would be more important in a situation with more classes, since the prior class has a far lower chance of being correct. It is also interesting to note that this is the task where the improvement of the k-NN features over the MLP features is decidedly strongest. This raises the question of whether it might be beneficial to use a higher number of articulatory features with lower cardinalities. As mentioned, the computational overhead of doing this would be minimal, since k-NN s require no training and can use the same neighbor search for several classifications based on different labels.

The distance-weighted k-NN features (`knn3`), however, perform poorly, consistently getting the highest error rate. It is not clear why this is the case, but it does indicate that a neighbor's rank among the other neighbors is a more useful source of information for the CRF than the neighbor's distance from the query point relative to those of the other neighbors.

As mentioned in Section 4.1, the rank-weighted k-NN features, `knn2`, performed better with a normalization factor of $\frac{1}{k}$ than with the more correct $\frac{2}{k(k+1)}$. The latter would make all the features fall between 0 and 1, like a probability; for the former the feature values range from 0 to 250. It is not obvious why this choice of normalization factor would be beneficial—it has no effect on the magnitudes of the features relative to one another. As we saw in Section 4.2.2, there are several trainable factors in the CRF for which the weights are learned, some of which incorporate the k-NN or MLP features. Using higher feature values could be seen as a form of prior weighting, from which the CRF seems to benefit slightly in this case.

The k-NN features gave the best performance in the CRF when they were based on 1-frame rather than 9-frame vectors. Initial experiments showed the opposite to be the case for the MLP features. When we consider the frame classification results from Chapter 3, this is unsurprising. For the MLP, 9-frame features clearly outperformed 1-frame features. For k-NN classification with normalized features, however, the mean AF error was lower for 1-frame features. This difference was very small in the frame classification case; the forced transcription is also evaluated on the frame level, but the fact that the CRF can model context might have made the difference bigger here.

Future Work

The poor performance of the distance-weighted k-NN features raises the question of whether another distance weighting than the one we used might give better results. We used inverse Euclidean distance, $\mathcal{D}^{-1}$, to make the `knn3` features (cf. Equation (4.3)); a natural alternative might be to vary the exponent, trying for instance $\mathcal{D}^{-\frac{1}{2}}$. Similarly, we could try different rank weightings, hoping to improve further upon the results with the `knn2` features. Currently, the weight of a neighbor grows linearly with its rank among the k nearest ones. Exponential or sigmoidal growth could be interesting alternatives here.

We have seen that the computational overhead of doing several k-NN classifications with different AF sets is relatively small. We have also seen that CRFs, since they do not need to model dependencies between the input variables, allow for rich, overlapping features. It would therefore be interesting to try to create k-NN features based on more than one AF labeling and input all of them to the CRF. While input features based on frame classification of the same AF set that we are doing forced transcription of are likely to be the most helpful, input features based on other AF labelings might add useful information. A set of binary AFs, for instance based on SPE [Chomsky and Halle, 1968], would be a promising alternative, as they often give high classification accuracies and might provide different information than the articulatory phonology-based AFs we are currently using.

It would also be interesting to experiment with adding meta-information to the CRF, similarly to Demuynck et al. [2011b] (cf. page 32). Their task and CRF model were different from ours, but several of their meta-information features might prove useful in our system—e.g., the duration of the phone utterance that an exemplar frame is drawn from might tell us something about the likelihood of pronunciation variation, as could the position of the exemplar frame in the word. For our application, other meta-information could also be added, like features based on the values of all the AFs for each exemplar frame.

4.3 Phone Classification and Recognition on TIMIT

In this section we will consider how we can use our k-NN features in a tandem hidden Markov model (HMM) like those described in Section 2.2. The system we will consider is a phone recognizer on the TIMIT corpus. The task is to transcribe the phones uttered, given the audio of an utterance. Unlike the frame classifiers, the HMM recognizer can take context into account through language modeling.

4.3.1 Data Description

The TIMIT Acoustic-Phonetic Continuous Speech Corpus [Garofolo et al., 1993] is a widely used database for ASR systems. It contains recordings of

630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences.

The standard training set for TIMIT contains 3696 utterances consisting of 3.12 hours of speech. For our experiments we have split this into a smaller training set of 3320 utterances and a development set of 376 utterances that is balanced across the dialect regions. The standard full test set of 1344 utterances (1.14 hours) is used.

Morris et al. [2008] report phone recognition results on TIMIT for an MLP tandem system; for comparison purposes we try to replicate much of their experimental setup. We use 39-dimensional PLPs for our acoustic observation vectors, extracted in the same way as described on page 67, except this time computed for a slightly longer window—25 rather than 20 ms. We try both 1-frame and 9-frame windows, where each observation vector is concatenated with its 4 preceding and 4 succeeding observation vectors. All derivatives are included in this concatenation, resulting in 351-dimensional observation vectors.

We follow Lee and Hon [1989] in using a set of 48 distinct phones for HMM modeling and mapping down to 39 phones for HMM evaluation. For the frame classification, the full set of 61 phones is used for both modeling and evaluation.

In our frame classification experiments on the STP data presented in Chapter 3, we excluded the frames that were labeled as silence, since they were not relevant to the applications considered. In a phone recognition context, however, silences must be included. Thus, the data we use for the experiments presented in this chapter, including those on frame classification, include silence frames.

4.3.2 Frame Classification

The focus of this section is on tandem systems for phone recognition. However, as we saw in Section 2.2, frame classification forms part of the basis of any tandem system. We will therefore report our phonetic frame classification results for TIMIT here.

MLP Baseline

The setup of our MLP phonetic frame classifier for TIMIT is mostly the same as the one we used for the STP corpus, described in Section 3.3. It is implemented in QuickNet, with one hidden layer in which the number of hidden units is tuned over the values 1000, 3000, 5000, 7000 and 10 000. Bunch size proved inconsequential in initial experiments and is mostly kept at the default of 16. The learning rate is tuned on a grid from 0.001 to 0.256 in powers of 2.

With 9-frame observation vectors this gives an error rate for the frame classification of phones just shy of 35%, as we can see from Table 4.2. Seemingly, this is substantially better than the error rate of almost 64% we reported for the MLP phonetic frame classifier on the STP corpus (cf. Table 3.3). There are several things to note here, however. First, the phone set that is being classified in this case is smaller (48 for TIMIT vs. 98 for STP) and thus has less confusability. Second, TIMIT is a substantially larger data set (the number of frames is seven times higher than for our STP data) and more training data generally gives a better classifier. Third, the conversational speaking style of STP makes for a harder phonetic classification task than the read speech in TIMIT. Fourth, STP was recorded over telephone lines, with the resulting distortions and lower bandwidth. And finally, frames labeled as silence are included in our TIMIT data, whereas we excluded them from our STP data. Because silence is the most frequent phonetic label in most speech corpora that include it, its inclusion tends to reduce frame error rates, possibly because the sheer number of silence frames makes this a good back-off hypothesis for the classifier when it is unsure of its decision. This is also reflected in the higher performance of the chance classifier on TIMIT: 88.1% error rate,[2] compared to 95.3% on STP. In light of this, the difference in MLP error rates is unsurprising.

k-NN Alternative

The k-NN setup is slightly different in these experiments from that described in Sections 3.4 and 4.2.4, partly because TIMIT is a larger data set than STP, and partly because silences are now included.

The size of TIMIT has made it necessary to parallelize the k-NN classification, but we are still using exact search and the core algorithm is the same as before. Since the bulk of the computational cost of the k-NN algorithm is finding the nearest neighbors of each of the data points in the test set, it is straightforward to divide the computation into arbitrarily small parts. Based on the capabilities of our available computing cluster, we chose to do the neighbor search of our data set[3] as 100 parallel tasks.

[2]Since the chance classifier always chooses the most frequent label and silence is the most frequent label, its error rate when classifying phones on TIMIT equals the proportion of frames that are not labeled as silence.

[3]For the frame classification task itself, the nearest neighbors need to be found only for the development and test sets, but for the k-NN tandem experiments described in

As can be seen from Table 4.2, the phonetic frame classification results for the k-NN classifier are substantially better on our TIMIT data (around 45% error) than they were on our STP data (around 70% error—cf. Table 3.14), as was also the case for the MLP classifier. However, the gap between the two classifiers has increased: On TIMIT, k-NN is doing worse compared to the MLP baseline than was the case on STP. This is true for both 1- and 9-frame windows.[4]

When using the same kind of normalization on the PLP acoustic observation vectors as we did on the MFCC vectors in our STP experiments, so that each dimension of the training data has mean zero and unit standard deviation (cf. page 66), the performance of the k-NN classifier is improved here as well—the relative decreases in error rate are 11.7% and 1.1% for 1-frame and 9-frame windows, respectively.

The number of nearest neighbors, k, is tuned on a grid of all odd values from 1 to 1001, but the optimum is below 60 in all of our experiments. Figure 4.3a shows a plot for the case with 9-frame normalized features.

Silence Reduction We mentioned that the inclusion of silence frames in our TIMIT data might contribute to the better frame classification performance compared to STP, since silence is by far the most frequent phone label in the data. However, this might also result in a tendency to classify many non-silence frames as silence. Confusability is inherent to any classifier, but when one label comes to dominate the prior distribution to a large extent, it becomes particularly problematic. There is also reason to think that silence is confusable with a larger set of other phonetic labels than other phones, since it has no particular characteristics except an absence of sound. The k-NN classifier might also be more vulnerable to this than a parametric classifier, since its decisions are taken by majority vote among the nearest neighbors. We saw an example of how the classes with the highest prior probabilities could dominate in our STP experiments (cf. page 83).

To counteract this effect, we experiment with reducing the amount of silence in the training set. Deselaers et al. [2007] experimented with an energy-dependent silence reduction, where only silence frames with energy above a certain threshold was discarded (the idea being that these would be

Section 4.3.5, we also need to find the nearest neighbors of the data points in the training set. Thus it is the entire data set that we have divided into 100 parts.

[4]The relative increases in error rate for k-NN vs. MLP frame classification with unnormalized observation vectors were 17.1% for 1-frame and 19.7% for 9-frame windows on STP. The corresponding numbers on TIMIT are 34.7% and 35.6%.

Table 4.2: Phonetic Frame Classification Results on TIMIT
We tune the parameter settings on the development set. Test set performance is reported only for the best settings for each classifier.

Chance Classifier

Frame Error Rate (%)	
Dev Set	Test Set
87.6	88.1

Multilayer Perceptron

Window Width	Hidden Units	Bunch Size	Learning Rate	**Frame Error Rate (%)** Dev Set	Test Set
1	10 000	16	0.064	39.9	-
9	7 000	16	0.008	32.7	34.9

***k*-Nearest Neighbors**

Window Width	Normalized	% Silence Removed	k	**Frame Error Rate (%)** Dev Set	Test Set
1	No	0	59	55.9	-
9	No	0	29	45.5	-
1	Yes	0	49	48.3	-
9	Yes	0	33	44.1	-
9	Yes	75	35	43.5	45.9

the most confusable with non-silence frames), but found that randomized silence reduction gave better results. Therefore, we randomly select a chosen percentage of the frames in the training set labeled as silence, and exclude these from the search for the nearest neighbors of any given frame. As we can see from Figure 4.3b, silence reduction up to 75% consistently improves the classifier's performance. However, by comparing to Figure 4.3a, we see that this improvement is relatively insubstantial compared to the impact of the choice of the number of nearest neighbors, k.[5]

[5] k is tuned separately for each of the six levels of silence reduction, and ranges from 33 to 49.

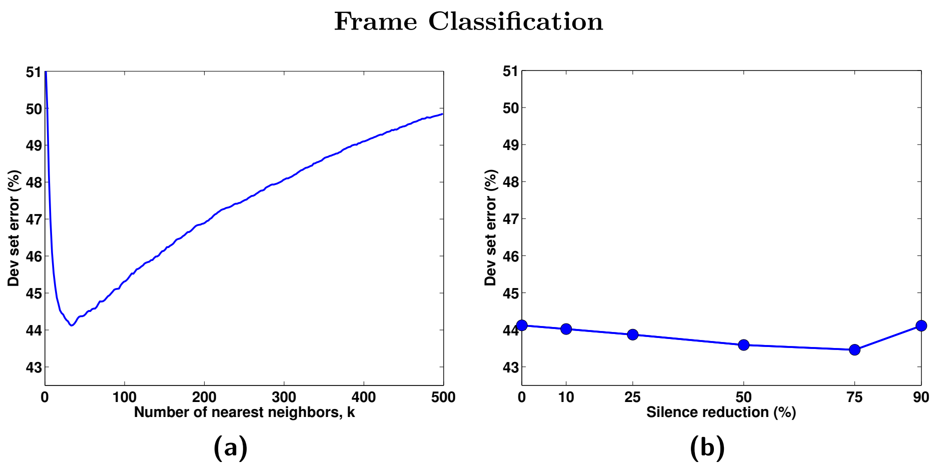


Figure 4.3: Plots of the effects of the number of nearest neighbors, k, and of the amount of silence removed from the training set, on the frame classifier's performance. In (a) the silence reduction is 0% and the plotted line shows the results for every odd value of k; in (b) k was tuned separately for each experiment, the results are marked by the circles and the line is an interpolation.

4.3.3 Hidden Markov Model-Based Recognizer

Moving on to the phone recognition task, we present in this section the hidden Markov models that are used in all of the following experimental setups: on their own, with PLP vectors as input; as part of the MLP tandem system (Section 4.3.4); as part of the phone-based k-NN tandem system (Section 4.3.5); and as part of the articulatory k-NN tandem system (Section 4.3.6). The HMMs we use are implemented with the Hidden Markov Model Toolkit (HTK) [Young et al., 2009] and based on a script from Cantab Research in Cambridge, UK.

We find that TIMIT is too small to reliably tie states for context-dependent phones,[6] and since context-dependent phone modeling is also far costlier computationally, we have chosen to use context-independent phones only for these experiments. We use up to 40 distributions in the Gaussian mixture model, tuning the number on the development set. Tuning of the word insertion penalty and the grammar scale factor (the `-p` and `-s` options of the `HVite` program of HTK) did not have a major impact on performance, and are left at their default values of 0 and 5, respectively. We use HTK's

[6] Morris et al. [2008] achieves an absolute improvement in error of 1% with context-dependent phones compared to context-independent phones, but in our experiments we have not been able to reproduce this improvement.

Table 4.3: Baseline Phone Recognition Results
With PLP acoustic observation vectors and MLP tandem features.

Features	Gaussians	Phone Error Rate (%)	
		Dev Set	Test Set
PLP	36	32.1	32.9
MLP tandem	38	29.9	31.3

standard back-off bigram language model with discounting (refer to Young et al. [2009, pp. 188–189] for details), trained on the transcriptions of the training set.

Results for phone recognition with PLPs as observation vectors are shown in the first line of Table 4.3.

4.3.4 MLP Tandem System

The MLP tandem system combines the MLP frame classifier and the baseline HMM in the manner outlined in Section 2.2. Instead of PLP acoustic vectors, we use outputs from the MLP frame classifier as input observations to the HMM. We use the MLP parameter settings that gave the best results on the frame classification task: 7000 hidden units, a learning rate of 0.008 and a bunch size of 16. Following Morris et al. [2008], we use linear outputs from the neural networks, which we then decorrelate using PCA. The results are shown in Table 4.3. These are a slight but statistically significant improvement over the PLP HMM ($p < 10^{-4}$), and they are on par with the context-independent phone results reported by Morris et al. [2008].

4.3.5 Phone-Based k-NN Tandem System

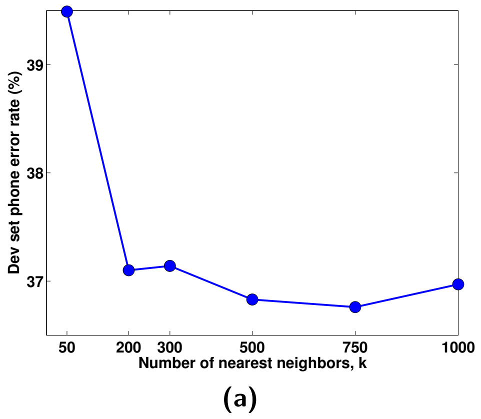
To create a tandem system using k-NN instead of MLP as our frame classifier, we again use the feature functions presented in Section 4.1 to create feature vectors based on the k-NN frame classification results. We input these as the observation vectors in the GMM-HMM phone recognizer, in the same way as the outputs from MLPs are used in a traditional tandem system, as described above. To the best of our knowledge, this work represents the first attempt to incorporate k-NNs in a tandem HMM system.

This work differs from previous work with k-NN in ASR in various ways. It is separated from the work of Golipour and O'Shaughnessy [2009; 2010; 2012], described on page 30, chiefly in that our k-NN classification is frame-based while theirs is segment-based. Classifying varying-length segments of the acoustic signal known to constitute single phones is a very different task from classifying short, fixed-length segments that constitute just a small part of the phone. Their lattice rescoring work [Golipour and O'Shaughnessy, 2010] is the most similar to ours. In both cases k-NN classification is used in an attempt to improve the performance of a GMM-HMM phone recognizer, but while they use their k-NN classifier on top of a GMM-HMM system, we use a GMM-HMM system on top of a k-NN classifier.

The work of Deselaers et al. [2007], described on page 29, has more similarities to the work we present here. They also use a k-NN frame classifier to attempt to improve upon the recognition accuracy of a GMM-HMM system. However, while we make feature vectors from the k-NN as input to the GMMs, Deselaers et al. use the k-NN to make probability estimates to replace the GMMs. Arguably, our system is analogous to a tandem HMM, while theirs is more like a hybrid HMM. It should also be noted that their probability estimates are based on something more like a volumetric k-NN [Fukunaga, 1990] than a voting k-NN in that, rather than counting how many of the k nearest neighbors of a point belong to each class, they consider the k nearest neighbors of the same class as the query point. We also note that in their probability estimate $\hat{p}_{k\text{-NN}}$, Equation (2.24), the exponent might be seen as analogous to our feature function `knn1`, Eq. (4.1). There is no such clear correspondence between $\hat{p}_{\text{kern}}$, Eq. (2.25), and our `knn2` or `knn3`, however—their scaling factor is constant whereas we do a weighting based on rank or distance of the neighbors.

We tuned the values of the parameters for the amount of silence reduction (cf. page 104) and the number of nearest neighbors used, k, directly on the phone recognizer performance on the development set. Their impact is shown in Figures 4.4a and 4.4b. We see that the optimum for the amount of silence removed is the same as it was for frame classification. The optimal k for phone recognition, on the other hand, is an order of magnitude higher than it was for frame classification.

That $k = 750$ gives the best phone recognition accuracy might seem surprising in relation to the marked deterioration in frame classifier performance for high ks shown in Figure 4.3a. We need to consider, however, that the nearest neighbors are being used quite differently in these two cases. The k-NN frame classifier uses the nearest neighbors only to make a single choice, based on majority vote, for the most likely class. In the k-NN tandem phone recognizer, a Gaussian mixture model is fitted to the features generated from the nearest neighbors. The k-NN tandem system then, can benefit from looking at the entire distribution of the nearest neighbors' labels, which may explain why it benefits from a much larger value of k.



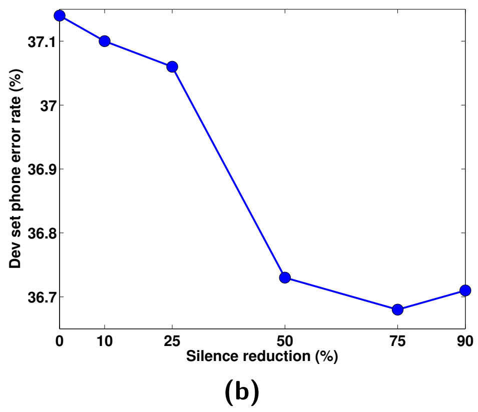


Figure 4.4: Tuning of the number of nearest neighbors (a) and the percentage of silence removed from the training set (b) on the phone recognizer performance.

Table 4.4: Phone-Based k-NN Tandem
Phone recognition results with three kinds of k-NN features as observation vectors, $k = 500$ and 50% silence reduction.

Feature Type	Phone Error Rate, Dev Set (%)
knn1	37.61
knn2	36.66
knn3	37.56

We presented three feature functions to generate our "posterior-like" k-NN features in Section 4.1: `knn1`, `knn2` and `knn3`. Table 4.4 shows the results for the three kinds of features. We see that the rank-weighted features, `knn2`, give the best performance, but that all three kinds are well below both baselines. Note also that the distance-weighted features, `knn3`, do better than the proportional features, `knn1`, for this task—unlike what was the case in the forced transcription CRF of Section 4.2.

Exclusion of Adjacent Frames One of our hypotheses for what might be causing these disappointing results is related to what happens when the nearest neighbors of a frame are drawn from the same part of the data set as the frame itself—i.e., when the so-called query set and search set are the same. This is the case when we make k-NN features for the training set.

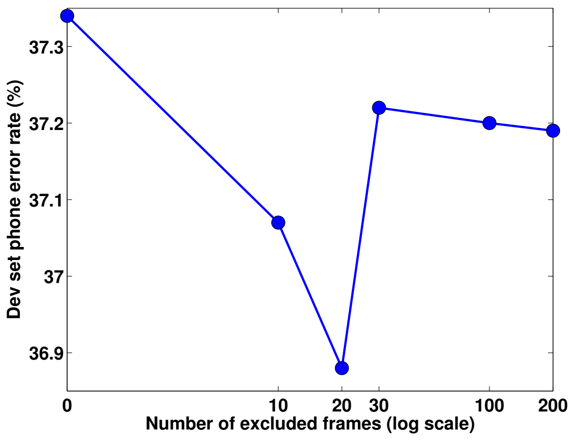


Figure 4.5: Tuning of the number of excluded frames on the phone recognizer performance.

When doing k-NN classification of the training set, a frame is not allowed to be its own neighbor. However, the frames immediately preceding and succeeding the current frame typically are. Since phone utterances in TIMIT have a mean duration of approximately 80 ms, 8 consecutive frames in the training set will often have the same label. These frames will tend to have very similar acoustic observation vectors, and thus be one another's nearest neighbors. This can make the HMM too confident that the k-NN classifications will be correct, since the labels of the very nearest neighbors in such cases will be a very strong predictor for the label of the query frame—far stronger than if the query frame is drawn from the test set. We therefore experiment with excluding a number of the adjacent frames from the neighbor search, to put the training and test sets on a more equal footing.

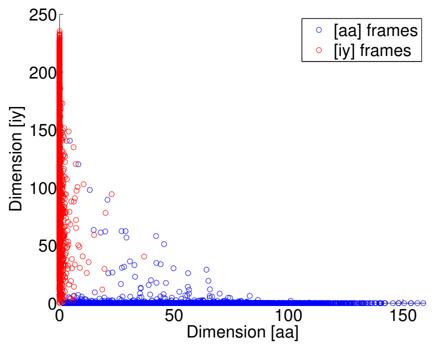
Excluding a number of the adjacent frames consistently improves the accuracy. However, as Figure 4.5 shows, the improvement, as well as the impact of the number of excluded frames chosen, is relatively small.

Log and PCA Transformations Another hypothesis for the cause of the poor results of the k-NN tandem system is that the k-NN features are hard to model with Gaussian mixtures, which are at the heart of the phone recognition HMM. Our 61-dimensional k-NN feature vectors are quite sparse. In the feature vector for a given frame, a dimension corresponding to a given phone will have a non-zero value only if one of the k nearest neighbors of the frame has that label. Figure 4.6a shows a scatter plot of the dimensions corresponding to the phones [aa] and [iy] for the feature vectors of all frames in the development set with one of these labels. As is to be expected, the features cluster along the axes (since a lot of the nearest neighbors of each frame will have the correct label, while few or none of them will have a neighbor labeled with a phone as different from [aa] as [iy] is), and we can see how these two dimensions might prove hard to model with a Gaussian mixture.

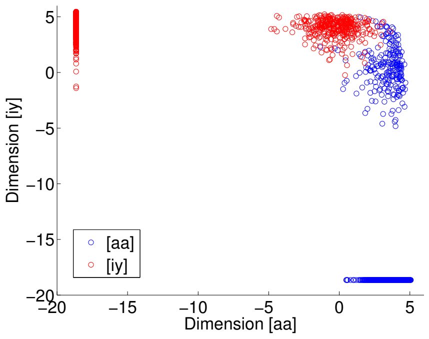
For this reason, doing a log transformation of the features seems like a natural choice. As noted (cf. page 24), this is also common in MLP tandem systems when using the posterior probability outputs rather than the linear outputs. Before we can log transform our vectors we must deal with the feature values that equal zero (since $\log 0 = -\infty$). Our initial solution was to simply replace all zeros with a fixed small value (around 10^{-8}) before taking the natural logarithm. This gave very poor results: the phone error rate increased from 37% to 47%. Figure 4.6b, which shows the [aa] and [iy] frames of the development set after this initial log transformation, gives us an idea of why the transformation was suboptimal: we get the clustering along the lines at $\ln(10^{-8}) = -18.4$ instead of along the axes, which is still arguably hard to fit a Gaussian mixture to. It is not obvious, however, why the results for these log features are worse than for the raw features, as it seems like it would be far easier to fit a Gaussian mixture to the features in Figure 4.6b than to those in Figure 4.6a.

We improved the results with log-transformed features by replacing the zeros with Gaussian noise, in a similar manner to Sun et al. [2012]. Some exploratory tuning of the parameters of the Gaussian noise indicated that a standard deviation of 1 and a mean that was two standard deviations below the minimum non-zero feature value gave the best results. Figure 4.7 shows what the distributions of values look like for knn2 features that have been log-transformed in this way. We see that there is a clear separation between the Gaussian noise and the rest of the log-transformed feature values, and that the latter also almost follow a Gaussian bell curve. In Figure 4.6c we see how the [aa] and [iy] frames of the development set are spread in the corresponding two dimensions after this modified log transformation. Comparing this to Figures 4.6a and 4.6b, we see that the log transformation with Gaussian noise retains discriminative power between [aa] and [iy] in their corresponding dimensions while appearing more amenable to Gaussian mixture modeling.

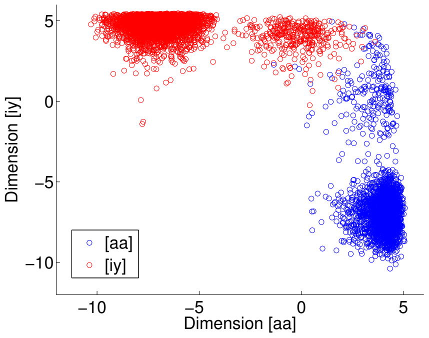
We also tried a standard PCA transformation on the features (without reducing their dimensionality). The second and third dimensions of the



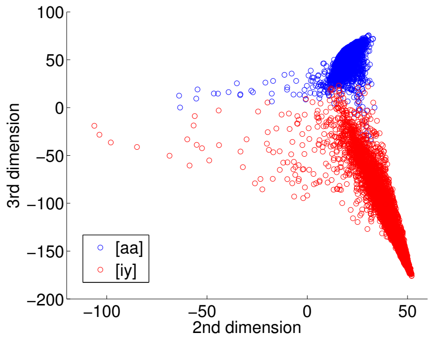
(a) Raw features, dimensions corresponding to [aa] and [iy].



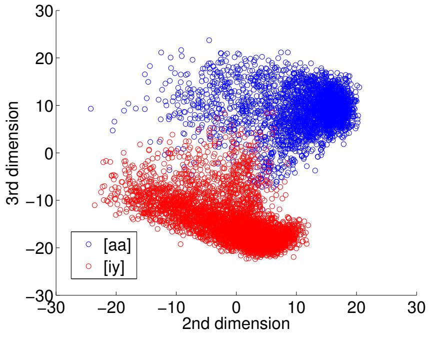
(b) Log-transformed features (fixed zeros), dimensions corresponding to [aa] and [iy].



(c) Log-transformed features (Gaussian zeros), dimensions corresponding to [aa] and [iy].



(d) PCA-transformed features, dimensions 2 and 3.



(e) Log- and PCA-transformed features (Gaussian zeros), dimensions 2 and 3.

Figure 4.6: Two dimensions of the k-NN feature vectors of all frames in the development set labeled [aa] and [iy], with and without log and PCA transformations. In Figure (b), zeros have been replaced by a fixed value before the log transformation; in Figures (c) and (e), Gaussian noise is used instead.

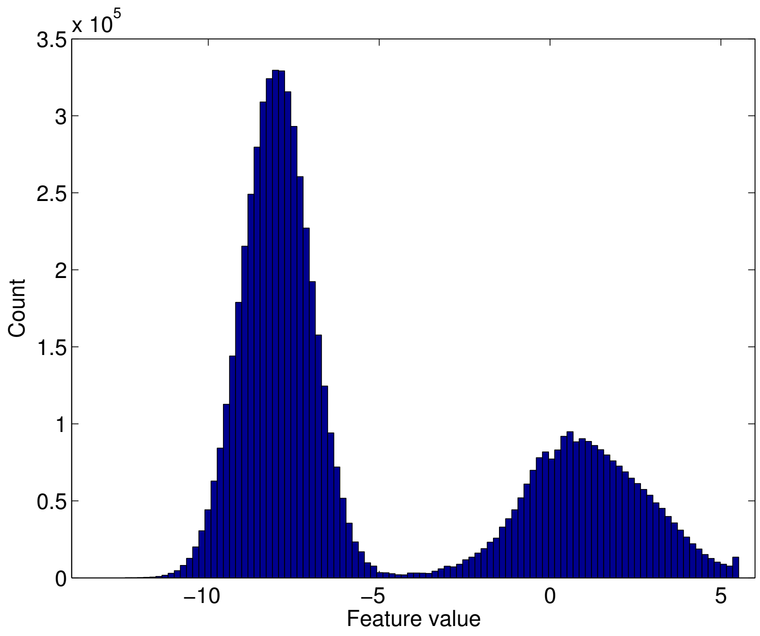


Figure 4.7: Histogram of the rank-weighted k-NN feature values (knn2) of the development set (across all dimensions), after log-transformation. Before taking the natural logarithm, all zeros were replaced with Gaussian noise with mean -8 and standard deviation 1.

resulting features[7] for the [aa] and [iy] frames of the development set are shown in Figures 4.6d and 4.6e—the former without, the latter with log transformation. Again we see that the log transformation seems to retain discriminative power while making GMM modeling easier.

As we can see from Table 4.5, however, the log transformation with Gaussian zeros also causes a deterioration in results, both with and without a PCA transformation. The deterioration is far smaller than it was for the fixed zero variant of the log transformation, but for the PCA-transformed features it is still substantial. The best results are achieved with PCA-transformed features with no log transformation, but the improvement that the PCA transformation gives over raw PLP features is marginal.

[7]We chose the second and third because the top PCA dimension is not very discriminative between these two phones, probably because it concentrates on separating the silence from the non-silence.

Table 4.5: k-NN Tandem With and Without Log
Phone recognition results with `knn2` features as observation vectors, $k = 500$ and 0% silence reduction Before taking the natural logarithm, all feature values equal to zero are replaced by Gaussian noise with mean -8 and standard deviation 1.

Log	PCA	Phone Error Rate, Dev Set (%)
No	No	37.0
Yes	No	37.9
No	Yes	36.8
Yes	Yes	40.0

4.3.6 Articulatory k-NN Tandem System

We have seen how the exclusion of adjacent frames and a PCA transformation improves the performance of the k-NN tandem system somewhat, but the phone error rate on the development set remains, at 36.8%, well above both the baselines, at 32.1% and 29.9%. We will now consider whether adding articulatory information to the k-NN classifiers can close this gap.

For a k-NN classifier, doing several classifications of the same data set based on different labelings has very little computational overhead, as previously mentioned. Our plan in this section is to do frame classification of AFs for the TIMIT data like we did for the STP data, and then make new k-NN features based on this classification. The nearest neighbors are the same, but the labels are different, and since the k-NN features for the tandem system are based on how many of the nearest neighbors belong to each label, the features will be different. The articulatory labeling can therefore supply an extra level of information to the system.

Table 4.6 shows the k-NN frame classification results for our eight articulatory features. Comparing this to Table 3.13, which shows our best k-NN frame classification results on STP, we see that the results are substantially better on TIMIT for all AFs except for GLOT. This is the same level of improvement as we saw for the phone classification, and may have some of the same causes—the larger data set and the inclusion of silence are both beneficial. The smaller inventory of phones does not help us for this task, however. For the phone classification task this gave us a smaller number of classes, which made for an easier classification task. For the AF classification the number of classes is the same, and since we derive the AF labels

Table 4.6: AF Frame Classification on TIMIT using k-NN
The amount of silence reduction and the number of nearest neighbors, k, was tuned separately for each AF on the development set.

	% Silence Removed	k	Frame Error Rate, Dev Set (%)
LIP-LOC	75	33	12.6
LIP-OPEN	90	27	9.9
TT-LOC	75	17	16.7
TT-OPEN	75	29	26.9
TB-LOC	90	41	23.6
TB-OPEN	75	41	27.8
VEL	90	15	3.2
GLOT	75	13	10.8

from the phonetic labeling,[8] the AF labeling arguably gets less accurate when the phone labels are fewer, since it is based on less detail; and a less accurate labeling makes for a harder classification task. On the other hand, the fact that TIMIT is manuscript-read rather than spontaneous speech is well known to make recognition tasks easier, and might make frame classification easier as well.

Our first attempt is to make k-NN features based solely on AF labels instead of on phone labels. Unlike the CRF experiments in Section 4.2, all eight AFs are used; this gives a dimensionality of 32 (the sum of the cardinalities of the AFs) compared to the 61 dimensions for the phonetic k-NN features. The results of this experiment are given on the third line of Table 4.7, and we see that they are slightly worse than the earlier phonetic k-NN tandem results (included on line two of the same table, for ease of comparison). We then try to make feature vectors based on both phone and AF labels of the nearest neighbors, giving a 93 dimensional vector (the earlier 32 plus 61 for each possible phone label). At 37.5%, this gives a lower phone error rate than using only AF labels (38.8%), but higher than using only phone labels (36.7%).

Since we are still doing worse than the PLP baseline (included in the first line of Table 4.7) it seems natural to try to augment the PLPs rather than replace them. When we append PLPs to the earlier k-NN features, we get

[8]See Table A.5 in the appendix for the details of this mapping.

Table 4.7: Articulatory k-NN Tandem Results
Error rates (in %) for the k-NN tandem phone recognizer with k-NN features based on phones and/or articulatory features. The first line gives the PLP baseline results.

PLPs Appended	Basis for k-NN Features	PCA	Feature Dimensionality	Phone Error Rate, Dev Set (%)
yes	-	no	39	32.1
no	phones	no	61	36.7
no	AFs	no	32	38.8
no	phones & AFs	no	93	37.5
yes	phones	no	100	33.4
yes	AFs	no	71	34.5
yes	phones & AFs	no	132	35.9
yes	phones	yes (3D)	42	32.5
yes	AFs	yes (3D)	42	32.2
yes	phones & AFs	yes (3D)	42	31.8

an improvement in accuracy, but it is still below the PLP baseline, whether we use phones, AFs or both (lines 5, 6 and 7, respectively, in Table 4.7).

The feature vectors that give the best result have 100 dimensions. In case the high dimensionality might be part of the problem, we reduce the dimensionality of the k-NN features with PCA. We tried adding the top 1, 3, 5, 7 and 10 PCA dimensions to the PLPs, and achieved the best performance with 3 dimensions. Adding these three-dimensional k-NN feature vectors to the PLPs hurts performance slightly when the k-NN features are based on only phones or only AFs, but when the we take the 93-dimensional k-NN feature vectors based on both phone and AF labels and reduce their dimensionality to 3 with PCA before appending them to the PLPs, we do achieve a slight improvement in accuracy on the development set. As we can see from Table 4.8 this improvement carries over to the test set, but according to a MAPSSWE test the difference is not statistically significant.

4.3.7 Discussion

Table 4.8 shows the performance on the development and test sets of the PLP and MLP tandem baselines and our best performing k-NN tandem system. k-NN tandem achieves far worse results than MLP tandem and is only barely able to improve upon the baseline PLP system. This is surpris-

Table 4.8: Test Set Results
Phone error rates on the development and test sets for the PLP and MLP tandem baselines, and for the best performing k-NN tandem system.

Features	Gaussians	Phone Error Rate (%)	
		Dev Set	Test Set
PLP	36	32.1	32.9
MLP tandem	38	29.9	31.3
PLP + k-NN tandem	38	31.8	32.8

ing, considering how well the same k-NN features performed in the CRF experiments described in Section 4.2. In this section we will try to examine possible reasons for this lack of improvement.

Both of the tandem systems in Table 4.8 are based on frame classifiers, and we saw a bigger gap between the frame classification performance of MLP and k-NN on the TIMIT data than we did on the STP data. On STP, the phonetic frame classification error rates for the MLP classifier and the best k-NN classifier without any feature transformation[9] were 63.9% and 70.7%, respectively. On TIMIT, the corresponding numbers are 34.9% and 45.9%, and this represents a bigger performance gap in both relative and absolute terms. There are several possible explanations for this. One is that the MLP makes better use of the larger data set than the k-NN classifier. On the other hand, one could argue that an exemplar-based classifier like k-NN should perform quite well on a large data set, since the large data set will contain many exemplars to choose from and might be hard to train a global model for. Another possible explanation is that the inclusion of silences in our TIMIT data creates a skewed prior problem for phonetic classification similar to what we saw for the AF classification on STP (cf. the discussion of Figure 3.8 on page 83). However, the k-NN results for phonetic classification are still far better on TIMIT than they are on STP, so this effect cannot be that pronounced.

There could, then, be something about a GMM-HMM system which makes k-NN features less suitable as observation vectors than they are for a CRF. As we have touched upon, the sparsity of the k-NN features might make them less amenable to GMM modeling than PLPs or MLP tandem features. We tried to make up for this with a log transformation in which

[9]Other than global normalization (cf. page 66), which we do not consider a feature transformation on par with the likes of PCA and LDA.

the zeros were replaced with Gaussian noise. As we saw in Figure 4.6, this appeared to give us features that would be easier to fit a Gaussian mixture to, but it still hurt the performance. There might be something about the log transformation itself that is detrimental to the HMM's performance, but this remains an open question.

We mentioned that our articulatory labels are likely to be less accurate for TIMIT than they were for STP, due to the less detailed phonetic transcription. The approximate nature of the articulatory labeling might have made the articulatory information less useful to the k-NN tandem system than we had hoped. It does seem like the articulatory labels did provide some useful information, though. Considering the three bottom lines of Table 4.7, where the effect of the differing dimensionality is eliminated by PCA, we see that features based on AFs do better than those based on phones, and those based on both phones and AFs do better than either of them separately. However, the differences are very small (as they were bound to be since the k-NN features in this case contribute only 3 of the vectors' 42 dimensions) and might be spurious.

Future Work

We have seen that drawing the nearest neighbors of a frame from the same part of the data set as the frame itself poses a potential problem when making the k-NN features for the tandem system. We tried to deal with this by excluding a number of adjacent frames from the neighbor search, which gave a slight improvement of results. There are three alternative approaches to this problem that we think are worth exploring. The difference between them is arguably one of degree.

The first of these approaches is to exclude from the neighbor search all frames from the same utterance as the query frame. One might argue that the most important thing is to exclude the frames from the same phone segment as the query frame, but there are bound to be some coarticulation effects across phones; by excluding the entire utterance from the neighbor search one can be certain that there are no coarticulation effects between the query frame and its neighbors.

The second approach goes one step further and would exclude all frames from the same speaker from the nearest neighbor search. The idea here is to avoid idiosyncrasies of the speaker leading other phones uttered by this speaker to be seen as more similar to the current phone than the same phone uttered by a different speaker. Since TIMIT has a high number of speakers (630), this is feasible. A confounding of speaker similarities with speech similarities can affect all classifiers, but exemplar-based classifiers are more sensitive to this than classifiers that try to build generalizable models, which usually entails some level of smoothing. On the other hand, this kind of exclusion of parts of the training set on a frame-by-frame basis is far more easily done with a k-NN classifier than with an MLP.

Finally, one might consider splitting the training data into a k-NN search set and a disjoint HMM training set. If the data were divided equally, this would halve the amount of training data for both the k-NN and the HMM, which would be an obvious drawback. On the other hand, training the HMM on data which are not seen by the k-NN classifier (from the results of which the feature vectors for the HMM are made) might be beneficial enough for a k-NN tandem system to make up for the reduction in the size of the training set.

Chapter 5

Conclusion

This last chapter summarizes the main contributions of this thesis, and points out some directions for future work.

5.1 Research Findings

Frame Classification by Nearest Neighbors We have investigated how a set of multi-valued articulatory features introduced by Livescu [2005] can be classified by nearest neighbor methods. We did extensive experiments on the frame classification of this set of AFs as well as of phones for the Switchboard Transcription Project corpus. The k-NN classifiers generally did slightly worse than the MLP baseline. However, using the posterior probability outputs of the MLPs as input to a k-NN classifier (what we call MLPP k-NN) outperformed the MLPs.

Concatenating nine observation vectors improves results for both k-NN and MLP. In spite of this high dimensionality, we were unable to get substantial gains from reducing it with linear transformations. We saw that PCA was unable to retain enough of the neighborhood structure in the top dimensions to reap much benefit from reducing the dimensionality. LDA limits the dimensionality to the number of classes minus one, resulting in a single dimension for some classification tasks, and fared worse than PCA despite being a supervised transformation. CCA, when used as a dimensionality reduction, has a lot of similarities with LDA but allows for a higher dimensionality when the label view is defined in a certain way; it attained better results than LDA. We also made a novel variant of CCA, which can be regarded as a multi-task variant of LDA. This transformation increases the allowed dimensionality for our task from 8 to 24, and outperformed regular CCA. Locality preserving projections, a manifold-based linear transformation that has shown promise in other ASR applications, was also unable to attain substantial improvements. We see that the linear transformations generally struggle to retain the discriminative information of the observation vectors, particularly when the dimensionality is greatly reduced.

Doing a global normalization of the concatenated observation vectors, giving them zero mean and unit standard deviation along each dimension, improved the performance of the k-NN classifier greatly—particularly with 1-frame vectors, where the obtained error rates were on par with those of our most successful linear transformations.

In our MLPP k-NN method, which gave the best results, we essentially use MLP classifiers of all eight AFs as a nonlinear transformation of the observation vectors. The decreases in error rates that this method gives compared to the MLPs suggest that it is able to capture useful dependencies between the AFs.

We also did frame classification experiments on the TIMIT corpus. Consisting of read rather than spontaneous speech, the TIMIT data proved an easier classification task than the STP data, but the performance gap between MLP and k-NN was bigger here. For our experiments on TIMIT we included silence frames, and we were able to improve performance slightly by reducing the amount of silence in the training data before searching for nearest neighbors.

k-NN Features in a CRF for Forced Transcription We have investigated how we could use the k-NN frame classification results in automatic transcription systems by way creating "posterior-like" features based on them. We constructed three feature functions for this task and applied the resulting feature vectors in two systems for automatic transcription.

In the first of these, a CRF for forced transcription of AFs on the STP data, we saw that the k-NN features improved slightly upon the results with standard MLP features We also saw that distance-weighting the k-NN features hurt performance, but that rank-weighting was beneficial.

k-NN Features in a Tandem HMM Phone Recognizer For the second transcription system, a tandem HMM for phone recognition on TIMIT, the results with k-NN features were substantially worse than those with MLP features. We have considered several possible explanations for this result. The larger performance gap between the MLP and k-NN frame classifiers on TIMIT compared to STP may have played a role. Another possible reason is that sparsity of the k-NN features made them unsuitable for Gaussian mixture modeling. We tried to mitigate this with a modified log transformation. This has proved beneficial in previous work, and graphical inspection of the resulting features looked promising, but the effect on the performance was detrimental.

We had hoped the k-NN features' ability to incorporate articulatory information without needing to train extra classifiers would be beneficial in these experiments. We did find some benefit, as demonstrated by gains over the purely phone-based k-NN features, but these were far from closing the performance gap to the MLP tandem system.

This work represents a first attempt at creating a k-NN tandem system. We believe this approach holds promise, particularly with respect to incorporating articulatory information, but more research is needed to realize this potential.

5.2 Future Work

We have pointed out some directions for future work within the specific areas of this thesis in Sections 3.8, 4.2.6 and 4.3.7. In this section we will present some broader ideas for further research.

We hope to use the CRF presented in Section 4.2, or a similar system, to do forced transcription of articulatory features for the entire Switchboard corpus. We believe this would be a valuable resource for the speech community. The resulting articulatory labeling would not be perfect, but it would be superior to AF labels that are phone-derived.

We would also be interested to see whether such forced-transcribed AF labels could improve our frame classification results. As noted, the phone-derived AF labeling used for the frame classification experiments reported in Chapter 3 is unreliable. A more reliable forced-transcribed labeling might make the classification task easier, as one would expect incorrect labels to be harder to learn from. The ground-truth labels that the forced transcription CRF is trained on would still be phone derived, so the improvement might be small. We could build upon this improvement, however, by repeating the process iteratively, using the AF labels output by the CRF as the ground truth in the next iteration—effectively doing a form of semi-supervised learning [Chapelle et al., 2006]. The results for the part of Switchboard that is not annotated at the phonetic level could be compared to AF labels derived from the phonetic labeling of the Semi-Supervised Switchboard Transcription Project corpus [Subramanya and Bilmes, 2009].

Another approach would be to experiment with deriving the AF labels from something other than the phonetic labeling to begin with. An intriguing possibility is to use actual articulatory measurements as a basis. Several articulatory databases are available, containing simultaneously recorded speech and articulatory measurements obtained by X-ray microbeams [Westbury, 1994], electromagnetic articulography [Wrench, 2000], or real-time magnetic resonance imaging [Narayanan et al., 2014]. The articulatory data in these corpora are continuous; to use them as a labeling—e.g., for frame classification—they would need to be discretized. This discretization of the articulatory space would be a natural candidate for learning. The ability of a frame classifier to obtain a low mean error rate for the set of AFs could be a workable optimization criterion. If the number of AFs and their cardinalities were fixed, we would have a relatively well-defined optimization problem. k-NN would be a very suitable method to use here. Since no training would be necessary for each discretization, we would be able to try a large number of them with far greater ease than with classifiers like MLPs, DNNs or support vector machines, which would have to be retrained for each discretization.

Being able to obtain a low frame classification error for an AF set does not necessarily imply that the AF set is useful, however. In the end, the test for an AF classifier will be how it works in an application. Input in a speech recognition engine would be the most natural test for usefulness. An articulatory tandem system like the one we presented in Section 4.3 is a start, but we believe articulatory modeling to have a greater potential in systems that are less reliant on phones as their basic unit. k-NN classification in conjunction with an articulatory speech recognition system with a similar architecture to the forced transcription CRF presented in Section 4.2, would be an excellent way of investigating the merit of different AF sets.

The experiments we have presented in this thesis have not been aimed at improving state-of-the-art results so much as exploring the promise of an articulatory approach to speech recognition combined with exemplar-based modeling. It is our hope that they have shed some light on both the merits and the problems of this approach, and can serve as a guide and an encouragement for future work in this field.

Appendix A

Phone Sets and Articulatory Feature Mappings

Table A.1: Arpabet Vowels
The Vowels of the standard Arpabet Phonetic Alphabet and their relationship to IPA symbols. (Table from Jurafsky [2004].)

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[i]	[iy]	lily	[ˈlɪli]	[l ih l iy]
[ɪ]	[ih]	lily	[ˈlɪli]	[l ih l iy]
[eɪ]	[ey]	daisy	[ˈdeɪzi]	[d ey z i]
[ɛ]	[eh]	poinsettia	[pɔɪnˈsɛɾiə]	[p oy n s eh dx iy ax]
[æ]	[ae]	aster	[ˈæstɚ]	[ae s t axr]
[ɑ]	[aa]	poppy	[ˈpɑpi]	[p aa p i]
[ɔ]	[ao]	orchid	[ˈɔrkɨd]	[ao r k ix d]
[ʊ]	[uh]	woodruff	[ˈwʊdrʌf]	[w uh d r ah f]
[oʊ]	[ow]	lotus	[ˈloʊɾəs]	[l ow dx ax s]
[u]	[uw]	tulip	[ˈtulɨp]	[t uw l ix p]
[ʌ]	[uh]	buttercup	[ˈbʌɾɚˌkʌp]	[b uh dx axr k uh p]
[ɝ]	[er]	bird	[ˈbɝd]	[b er d]
[aɪ]	[ay]	iris	[ˈaɪrɨs]	[ay r ix s]
[aʊ]	[aw]	sunflower	[ˈsʌnflaʊɚ]	[s ah n f l aw axr]
[oɪ]	[oy]	poinsettia	[pɔɪnˈsɛɾiə]	[p oy n s eh dx iy ax]
[ju]	[y uw]	feverfew	[fivɚfju]	[f iy v axr f y u]
[ə]	[ax]	woodruff	[ˈwʊdrəf]	[w uh d r ax f]
[ɨ]	[ix]	tulip	[ˈtulɨp]	[t uw l ix p]
[ɚ]	[axr]	heather	[ˈhɛðɚ]	[h eh dh axr]
[ʉ]	[ux]	dude[1]	[dʉd]	[d ux d]

Table A.2: Arpabet Consonants
The consonants of the standard Arpabet Phonetic Alphabet and their relationship to IPA symbols. (Table from Jurafsky [2004].)

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[p]	[p]	parsley	[ˈparsli]	[p aa r s l iy]
[t]	[t]	tarragon	[ˈtærəgɑn]	[t ae r ax g aa n]
[k]	[k]	catnip	[ˈkætnɨp]	[k ae t n ix p]
[b]	[b]	bay	[beɪ]	[b ey]
[d]	[d]	dill	[dɪl]	[d ih l]
[g]	[g]	garlic	[ˈgɑrlɨk]	[g aa r l ix k]
[m]	[m]	mint	[mɪnt]	[m ih n t]
[n]	[n]	nutmeg	[ˈnʌtmɛg]	[n ah t m eh g
[ŋ]	[ng]	ginseng	[ˈdʒɪnsɨŋ]	[jh ih n s ix ng]
[f]	[f]	fennel	[ˈfɛnl̩]	[f eh n el]
[v]	[v]	clove	[kloʊv]	[k l ow v]
[θ]	[th]	thistle	[ˈθɪsl̩]	[th ih s el]
[ð]	[dh]	heather	[ˈhɛðɚ]	[h eh dh axr]
[s]	[s]	sage	[seɪdʒ]	[s ey jh]
[z]	[z]	hazelnut	[ˈheɪzl̩nʌt]	[h ey z el n ah t]
[ʃ]	[sh]	squash	[skwɑʃ]	[s k w a sh]
[ʒ]	[zh]	ambrosia	[æmˈbroʊʒə]	[ae m b r ow zh ax]
[tʃ]	[ch]	chicory	[ˈtʃɪkɚi]	[ch ih k axr iy]
[dʒ]	[jh]	sage	[seɪdʒ]	[s ey jh]
[l]	[l]	licorice	[ˈlɪkɚɨʃ]	[l ih k axr ix sh]
[w]	[w]	kiwi	[ˈkiwi]	[k iy w iy]
[r]	[r]	parsley	[ˈpɑrsli]	[p aa r s l iy]
[j]	[y]	yew	[yu]	[y uw]
[h]	[h]	horseradish	[ˈhɔrsrædɪʃ]	[h ao r s r ae d ih sh]
[ʔ]	[q]	uh-oh	[ʔʌʔoʊ]	[q ah q ow]
[ɾ]	[dx]	butter	[ˈbʌɾɚ]	[b ah dx axr]
[ɾ̃]	[nx]	wintergreen	[wɪɾ̃ɚgrin]	[w ih nx axr g r i n]
[l̩]	[el]	thistle	[ˈθɪsl̩]	[th ih s el]

Table A.3: Definition of the Articulatory Feature Set
The set of articulatory features based on the vocal tract variables of articulatory phonology introduced by Livescu [2005] and used for the experiments in this thesis.

Feature	Name	Values
Lip constriction location	LIP-LOC	0 = protruded 1 = labial 2 = dental
Lip opening degree	LIP-OPEN	0 = closed 1 = critical 2 = narrow 3 = wide
Tongue tip constriction location	TT-LOC	0 = inter-dental 1 = alveolar 2 = palato-alveolar 3 = retroflex
Tongue tip opening degree	TT-OPEN	0 = closed 1 = critical 2 = narrow 3 = mid-narrow 4 = mid 5 = wide
Tongue body constriction location	TB-LOC	0 = palatal 1 = velar 2 = uvular 3 = pharyngeal
Tongue body opening degree	TB-OPEN	0 = closed 1 = critical 2 = narrow 3 = mid-narrow 4 = mid 5 = wide
Velum opening degree	VEL	0 = closed 1 = open
Glottis opening degree	GLOT	0 = closed 1 = critical 2 = wide

Table A.4: Phone Set and Articulatory Feature Mapping for the STP Corpus In our phonetic labeling for the STP data, we depart from the standard Arpabet in three ways. (1) We split the diphthongs ([aw], [ay], [ey], [ow] and [oy]) in two parts—e.g., [ay1] and [ay2]. (2) Similarly, we split the plosives ([b], [d], [g], [k], [p] and [t]) into occlusion and burst segments—e.g., [pcl] and [p]. (3) Finally, we use the diacritic [_n] for nasalized variants of certain vowels and diphthongs that are ordinarily not nasalized—e.g., [aa_n].The transcription symbol for silence is [sil], and [?] has been used when the transcriber could not make out the sound.

Phone	LIP-LOC	LIP-OPEN	TT-LOC	TT-OPEN	TB-LOC	TB-OPEN	VEL	GLOT
[aa]	1	3	1	5	3	3	0	1
[aa_n]	1	3	1	5	3	3	1	1
[ae]	1	3	1	5	1	5	0	1
[ae_n]	1	3	1	5	1	5	1	1
[ah]	1	3	1	4	2	4	0	1
[ah_n]	1	3	1	4	2	4	1	1
[ao]	0	3	1	5	3	3	0	1
[ao_n]	0	3	1	5	3	3	1	1
[aw1_n]	1	3	1	5	1	5	1	1
[aw2_n]	0	2	2	5	2	3	1	1
[aw1]	1	3	1	5	1	5	0	1
[aw2]	0	2	2	5	2	3	0	1
[ax]	1	3	1	4	2	4	0	1
[ax_n]	1	3	1	4	2	4	1	1
[axr]	1	3	3	2	2	5	0	1
[ay1_n]	1	3	1	5	3	3	1	1
[ay2_n]	1	3	1	3	0	3	1	1
[ay1]	1	3	1	5	3	3	0	1
[ay2]	1	3	1	3	0	3	0	1
[b]	1	1	1	4	2	5	0	1
[bcl]	1	0	1	4	2	5	0	1
[ch]	1	3	2	1	0	4	0	2
[d]	1	3	1	1	1	4	0	1
[dcl]	1	3	1	0	1	4	0	1
[dh]	1	3	0	1	2	4	0	1
[dh_n]	1	3	0	1	2	4	1	1
[dx]	1	3	1	2	1	4	0	1
[eh]	1	3	1	4	0	4	0	1
[eh_n]	1	3	1	4	0	4	1	1
[el]	1	3	1	0	2	2	0	1
[el_n]	1	3	1	0	2	2	1	1
[em]	1	0	1	4	2	4	1	1
[en]	1	3	1	0	2	4	1	1
[eng]	1	3	2	5	1	0	1	1
[epi]	0	0	0	0	0	2	0	0
[er]	1	3	3	2	2	5	0	1
[er_n]	1	3	3	2	2	5	1	1

Continued on next page

Table A.4 – *Continued from previous page*

Phone	LIP-LOC	LIP-OPEN	TT-LOC	TT-OPEN	TB-LOC	TB-OPEN	VEL	GLOT
[ey1_n]	1	3	1	4	0	4	1	1
[ey2_n]	1	3	1	3	0	3	1	1
[ey1]	1	3	1	4	0	4	0	1
[ey2]	1	3	1	3	0	3	0	1
[f]	2	1	1	4	1	4	0	2
[g]	1	3	2	5	1	1	0	1
[gcl]	1	3	2	5	1	0	0	1
[hh]	1	3	1	4	2	4	0	2
[hh_n]	1	3	1	4	2	4	1	2
[ih]	1	3	1	3	0	3	0	1
[ih_n]	1	3	1	3	0	3	1	1
[ix]	1	3	1	3	0	4	0	1
[ix_n]	1	3	1	3	0	4	1	1
[iy]	1	3	1	3	0	2	0	1
[iy_n]	1	3	1	3	0	2	1	1
[jh]	1	3	2	1	0	4	0	1
[k]	1	3	2	5	1	1	0	2
[kcl]	1	3	2	5	1	0	0	2
[l]	1	3	1	0	2	2	0	1
[l_n]	1	3	1	0	2	2	1	1
[lg]	1	3	1	0	2	2	0	1
[lg_n]	1	3	1	0	2	2	1	1
[m]	1	0	1	4	2	4	1	1
[n]	1	3	1	0	2	4	1	1
[ng]	1	3	2	5	1	0	1	1
[nx]	1	3	1	2	1	4	1	1
[ow1_n]	0	3	2	5	2	3	1	1
[ow2_n]	0	2	2	5	1	2	1	1
[ow1]	0	3	2	5	2	3	0	1
[ow2]	0	2	2	5	1	2	0	1
[oy1_n]	0	3	1	5	2	3	1	1
[oy2_n]	1	3	1	3	0	3	1	1
[oy1]	0	3	1	5	2	3	0	1
[oy2]	1	3	1	3	0	3	0	1
[p]	1	3	1	4	2	5	0	2
[pcl]	1	0	1	4	2	5	0	2
[q]	1	3	1	4	2	4	0	0
[r]	1	3	3	2	2	5	0	1
[r_n]	1	3	3	2	2	5	1	1
[s]	1	3	1	1	2	4	0	2
[sh]	1	3	2	1	0	4	0	2
[t]	1	3	1	1	1	4	0	2
[tcl]	1	3	1	0	1	4	0	2
[th]	1	3	0	1	2	4	0	2
[uh]	0	3	2	5	2	3	0	1
[uh_n]	0	3	2	5	2	3	1	1

Continued on next page

Table A.4 – *Continued from previous page*

Phone	LIP-LOC	LIP-OPEN	TT-LOC	TT-OPEN	TB-LOC	TB-OPEN	VEL	GLOT
[uw]	0	2	2	5	1	2	0	1
[uw_n]	0	2	2	5	1	2	1	1
[ux]	0	2	1	3	0	3	0	1
[ux_n]	0	2	1	3	0	3	1	1
[v]	2	1	1	4	1	4	0	1
[v_n]	2	1	1	4	1	4	1	1
[w]	0	2	2	5	2	2	0	1
[w_n]	0	2	2	5	2	2	1	1
[y]	1	3	1	3	0	2	0	1
[y_n]	1	3	1	3	0	2	1	1
[z]	1	3	1	1	2	4	0	1
[z_n]	1	3	1	1	2	4	1	1
[zh]	1	3	2	1	0	4	0	1
[sil]	2	0	0	0	0	0	0	0
[?]	0	0	0	0	0	0	0	0

Table A.5: Phone Set and Articulatory Feature Mapping for the TIMIT Corpus The phonetic labeling of the TIMIT data departs from the standard Arpabet mainly in splitting the plosives into occlusion and burst segments (e.g., [pcl] and [p]) like we saw in the STP labeling in the table above. There are two transcription symbols for silence: [pau] for short pauses and [h#] for longer silences.

Phone	LIP-LOC	LIP-OPEN	TT-LOC	TT-OPEN	TB-LOC	TB-OPEN	VEL	GLOT
[aa]	1	3	1	5	3	3	0	1
[ae]	1	3	1	5	1	5	0	1
[ah]	1	3	1	4	2	4	0	1
[ao]	0	3	1	5	3	3	0	1
[aw]	1	3	1	5	1	5	0	1
[ax]	1	3	1	4	2	4	0	1
[axh]	1	3	1	4	2	4	0	2
[axr]	1	3	3	2	2	5	0	1
[ay]	1	3	1	5	3	3	0	1
[b]	1	1	1	4	2	5	0	1
[bcl]	1	0	1	4	2	5	0	1
[ch]	1	3	2	1	0	4	0	2
[d]	1	3	1	1	1	4	0	1
[dcl]	1	3	1	0	1	4	0	1
[dh]	1	3	0	1	2	4	0	1
[dx]	1	3	1	2	1	4	0	1

Continued on next page

Table A.5 – *Continued from previous page*

Phone	LIP-LOC	LIP-OPEN	TT-LOC	TT-OPEN	TB-LOC	TB-OPEN	VEL	GLOT
[eh]	1	3	1	4	0	4	0	1
[el]	1	3	1	0	2	2	0	1
[em]	1	0	1	4	2	4	1	1
[en]	1	3	1	0	2	4	1	1
[eng]	1	3	1	0	2	4	1	1
[epi]	0	0	0	0	0	2	0	0
[er]	1	3	3	2	2	5	0	1
[ey]	1	3	1	4	0	4	0	1
[f]	2	1	1	4	1	4	0	2
[g]	1	3	2	5	1	1	0	1
[gcl]	1	3	2	5	1	0	0	1
[hh]	1	3	1	4	2	4	0	2
[hv]	1	3	1	4	2	4	0	1
[ih]	1	3	1	3	0	3	0	1
[ix]	1	3	1	3	0	3	0	1
[iy]	1	3	1	3	0	2	0	1
[jh]	1	3	2	1	0	4	0	1
[k]	1	3	2	5	1	1	0	2
[kcl]	1	3	2	5	1	0	0	2
[l]	1	3	1	0	2	2	0	1
[m]	1	0	1	4	2	4	1	1
[n]	1	3	1	0	2	4	1	1
[ng]	1	3	2	5	1	0	1	1
[nx]	1	3	2	5	1	0	1	1
[ow]	0	3	2	5	2	3	0	1
[oy]	0	3	1	5	2	3	0	1
[p]	1	3	1	4	2	5	0	2
[pcl]	1	0	1	4	2	5	0	2
[q]	1	3	2	5	1	1	0	2
[r]	1	3	3	2	2	5	0	1
[s]	1	3	1	1	2	4	0	2
[sh]	1	3	2	1	0	4	0	2
[t]	1	3	1	1	1	4	0	2
[tcl]	1	3	1	0	1	4	0	2
[th]	1	3	0	1	2	4	0	2
[uh]	0	3	2	5	2	3	0	1
[uw]	0	2	2	5	1	2	0	1
[ux]	0	2	2	5	1	2	0	1
[v]	2	1	1	4	1	4	0	1
[w]	0	2	2	5	2	2	0	1
[y]	1	3	1	3	0	2	0	1
[z]	1	3	1	1	2	4	0	1
[zh]	1	3	2	1	0	4	0	1
[pau]	0	0	0	0	0	2	0	0
[h#]	2	0	0	0	0	0	0	0

Appendix B

Histograms of Label Distributions

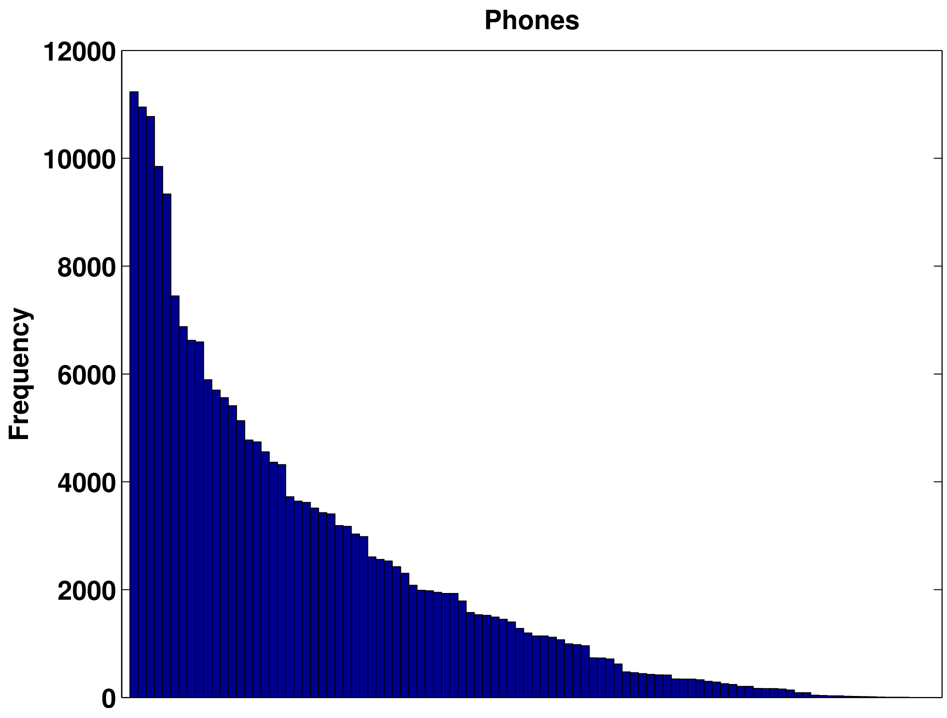


Figure B.1: Histogram of the frequencies of the 98 phone labels in the STP data used for the experiments presented in Chapter 3, from highest to lowest. The total number of frames is just shy of 220 000. We see that, except for the five most frequent phones, the frequencies approximately follow a Zipfian distribution.

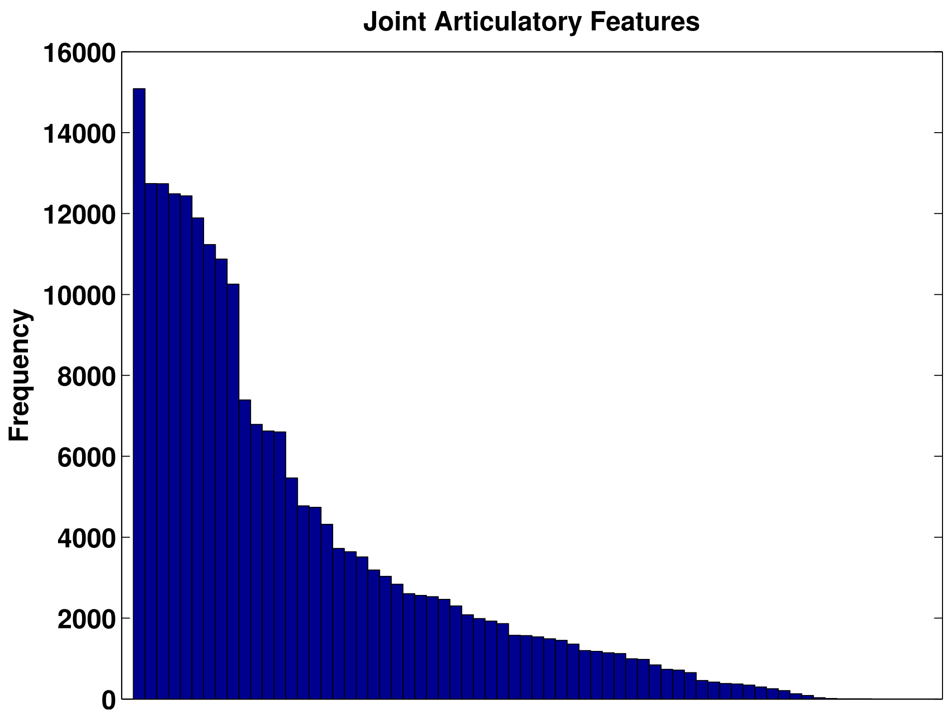


Figure B.2: Although the number of possible combinations of the values of our eight articulatory features is 41 472, only 64 such combinations occur in the data set used for the experiments presented in Chapter 3. This histogram shows the distribution of these combinations, which are used as classes in the Joint AF experiments in Section 3.4. As in the previous figure, the total number of frames is slightly below 220 000.

Bibliography

[**Akaho, 2001**] Shotaro Akaho, "A Kernel Method for Canonical Correlation Analysis," in *Proceedings of the International Meeting of the Psychometric Society (IMPS)*, Osaka, Japan, Sep. 2001.

[**Albright, 1958**] Robert William Albright, "The International Phonetic Alphabet: Its Background and Development," *International Journal of American Linguistics*, vol. 24, no. 1, 1958.

[**Ananthakrishnan and Engwall, 2011a**] G. Ananthakrishnan and Olov Engwall, "Mapping Between Acoustic and Articulatory Gestures," *Speech Communication*, vol. 53, no. 4, 2011.

[**Ananthakrishnan and Engwall, 2011b**] G. Ananthakrishnan and Olov Engwall, "Resolving Non-Uniqueness in the Acoustic-to-Articulatory Mapping," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[**Andoni and Indyk, 2006**] Alexandr Andoni and Piotr Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, Berkeley, California, 2006.

[**Arora and Livescu, 2013**] Raman Arora and Karen Livescu, "Multi-View CCA-Based Acoustic Features for Phonetic Recognition Across Speakers and Domains," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[**Arya et al., 1998**] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions," *Journal of the ACM*, vol. 45, no. 6, 1998.

[**Asaei et al., 2010b**] Afsaneh Asaei, Hervé Bourlard, and Benjamin Picart, “Investigation of KNN Classifier on Posterior Features Towards Application in Automatic Speech Recognition,” Idiap Research Institute (Idiap-RR-11-2010), Technical report, 2010.

[**Asaei et al., 2010a**] Afsaneh Asaei, Benjamin Picart, and Hervé Bourlard, “Analysis of Phone Posterior Feature Space Exploiting Class-Specific Sparsity and Mlp-Based Similarity Measure,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, Texas, 2010.

[**Baker, 1975**] James K. Baker, “The DRAGON system – An overview,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, 1975.

[**Baudat and Anouar, 2000**] Gaston Baudat and Fatiha Anouar, “Analysis of Phone Posterior Feature Space Exploiting Class-Specific Sparsity and MLP-Based Similarity Measure,” *Neural Computation*, vol. 12, no. 10, 2000.

[**Belkin and Niyogi, 2003**] Mikhail Belkin and Partha Niyogi, “Laplacian Eigenmaps for Dimensionality Reduction and Data,” *Neural Computation*, vol. 15, no. 6, 2003.

[**Bellman and Dreyfus, 1962**] Richard E. Bellman and Stuart E. Dreyfus, *Applied Dynamic Programming.* Princeton University Press, 1962.

[**Bengio, 2009**] Yoshua Bengio, “Learning Deep Architectures for AI,” *Foundations and Trends in Machine Learning*, vol. 2, no. 1, 2009.

[**Bilmes, 2006**] Jeff A. Bilmes, “What HMMs Can Do,” *IEICE Transactions on Information and Systems*, vol. E89-D, no. 3, Mar. 2006.

[**Bilmes and Bartels, 2005**] Jeff A. Bilmes and Chris Bartels, “Graphical Model Architectures for Speech Recognition,” *IEEE Signal Processing Magazine*, vol. 22, no. 5, 2005.

[**Bisani and Ney, 2004**] Maximilian Bisani and Hermann Ney, “Bootstrap Estimates for Confidence Intervals in ASR Performance Evaluation,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Montreal, Canada, 2004.

[**Bishop, 2006**] Christopher M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[**Bogert et al., 1963**] Bruce P. Bogert, Michael J.R. Healy, and John W. Tukey, “The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum and Saphe Cracking,” in *Time Series Analysis*, M. Rosenblatt, Editor. John Wiley & Sons, 1963, ch. 15.

[**Borga, 1998**] Magnus Borga, “Learning Multidimensional Signal Processing,” Ph.D. dissertation, Linköping University, 1998.

[**Bourlard and Morgan, 1998**] Hervé Bourlard and Nelson Morgan, “Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions,” in *Adaptive Processing of Sequences and Data Structures.* Springer, 1998.

[**Bourlard and Morgan, 1994**] Hervé Bourlard and Nelson Morgan, *Connectionist Speech Recognition – A Hybrid Approach.* Kluwer Academic Publishers, 1994.

[**Bourlard et al., 1996**] Hervé Bourlard, Stéphane Dupont, and Christophe Ris, “Multi Stream Speech Recognition,” Idiap Research Institute, Research Report (Idiap-RR 96-07), Technical report, 1996.

[**Bromberg et al., 2007**] Ilana Bromberg, Qiang Fu, Jun Hou, Jinyu Li, Chengyuan Ma, Brett Matthews, Antonio Moreno-daniel, Jeremy Morris, Sabato Marco Siniscalchi, Yu Tsao, and Yu Wang, “Detection-Based ASR in the Automatic Speech Attribute Transcription Project,” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Antwerp, Belgium, 2007.

[**Browman and Goldstein, 1986**] Catherine P. Browman and Louis Goldstein, “Towards an Articulatory Phonology,” in *Phonology Yearbook*, 3rd ed., C. Ewen and J. Anderson, Editors. Cambridge University Press, 1986, vol. 3.

[**Browman and Goldstein, 1989**] Catherine P. Browman and Louis Goldstein, “Articulatory Gestures As Phonological Units,” *Phonology*, vol. 6, no. 02, 1989.

[**Browman and Goldstein, 1992**] Catherine P. Browman and Louis Goldstein, "Articulatory Phonology: An Overview," *Phonetica*, vol. 49, no. 3-4, 1992.

[**Brunelli and Poggio, 1993**] Roberto Brunelli and Tomaso Poggio, "Face Recognition: Features Versus Templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, 1993.

[**Çetin et al., 2007**] Özgür Çetin, Arthur Kantor, Simon King, Chris Bartels, Mathew Magimai-Doss, Joe Frankel, and Karen Livescu, "An Articulatory Feature-Based Tandem Approach and Factored Observation Modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, 2007.

[**Chapelle et al., 2006**] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, Editors, *Semi-Supervised Learning.* MIT Press, 2006.

[**Chaudhuri et al., 2009**] Kamalika Chaudhuri, Sham M. Kakade, Karen Livescu, and Karthik Sridharan, "Multi-View Clustering via Canonical Correlation Analysis," in *Proceedings of the International Conference on Machine Learning (ICML)*, Montreal, Canada, 2009.

[**Chen and Wang, 2009**] I-Fan Chen and Hsin-Min Wang, "Articulatory Feature Asynchrony Analysis and Compensation in Detection-Based ASR," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Brighton, UK, 2009.

[**Chomsky and Halle, 1968**] Noam Chomsky and Morris Halle, *The Sound Pattern of English.* MIT Press, 1968.

[**Clark, 2006**] Rob Clark, *Speech Processing.* Simon King / University of Edinburgh, 2006.

[**Cole et al., 1995**] Ronald Cole, Mike Noel, Terri Lander, and Terry Durham, "New Telephone Speech Corpora at CSLU," in *Proceedings of Eurospeech*, Madrid, Spain, 1995.

[**Cover, 1968**] Thomas M. Cover, "Estimation by the Nearest Neighbor Rule," *IEEE Transactions on Information Theory*, vol. 14, no. 1, 1968.

[**Cover and Hart, 1967**] Thomas M. Cover and Peter E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, 1967.

[**De Bie and De Moor, 2003**] Tijl De Bie and Bart De Moor, "On the Regularization of Canonical Correlation Analysis," in *Proceedings of the International Symposium on Independent Component Analysis and Blind Source Separation*, Nara, Japan, 2003.

[**De la Torre, 2008**] Fernando De la Torre, "A Least-Squares Unified View of PCA, LDA, CCA and Spectral Graph Methods," CMU-RI-TR-08-29, Robotics Institute, Carnegie Mellon University, Technical report, 2008.

[**De Wachter et al., 2007**] Mathias De Wachter, Mike Matton, Kris Demuynck, Patrick Wambacq, Ronald Cools, and Dirk Van Compernolle, "Template-Based Continuous Speech Recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 4, 2007.

[**Deerwester et al., 1990**] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, 1990.

[**Demuynck et al., 2011a**] Kris Demuynck, Dino Seppi, Hugo Van Hamme, and Dirk Van Compernolle, "Progress in Example Based Automatic Speech Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[**Demuynck et al., 2011b**] Kris Demuynck, Dino Seppi, Dirk Van Compernolle, Patrick Nguyen, and Geoffrey Zweig, "Integrating Meta-Information into Exemplar-Based Speech Recognition with Segmental Conditional Random Fields," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[**Deng et al., 1997**] Li Deng, Graham Ramsay, and Don Sun, "Production Models As a Structural Basis for Automatic Speech Recognition," *Speech Communication*, vol. 22, no. 2-3, 1997.

[**Deselaers et al., 2007**] Thomas Deselaers, Georg Heigold, and Hermann Ney, "Speech Recognition With State-based Nearest Neighbour Classifiers," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Antwerp, Belgium, 2007.

[**Dietterich, 1998**] Thomas G. Dietterich, “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms,” *Neural Computation*, vol. 10, no. 7, 1998.

[**Dijkstra, 1959**] Edsger W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik*, vol. 1, 1959.

[**Edwards, 1948**] Allen L. Edwards, “Note on the ‘Correction for Continuity’ in Testing the Significance of the Difference Between Correlated Proportions,” *Psychometrika*, vol. 13, no. 3, 1948.

[**Efron and Tibshirani, 1993**] Bradley Efron and Robert Tibshirani, *An Introduction to the Bootstrap*, in series Monographs on Statistics and Applied Probability. CRC Press, 1993.

[**Ek et al., 2008**] Carl Henrik Ek, Jon Rihan, Philip H.S. Torr, Grégory Rogez, and Neil D. Lawrence, “Ambiguity Modeling in Latent Spaces,” in *Machine Learning for Multimodal Interaction.* Springer, 2008.

[**Ellis, 2000**] Daniel P. W. Ellis, “ICSI Speech FAQ: 6.3 How are neural nets trained?” http://www1.icsi.berkeley.edu/Speech/faq/nn-train.html, 2000.

[**Ellis, 2005**] Daniel P. W. Ellis, “PLP and RASTA (and MFCC, and Inversion) in Matlab,” http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/, 2005.

[**Ellis, 2013**] Daniel P. W. Ellis, “Reproducing the feature outputs of common programs using Matlab and melfcc.m,” http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/mfccs.html, 2013.

[**Ellis et al., 2001**] Daniel P. W. Ellis, Rita Singh, and Sunil Sivadas, “Tandem Acoustic Modeling in Large-Vocabulary Recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Salt Lake City, Utah, 2001.

[**Fisher, 1936**] Ronald Aylmer Fisher, “The Use of Multiple Measurements in Taxonomic Problems,” *Annals of Eugenics*, vol. 7, 1936.

[**Fosler-Lussier et al., 2013**] Eric Fosler-Lussier, Yanzhang He, Preethi Jyothi, and Rohit Prabhavalkar, “Conditional Random Fields in Speech, Audio, and Language Processing,” *Proceedings of the IEEE*, vol. 101, no. 5, 2013.

[**Frankel et al., 2004**] Joe Frankel, Mirjam Wester, and Simon King, "Articulatory Feature Recognition Using Dynamic Bayesian Networks," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Jeju Island, Korea, 2004.

[**Frankel et al., 2007a**] Joe Frankel, Mathew Magimai-Doss, Simon King, Karen Livescu, and Özgür Çetin, "Articulatory Feature Classifiers Trained on 2000 hours of Telephone Speech," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Antwerp, Belgium, 2007.

[**Frankel et al., 2007b**] Joe Frankel, Mirjam Wester, and Simon King, "Articulatory Feature Recognition Using Dynamic Bayesian Networks," *Computer Speech & Language*, vol. 21, no. 4, 2007.

[**Friedman, 1997**] Jerome H. Friedman, "On Bias, Variance , 0/1-Loss , and the Curse-of-Dimensionality," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, 1997.

[**Friedman et al., 1977**] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematical Software*, vol. 3, no. 3, 1977.

[**Fukunaga, 1990**] Keinosuke Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. Academic Press, 1990.

[**Gale and Sampson, 1995**] William Gale and Geoffrey Sampson, "Good-Turing Smoothing Without Tears," *Journal of Quantitative Linguistics*, vol. 2, no. 3, 1995.

[**Gales and Young, 2007**] Mark Gales and Steve Young, "The Application of Hidden Markov Models in Speech Recognition," *Foundations and Trends in Signal Processing*, vol. 1, no. 3, 2007.

[**Garofolo et al., 1993**] John Garofolo, Lori Lamel, William Fisher, Jonathan Fiscus, David Pallett, Nancy Dahlgren, and Victor Zue, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," https://catalog.ldc.upenn.edu/LDC93S1, Linguistic Data Consortium, Philadelphia, 1993.

[**Gehring et al., 2013**] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel, "Extracting Deep Bottleneck Features Using Stacked Auto-Encoders," in *Proceedings of the IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), Vancouver, Canada, 2013.

[**Gemmeke et al., 2009**] Jort Florent Gemmeke, Louis ten Bosch, Lou Boves, and Bert Cranen, "Using Sparse Representations for Exemplar Based Continuous Digit Recognition," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Glasgow, Scotland, 2009.

[**Gemmeke et al., 2011**] Jort Florent Gemmeke, Tuomas Virtanen, and Antti Hurmalainen, "Exemplar-Based Sparse Representations for Noise Robust Automatic Speech Recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, 2011.

[**Ghosh and Narayanan, 2011**] Prasanta Kumar Ghosh and Shrikanth S. Narayanan, "A Subject-Independent Acoustic-to-Articulatory Inversion," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[**Gillick and Cox, 1989**] Larry Gillick and Stephen J. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Glasgow, Scotland, 1989.

[**Glass, 2003**] James R. Glass, "A Probabilistic Framework for Segment-Based Speech Recognition," *Computer Speech and Language*, vol. 17, no. 2-3, 2003.

[**Godfrey et al., 1992**] John J. Godfrey, Edward C. Holliman, and Jane McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research and Development," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, San Francisco, California, 1992.

[**Golipour and O'Shaughnessy, 2009**] Ladan Golipour and Douglas O'Shaughnessy, "Context-Independent Phoneme Recognition Using a k-Nearest Neighbour Classification Approach," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009.

[**Golipour and O'Shaughnessy, 2010**] Ladan Golipour and Douglas O' Shaughnessy, "Phoneme Classification and Lattice Rescoring Based on a *k*-NN Approach," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Chiba, Japan, 2010.

[**Golipour and O'Shaughnessy, 2012**] Ladan Golipour and Douglas O' Shaughnessy, "A Segmental Non-Parametric-Based Phoneme Recognition Approach at the Acoustical Level," *Computer Speech & Language*, vol. 26, no. 4, 2012.

[**Golipour and O'Shaughnessy, 2007**] Ladan Golipour and Douglas O'Shaughnessy, "A New Approach for Phoneme Segmentation of Speech Signals," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Antwerp, Belgium, 2007.

[**Golub and Van Loan, 2012**] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, 4th ed. The John Hopkins University Press, 2012.

[**Greenberg, 1996**] Steven Greenberg, "The Switchboard Transcription Project," in *CLSP/JHU Workshop on Innovative Techniques in Continuous Large Vocabulary Speech Recognition*, Baltimore, Maryland, 1996.

[**Greenberg et al., 1996**] Steven Greenberg, Joy Hollenback, and Daniel P. W. Ellis, "Insights into Spoken Language Gleaned from Phonetic Transcription of the Switchboard Corpus," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Atlanta, Georgia, 1996.

[**Grézl et al., 2007**] František Grézl, Martin Karafiát, Stanislav Kontár, and Jan Černocký, "Probabilistic and Bottle-Neck Features for LVCSR of Meetings," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, 2007.

[**Halle, 1992**] Morris Halle, "Phonological Features," in *The International Encyclopedia of Linguistics*, William Bright, Editor. Oxford University Press, 1992.

[**Harborg, 1990**] Erik Harborg, "Hidden Markov Models Applied to Automatic Speech Recognition," Dr. ing. thesis, Norwegian Institute of Technology, 1990.

[**Hardcastle, 1972**] William J. Hardcastle, "The Use of Electropalatography in Phonetic Research," *Phonetica*, vol. 25, no. 4, 1972.

[**Hardoon et al., 2004**] David R. Hardoon, Sandor Szedmak, and John Shawe-Taylor, “Canonical Correlation Analysis: An Overview with Application to Learning Methods,” *Neural Computation*, vol. 16, no. 12, 2004.

[**Hardoon et al., 2007**] David R. Hardoon, Janaina Mourao-Miranda, Michael Brammer, and John Shawe-Taylor, “Unsupervised Analysis of fMRI Data Using Kernel Canonical Correlation.” *NeuroImage*, vol. 37, no. 4, 2007.

[**Harris, 1994**] John Harris, *English Sound Structure.* Blackwell, 1994.

[**Hayes, 2013**] Brian Hayes, “First Links in the Markov Chain,” *American Scientist*, vol. 101, no. 2, 2013.

[**He and Niyogi, 2003**] Xiaofei He and Partha Niyogi, “Locality Preserving Projections,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 16, 2003.

[**Heigold et al., 2012**] Georg Heigold, Hermann Ney, Ralf Schlüter, and Simon Wiesler, “Discriminative Training for Automatic Speech Recognition,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, 2012.

[**Hermansky, 1990**] Hynek Hermansky, “Perceptual linear predictive (PLP) analysis of speech,” *The Journal of the Acoustical Society of America*, vol. 87, no. 4, 1990.

[**Hermansky et al., 2000**] Hynek Hermansky, Daniel P. W. Ellis, and Sangita Sharma, “Tandem Connectionist Feature Extraction for Conventional HMM Systems,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Istanbul, Turkey, 2000.

[**Hestenes, 1958**] Magnus R. Hestenes, “Inversion of Matrices by Biorthogonalization and Related Results,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 6, no. 1, 1958.

[**Hinton et al., 2012**] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury, “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, 2012.

[**Holmes and Holmes, 2001**] John Holmes and Wendy Holmes, *Speech Synthesis and Recognition*, 2nd ed. Taylor and Francis, 2001.

[**Hotelling, 1936**] Harold Hotelling, "Relations between Two Sets of Variates," *Biometrika*, vol. 27, 1936.

[**Houle and Sakuma, 2005**] Michael E. Houle and Jun Sakuma, "Fast Approximate Similarity Search in Extremely High-Dimensional Data Sets," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Tokyo, Japan, 2005.

[**Hu et al., 2010**] Chi Hu, Xiaodan Zhuang, and Mark Hasegawa-Johnson, "FSM-Based Pronunciation Modeling using Articulatory Phonological Code," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Chiba, Japan, 2010.

[**Huang et al., 2001**] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.* Prentice Hall, 2001.

[**Hunt, 1988**] Melvyn J. Hunt, "Evaluating the Performance of Connected-Word Speech Recognition Systems," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New York, New York, 1988.

[**ICSI, 2012**] "International Computer Science Institute (ICSI) Speech Software," http://www.icsi.berkeley.edu/icsi/groups/speech/software, 2012.

[**Indyk and Motwani, 1998**] Piotr Indyk and Rajeev Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, Dallas, Texas, 1998.

[**Jelinek et al., 1975**] Frederick Jelinek, Lalit R. Bahl, and Robert L. Mercer, "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech," *IEEE Transactions on Information Theory*, vol. 21, no. 3, 1975.

[**Johnson et al., 2004**] David Johnson, Dan Ellis, Chris Oei, Chuck Wooters, and Philipp Faerber, "ICSI QuickNet software package," http://www.icsi.berkeley.edu/Speech/qn.html, 2004.

[**Jolliffe, 2002**] Ian T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[**Jurafsky, 2004**] Daniel Jurafsky, “LINGUIST238: Introduction to Computer Speech and Language Processing,” http://web.stanford.edu/class/linguist238/, Stanford University, 2004.

[**Jyothi et al., 2011**] Preethi Jyothi, Karen Livescu, and Eric Fosler-Lussier, “Lexical Access Experiments with Context-Dependent Articulatory Feature-Based Models,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[**Kindermann and Snell, 1980**] Ross Kindermann and J. Laurie Snell, *Markov Random Fields and Their Applications*, in series Contemporary Mathematics. American Mathematical Society, 1980.

[**King and Taylor, 2000**] Simon King and Paul Taylor, “Detection of Phonological Features in Continuous Speech Using Neural Networks,” *Computer Speech & Language*, vol. 14, no. 4, 2000.

[**Kirchhoff, 1999**] Katrin Kirchhoff, “Robust Speech Recognition Using Articulatory Information,” Ph.D. dissertation, University of Bielefeld, 1999.

[**Kirchhoff et al., 2002**] Katrin Kirchhoff, Gernot A. Fink, and Gerhard Sagerer, “Combining Acoustic and Articulatory Feature Information for Robust Speech Recognition,” *Speech Communication*, vol. 37, no. 3-4, 2002.

[**Kneser and Ney, 1995**] Reinhard Kneser and Hermann Ney, “Improved Backing-off for M-Gram Language Modeling,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Detroit, Michigan, 1995.

[**Labiak and Livescu, 2011**] John Labiak and Karen Livescu, “Nearest Neighbors with Learned Distances for Phonetic Frame Classification,” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Florence, Italy, 2011.

[**Lafferty et al., 2001**] John Lafferty, Andrew McCallum, and Fernando C.N. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data,” in *Proceedings of the*

International Conference on Machine Learning (ICML), Williamstown, Massachusetts, 2001.

[**Lee and Siniscalchi, 2013**] Chin-Hui Lee and Sabato Marco Siniscalchi, "An Information-Extraction Approach to Speech Processing: Analysis, Detection, Verification, and Recognition," *Proceedings of the IEEE*, vol. 101, no. 5, 2013.

[**Lee and Hon, 1989**] Kai-Fu Lee and Hsiao-Wuen Hon, "Speaker-Independent Phone Recognition Using Hidden Markov Models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, no. 11, 1989.

[**Lefèvre, 2003**] Fabrice Lefèvre, "Non-Parametric Probability Estimation for HMM-Based Automatic Speech Recognition," *Computer Speech & Language*, vol. 17, no. 2-3, 2003.

[**Liu et al., 2004**] Ting Liu, Andrew W. Moore, Alex Gray, and Ke Yang, "An Investigation of Practical Approximate Nearest Neighbor Algorithms," *Advances in Neural Information Processing Systems (NIPS)*, vol. 17, 2004.

[**Livescu, 2005**] Karen Livescu, "Feature-Based Pronunciation Modeling for Automatic Speech Recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[**Livescu and Glass, 2004**] Karen Livescu and James Glass, "Feature-based Pronunciation Modeling with Trainable Asynchrony Probabilities," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Jeju Island, Korea, 2004.

[**Livescu and Stoehr, 2009**] Karen Livescu and Mark Stoehr, "Multi-View Learning of Acoustic Features for Speaker Recognition," in *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Merano, Italy, 2009.

[**Livescu et al., 2007c**] Karen Livescu, Ari Bezman, Nash Borges, Lisa Yung, Özgür Çetin, Joe Frankel, Simon King, Mathew Magimai-Doss, Xuemin Chi, and Lisa Lavoie, "Manual Transcription of Conversational Speech at the Articulatory Feature Level," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, 2007.

[**Livescu et al., 2007b**] Karen Livescu, Özgür Çetin, Mark Hasegawa-Johnson, Simon King, Chris Bartels, Nash Borges, Arthur Kantor, Partha Lal, Lisa Yung, Ari Bezman, Stephen Dawson-Haggerty, and Bronwyn Woods, "Articulatory Feature-Based Methods for Acoustic and Audio-Visual Speech Recognition: 2006 JHU Summer Workshop Final Report (Tech. Rep. No. WS06)," Center for Language and Speech Processing, Johns Hopkins University, Technical report, 2007.

[**Livescu et al., 2007a**] Karen Livescu, Özgür Çetin, Mark Hasegawa-Johnson, Simon King, Chris Bartels, Nash Borges, Arthur Kantor, Partha Lal, Lisa Yung, Ari Bezman, Stephen Dawson-Haggerty, Bronwyn Woods, Joe Frankel, Mathew Magimai-Doss, and Kate Saenko, "Articulatory Feature-Based Methods for Acoustic and Audio-Visual Speech Recognition: Summary from the 2006 JHU Summer workshop," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, 2007.

[**Loizou and Maybank, 1987**] George Loizou and Stephen J. Maybank, "The Nearest Neighbor and the Bayes Error Rates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 2, 1987.

[**Mari et al., 1997**] Jean-François Mari, Jean-Paul Haton, and Abdelaziz Kriouile, "Automatic Word Recognition Based on Second-Order Hidden Markov Models," *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 1, 1997.

[**Markov, 1913**] Andrey A. Markov, "An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains," *Bulletin of the Imperial Academy of Sciences of St. Petersburg*, vol. 7, no. 3, 1913.

[**McCarthy, 2001**] John J. McCarthy, "Nonlinear Phonology," Linguistics Department Faculty Publication Series, Paper 49, University of Massachusetts, Amherst, Technical report, 2001.

[**McGowan and Berger, 2009**] Richard S. McGowan and Michael A. Berger, "Acoustic-Articulatory Mapping in Vowels by Locally Weighted Regression," *The Journal of the Acoustical Society of America*, vol. 126, no. 4, 2009.

[**McLachlan and Peel, 2000**] Geoffrey McLachlan and David Peel, *Finite Mixture Models*, in series Wiley Series in Probability and Statistics. John Wiley & Sons, 2000.

[**McNemar, 1947**] Quinn McNemar, "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages," *Psychometrika*, vol. 12, no. 2, 1947.

[**Metze, 2007**] Florian Metze, "Discriminative Speaker Adaptation Using Articulatory Features," *Speech Communication*, vol. 49, no. 5, 2007.

[**Metze, 2005**] Florian Metze, "Articulatory Features for Conversational Speech Recognition," Ph.D. dissertation, Universität Karlsruhe, 2005.

[**Metze and Waibel, 2002**] Florian Metze and Alex Waibel, "A Flexible Stream Architecture for ASR Using Articulatory Features," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Denver, Colorado, 2002.

[**Mika et al., 1999**] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, and Klaus-Robert Müller, "Fisher Discriminant Analysis with Kernels," in *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, Madison, Wisconsin, 1999.

[**Mitra et al., 2011**] Vikramjit Mitra, Hosung Nam, Carol Y. Espy-Wilson, Elliot Saltzman, and Louis Goldstein, "Speech Inversion: Benefits of Tract Variables over Pellet Trajectories," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[**Mitra et al., 2014**] Vikramjit Mitra, Ganesh Sivaraman, and Hosung Nam, "Articulatory Features from Deep Neural Networks and Their Role in Speech Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014.

[**Mohri et al., 2002**] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech & Language*, vol. 16, no. 1, 2002.

[**Morgan and Bourlard, 1995**] Nelson Morgan and Hervé Bourlard, "Continuous Speech Recognition – An Introduction to the Hybrid HMM/Connectionist Approach," *IEEE Signal Processing Magazine*, vol. 12, no. 3, 1995.

[**Morris and Fosler-Lussier, 2006**] Jeremy Morris and Eric Fosler-Lussier, "Combining Phonetic Attributes Using Conditional Random

Fields," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Pittsburgh, Pennsylvania, 2006.

[**Morris and Fosler-Lussier, 2008**] Jeremy Morris and Eric Fosler-Lussier, "Conditional Random Fields for Integrating Local Discriminative Classifiers," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 3, 2008.

[**Myers and Rabiner, 1981**] Cory S. Myers and Lawrence R. Rabiner, "A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word Recognition," *The Bell System Technical Journal*, vol. 60, no. 7, 1981.

[**Næss et al., 2011**] Arild Brandrud Næss, Karen Livescu, and Rohit Prabhavalkar, "Articulatory Feature Classification Using Nearest Neighbors," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Florence, Italy, 2011.

[**Narayanan et al., 2011**] Shrikanth S. Narayanan, Erik Bresch, Prasanta Kumar Ghosh, Louis Goldstein, Athanasios Katsamanis, Yoon Kim, Adam Lammert, Michael Proctor, Vikram Ramanarayanan, and Yinghua Zhu, "A Multimodal Real-Time MRI Articulatory Corpus for Speech Research," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Florence, Italy, 2011.

[**Narayanan et al., 2014**] Shrikanth S. Narayanan, Asterios Toutios, Vikram Ramanarayanan, Adam Lammert, Jangwon Kim, Sungbok Lee, Krishna Nayak, Yoon-Chul Kim, Yinghua Zhu, Louis Goldstein, Dani Byrd, Erik Bresch, Prasanta Kumar Ghosh, Athanasios Katsamanis, and Michael Proctor, "Real-Time Magnetic Resonance Imaging and Electromagnetic Articulography Database for Speech Production Research," *The Journal of the Acoustical Society of America*, vol. 136, no. 3, 2014.

[**Neri et al., 2002**] Ambra Neri, Catia Cucchiarini, Helmer Strik, and Lou Boves, "The Pedagogy-Technology Interface in Computer Assisted Pronunciation Training," *Computer Assisted Language Learning*, vol. 15, no. 5, 2002.

[**Omohundro, 1989**] Stephen M. Omohundro, *Five Balltree Construction Algorithms.* International Computer Science Institute, 1989.

[**Oppenheim and Schafer, 2004**] Alan V. Oppenheim and Ronald W. Schafer, “From Frequency to Quefrency: A History of the Cepstrum,” *IEEE Signal Processing Magazine*, vol. 21, no. 5, 2004.

[**Ostendorf, 1999**] Mari Ostendorf, “Moving Beyond the ‘Beads-on-a-String’ Model of Speech,” in *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Keystone, Colorado, 1999.

[**Paul and Baker, 1992**] Douglas B. Paul and Janet M. Baker, “The Design for the Wall Street Journal-Based CSR Corpus,” in *Proceedings of the Human Language Technology Conference (HLT)*, Pacific Grove, California, 1992.

[**Pearl, 1988**] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[**Pearson, 1901**] Karl Pearson, “On Lines and Planes of Closest Fit to Systems of Points in Space,” *Philosophical Magazine*, vol. 2, 1901.

[**Pellom et al., 2001**] Bryan L. Pellom, Ruhi Sarikaya, and John H.L. Hansen, “Fast Likelihood Computation Techniques in Nearest-Neighbor Based Search for Continuous Speech Recognition,” *IEEE Signal Processing Letters*, vol. 8, no. 8, 2001.

[**Povey and Woodland, 1999**] Daniel Povey and Phil C. Woodland, “Frame Discrimination Training for HMMs for Large Vocabulary Speech Recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Phoenix, Arizona, 1999.

[**Prabhavalkar et al., 2011**] Rohit Prabhavalkar, Eric Fosler-Lussier, and Karen Livescu, “A Factored Conditional Random Field Model for Articulatory Feature Forced Transcription,” in *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Big Island, Hawaii, 2011.

[**Price et al., 1988**] Patti Price, William M. Fisher, Jared Bernstein, and David S. Pallett, “The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New York, New York, 1988.

[**Qian et al., 2011**] Yanmin Qian, Ji Xu, Daniel Povey, and Jia Liu, “Strategies for Using MLP Based Features with Limited Target-Language

Training Data," in *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Big Island, Hawaii, 2011.

[**Qin and Carreira-Perpiñán, 2009**] Chao Qin and Miguel Á. Carreira-Perpiñán, "The Geometry of the Articulatory Region That Produces a Speech Sound," in *Asilomar Conference on Signals, Systems, and Computers*, 2009.

[**Qin and Carreira-Perpiñán, 2010**] Chao Qin and Miguel Á. Carreira-Perpiñán, "Articulatory Inversion of American English /ɹ/ by Conditional Density Modes," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Chiba, Japan, 2010.

[**Rabiner, 1989**] Lawrence R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, 1989.

[**Rabiner and Juang, 1993**] Lawrence R. Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[**Richmond, 2006**] Korin Richmond, "A Trajectory Mixture Density Network for the Acoustic-Articulatory Inversion Mapping," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Pittsburgh, Pennsylvania, 2006.

[**Richmond, 2007**] Korin Richmond, "Trajectory Mixture Density Networks with Multiple Mixtures for Acoustic-Articulatory Inversion," in *Advances in Nonlinear Speech Processing (NOLISP)*, Victalonia, Spain, 2007.

[**Rosenblatt, 1962**] Frank Rosenblatt, *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, 1962.

[**Rumelhart et al., 1986**] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, 1986.

[**Sainath et al., 2010**] Tara N. Sainath, Bhuvana Ramabhadran, David Nahamoo, Dimitri Kanevsky, and Abhinav Sethy, "Sparse Representation Features for Speech Recognition." in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Chiba, Japan, 2010.

[**Sainath et al., 2011**] Tara N. Sainath, Bhuvana Ramabhadran, David Nahamoo, and Dimitri Kanevsky, "Reducing Computational Complexities of Exemplar-Based Sparse Representations With Applications to Large Vocabulary Speech Recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Florence, Italy, 2011.

[**Sainath et al., 2012**] Tara N. Sainath, Bhuvana Ramabhadran, David Nahamoo, Dimitri Kanevsky, Dirk Van Compernolle, Kris Demuynck, Jort Florent Gemmeke, Jerome R. Bellegarda, and Shiva Sundaram, "Exemplar-Based Processing for Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, 2012.

[**Sakoe and Chiba, 1978**] Hiroaki Sakoe and Seibi Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, 1978.

[**Salzmann et al., 2010**] Mathieu Salzmann, Carl Henrik Ek, Raquel Urtasun, and Trevor Darrell, "Factorized Orthogonal Latent Spaces," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 2010.

[**Saon and Chien, 2012**] George Saon and Jen-Tzung Chien, "Large-Vocabulary Continuous Speech Recognition Systems: A Look at Some Recent Advances," *IEEE Signal Processing Magazine*, vol. 29, no. 6, 2012.

[**Seide et al., 2011**] Frank Seide, Gang Li, and Dong Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Florence, Italy, 2011.

[**Silverman, 1986**] Bernard W. Silverman, *Density Estimation for Statistics and Data Analysis*, in series Monographs on Statistics and Applied Probability. CRC Press, 1986.

[**Singh-Miller and Collins, 2009**] Natasha Singh-Miller and Michael Collins, "Learning Label Embeddings for Nearest-Neighbor Multi-Class Classification with an Application to Speech Recognition," *Advances in Neural Information Processing Systems (NIPS)*, vol. 22, 2009.

[**Siniscalchi et al., 2007**] Sabato Marco Siniscalchi, Torbjørn Svendsen, and Chin-Hui Lee, "Towards Bottom-Up Continuous Phone Recognition," in *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Kyoto, Japan, 2007.

[**Siniscalchi et al., 2012**] Sabato Marco Siniscalchi, Dau-Cheng Lyu, Torbjørn Svendsen, and Chin-Hui Lee, "Experiments on Cross-Language Attribute Detection and Phone Recognition with Minimal Target-Specific Training Data," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 3, 2012.

[**Siniscalchi et al., 2013b**] Sabato Marco Siniscalchi, Torbjørn Svendsen, and Chin-Hui Lee, "A Bottom-Up Modular Search Approach to Large Vocabulary Continuous Speech Recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, 2013.

[**Siniscalchi et al., 2013a**] Sabato Marco Siniscalchi, Dong Yu, Li Deng, and Chin-Hui Lee, "Exploiting Deep Neural Networks for Detection-Based Speech Recognition," *Neurocomputing*, vol. 106, Apr. 2013.

[**Sivadas and Hermansky, 2004**] Sunil Sivadas and Hynek Hermansky, "On Use of Task Independent Training Data in Tandem Feature Extraction," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Montreal, Canada, 2004.

[**Stenson et al., 1992**] Nancy Stenson, Bruce Downing, Jan Smith, and Karin Smith, "The Effectiveness of Computer-Assisted Pronunciation Training," *Calico Journal*, vol. 9, no. 4, 1992.

[**Stephenson et al., 2000**] Todd A. Stephenson, Hervé Bourlard, Samy Bengio, and Andrew C. Morris, "Automatic Speech Recognition Using Dynamic Bayesian Networks with both Acoustic and Articulatory Variables," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Beijing, China, 2000.

[**Stüker et al., 2003b**] Sebastian Stüker, Florian Metze, Tanja Schultz, and Alex Waibel, "Integrating Multilingual Articulatory Features into Speech Recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Geneva, Switzerland, 2003.

[**Stüker et al., 2003a**] Sebastian Stüker, Tanja Schultz, Florian Metze, and Alex Waibel, "Multilingual Articulatory Features," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Hong Kong, 2003.

[**Subramanya and Bilmes, 2009**] Amarnag Subramanya and Jeff A. Bilmes, "The Semi-Supervised Switchboard Transcription Project," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Brighton, UK, 2009.

[**Sun et al., 2012**] Yang Sun, Bert Cranen, Jort Florent Gemmeke, Lou Boves, and Mathew Magimai-Doss, "Using Sparse Classification Outputs as Feature Observations for Noise-Robust ASR," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Portland, Oregon, 2012.

[**Sundaram and Bellegarda, 2010**] Shiva Sundaram and Jerome R. Bellegarda, "Latent Perceptual Mapping: A New Acoustic Modeling Framework for Speech Recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Chiba, Japan, 2010.

[**Sundaram and Bellegarda, 2012**] Shiva Sundaram and Jerome R. Bellegarda, "Latent Perceptual Mapping with Data-Driven Variable-Length Acoustic Units for Template-Based Speech Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, 2012.

[**Sutton and McCallum, 2011**] Charles Sutton and Andrew McCallum, "An Introduction to Conditional Random Fields," *Foundations and Trends in Machine Learning*, vol. 4, no. 4, 2011.

[**Tang and Rose, 2008**] Yun Tang and Richard Rose, "A Study of Using Locality Preserving Projections for Feature Extraction in Speech Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, Nevada, 2008.

[**Tüske et al., 2013**] Zoltán Tüske, Ralf Schlüter, and Hermann Ney, "Deep Hierarchical Bottleneck MRASTA Features for LVCSR," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[**van der Maaten, 2010**] Laurens van der Maaten, "Matlab Toolbox for Dimensionality Reduction (v0.7.2)," http://lvdmaaten.github.io/drtoolbox/, 2010.

[**van der Maaten et al., 2009**] Laurens van der Maaten, Eric Postma, and Jaap van den Herik, "Dimensionality Reduction: A Comparative Review," Tilburg centre for Creative Computing, Tilburg University (TiCC TR 2009-005), Technical report, 2009.

[**Vinyals and Ravuri, 2011**] Oriol Vinyals and Suman V. Ravuri, "Comparing Multilayer Perceptron to Deep Belief Network Tandem Features for Robust ASR," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[**Wakita, 1979**] Hisashi Wakita, "Estimation of VocaI-Tract Shapes from Acoustical Analysis of the Speech Wave: The State of the Art," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 27, no. 3, 1979.

[**Walpole et al., 2012**] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, and Keying Ye, *Probability and Statistics for Engineers and Scientists*, 9th ed. Pearson, 2012.

[**Watson and Tsoi, 1992**] Brett Watson and Ah Chung Tsoi, "Second Order Hidden Markov Models for Speech Recognition," in *Proceedings of the Australian International Conference on Speech Science and Technology (SST)*, Brisbane, Australia, 1992.

[**Weinberger, 2007**] Kilian Q. Weinberger, "Large Margin Nearest Neighbors (LMNN) toolkit," http://www.cse.wustl.edu/~kilian/code/files/LMNN.zip, 2007.

[**Westbury, 1994**] John R. Westbury, "X-Ray Microbeam Speech Production Database User's Handbook," University of Wisconsin, Madison, Technical report, 1994.

[**Wrench, 2000**] Alan A. Wrench, "A Multi-Channel/Multi-Speaker Articulatory Database for Continuous Speech Recognition Research." in *Phonus 5, Proceedings of the Seminar on Speech Production*, Saarbrücken, Germany, 2000.

[**Wrench and Richmond, 2000**] Alan A. Wrench and Korin Richmond, "Continuous Speech Recognition Using Articulatory Data," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Beijing, China, 2000.

[**Wright et al., 2009**] John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma, "Robust Face Recognition via Sparse

Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, 2009.

[**Young et al., 2009**] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland, *The HTK book (for HTK version 3.4).* Cambridge University Engineering Department, 2009.

[**Yu and Seltzer, 2011**] Dong Yu and Michael L. Seltzer, "Improved Bottleneck Features Using Pretrained Deep Neural Networks," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Florence, Italy, 2011.

[**Zheng et al., 2013**] Xin Zheng, Zhiyong Wu, Binbin Shen, Helen Meng, and Lianhong Cai, "Investigation of Tandem Deep Belief Network Approach for Phoneme Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[**Zweig, 1998**] Geoffrey Zweig, "Speech Recognition with Dynamic Bayesian Networks," Ph.D. dissertation, University of California, Berkeley, 1998.

[**Zweig and Nguyen, 2009**] Geoffrey Zweig and Patrick Nguyen, "A Segmental CRF Approach to Large Vocabulary Continuous Speech Recognition," in *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Merano, Italy, 2009.

[**Zweig and Nguyen, 2010**] Geoffrey Zweig and Patrick Nguyen, "SCARF: A Segmental Conditional Random Field Toolkit for Speech Recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Chiba, Japan, 2010.