



Constrained Non-negative Matrix Factorization for Vocabulary Acquisition from Continuous Speech

Meng SUN

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering

Constrained Non-negative Matrix Factorization for Vocabulary Acquisition from Continuous Speech

Meng SUN

Jury:

Prof. dr. ir. Mr. Hugo Hens, chair

Prof. dr. ir. Hugo Van hamme, promotor

Prof. dr. ir. Tinne Tuytelaars

Prof. dr. ir. Marc Moonen

Prof. dr. ir. Bart De Moor

Prof. dr. ir. Patrick Wambacq

Dr. ir. Martin Heckmann

(Honda Research Institute Europe, Germany.)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering

20 November 2012

© Katholieke Universiteit Leuven – Faculty of Engineering
Kasteelpark Arenberg 10, box 2441, B-3001 Heverlee(Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2012/7515/126
ISBN 978-94-6018-592-2

Preface

It is an amazing and ambitious project to teach computers to learn speech. My work actually focuses on a small part of the whole picture: the mathematical and computational aspect of spoken word learning. Towards the acquisition of speech in computers, there is still a long way to go, but I am sure the journey of exploring those unknown lands will be wonderful and fruitful. I think I am really lucky to have such a research topic to work on in the past four years. On the one hand, I have a great opportunity to extend my knowledge on mathematics and to implement them when investigating the models and algorithms for word learning. On the other hand, the research is on real-world speech signal processing which makes my work stay away from boring intellectual games. I cannot finish this work without the help of my colleagues and friends. I appreciate their help so much!

First of all, I would like to thank my supervisor Prof. Hugo Van hamme for giving me the opportunity to study in Leuven and for his continuous supports. Without his enthusiastic help, I cannot broaden my knowledge quickly, improve my research ability smoothly, nor finish my PhD study successfully.

I would like to thank Dr. Jort Gemmeke for helping revise the thesis. Thanks to Prof. Patrick Wambacq for being my jury and revising my thesis. Thanks to Prof. Dirk Van Compernelle for answering my questions. Thanks to my colleagues, past and present, with whom I had a very pleasant time. I wish all of you a happy life and a bright future.

I also would like to thank Prof. Hugo Hens, Prof. Tinne Tuytelaars, Prof. Marc Monnen, Prof. Bart De Moor and Dr. Martin Heckmann for their willingness to be my jury and their comments on the thesis.

My thanks also go to the China Scholarship Council (CSC), the European Union and KULeuven for providing financial supports to me.

Thanks to my Chinese friends in Leuven for helping me getting familiar with the

life here, for giving me helpful suggestions, for treating me with delicious food and for celebrating traditional festivals together with me. No matter where we are in future, *a bosom friend afar brings a distant land near*.

I am also in gratitude to Prof. Zhengming Wang, my supervisor in home university, who led me into the wonderful world of scientific research at the very beginning. The frequent emails between us encouraged me to move forward.

Finally, thanks to my family, my sister and my girlfriend for their love and unconditional support in my life.

Living and studying in Leuven is definitely a pleasure for me, especially when drinking beers in some “Markts” surrounded by historical buildings, when having a meal at Alma in front of the grass with a beautiful sunset background and when jumping onto the trains to travel through Europe in summer holidays. The relaxing environment inspires fantastic thinking which further triggers creative minds. I will remember the happy time in Leuven forever.

Abstract

One desideratum in designing cognitive robots is autonomous learning of communication skills, just like humans. The primary step towards this goal is vocabulary acquisition. Being different from the training procedures of the state-of-the-art automatic speech recognition (ASR) systems, vocabulary acquisition cannot rely on prior knowledge of language in the same way. Like what infants do, the acquisition process should be data-driven with multi-level abstraction and coupled with multi-modal inputs. To avoid lengthy training efforts in a word-by-word interactive learning process, a clever learning agent should be able to acquire vocabularies from continuous speech automatically.

The work presented in this thesis is entitled *Constrained Non-negative Matrix Factorization for Vocabulary Acquisition from Continuous Speech*. Enlightened by the extensively studied techniques in ASR, we design computational models to discover and represent vocabularies from continuous speech with little prior knowledge of the language to be learned.

Starting with a recently proposed non-negative matrix factorization (NMF) approach to vocabulary acquisition, this work targets at vocabulary representations with high accuracy and fast learning rate. The NMF learning discovers repeated words in spoken data represented as a bag-of-features (BoF) and provides a BoF description of the vocabulary. This thesis advances the state-of-the-art in this area in three respects:

(1) Accuracy improvements of NMF-based word discovery and subsequent ASR using the techniques of soft vector quantization, using multiple codebooks, integrating multiple time scales and using multiple contextual dependencies. Experiments show that the obtained accuracies approach those obtained with hidden Markov models (HMM) trained with transcribed data. However, the above improvements of the NMF model are at the expense of high computational complexity and require sufficient labeled data as supervision. These concerns are addressed in contributions (2) and (3) below.

(2) The NMF method does not constrain its BoF word representations to be generated by actual data sequences. To implement this constraint, the NMF problem is regularized by expressing graph adjacency between the features of the BoF description. In this context, “adjacency” is “temporal proximity”, but the method was also applied successfully to bag-of-feature representations with spatial proximity in image processing. The regularization desirably guides NMF to detect more realistic patterns in the temporal domain. The method demonstrates superior performance over baselines on data sets of speech, images and documents.

(3) The BoF word representations learned by NMF map the acoustic features directly to word activations. With non-negative matrix tri-factorization (NMTF), sub-word units are learned, which improves the representation accuracy of the vocabulary and increases the learning rate in the sense that fewer labeled examples are required. The discovered sub-word units are shown to be closely related to HMM states. Therefore NMF and the NMTF are integrated into a process of non-negative Tucker decomposition (NTD) for unsupervised learning of HMMs. Joint training techniques of NTD and HMMs are proposed for unsupervised sequential pattern discovery. With a few labeled data to link the obtained sequential patterns to grounding labels, the model is able to work as a speech recognizer with better performance than both the unsupervised EM training of HMMs and the NMF model.

Apart from the improvement in accuracy and learning rate, vocabulary acquisition can now result in an HMM, which greatly simplifies the decoding of new utterances in terms of the acquired vocabulary. Like for contribution (2), the resulting method for unsupervised HMM learning is formulated generically with possible applications outside the domain of speech processing.

Abbreviations

ACONRS	Acquisition of Communication and Recognition Skills
AF	Articulatory Feature
ASR	Automatic Speech Recognition
BoF	Bag of Features
BW	Baum-Welch algorithm
CDHMM	Continuous Density Hidden Markov Model
DDHMM	Discrete Density Hidden Markov Model
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
EER	Equal Error Rate
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GNMF	Graph Regularized Non-negative Matrix Factorization
HAC	Histogram of Acoustic Co-occurrences
HMM	Hidden Markov Model
KLD	Kullback-Leibler Divergence
KLDHMM	Kullback-Leiber Divergence Hidden Markov Model
L1GNMF	Graph Regularized Non-negative Matrix Factorization with ℓ_1 Normalization
MFCC	Mel-Frequency Cepstral Coefficient
MLE	Maximum Likelihood Estimation
NMF	Non-negative Matrix Factorization
NMTF	Non-negative Matrix Tri-Factorization
NTD	Non-negative Tucker Decomposition
PCA	Principle Component Analysis

PLSA	Probabilistic Latent Semantic Analysis
SA	Simulated Annealing
SCDHMM	Semi-Continuous Density Hidden Markov Model
SIFT	Scale Invariant Feature Transform
SVD	Singular Value Decomposition
SVM	Support Vector Machine
UWER	Unordered Word Error Rate
VCV	Vowel-Consonant-Vowel database
VQ	Vector Quantization
WER	Word Error Rate

List of Symbols

General Matrix Notations and Operations	
\mathbf{A}	Matrix
\mathbf{A}_j	The j -th column of matrix \mathbf{A}
$A_{i,j}$	The element with row index i and column index j in matrix \mathbf{A}
\mathbf{A}^T	The transpose of matrix \mathbf{A}
$\mathbf{A}^{(\text{mark})}$	Mark of the matrix \mathbf{A} to denote different information streams
\otimes	Kronecker product
\odot	Element-wise multiplication
\oslash	Element-wise division
Reserved Notations for Specific Meanings	
blkdiag	The operation to construct a block diagonal matrix
\mathcal{C}_i	The codeword with index i
\mathbf{G}	The grounding matrix
\mathcal{G}_m	The multivariate Gaussian with index m
Γ	The threshold to make an adjacency matrix binary
\mathbf{H}	The matrix containing the weights of the patterns in NMF
\mathbf{L}	The graph Laplacian
λ	The regularization parameter
$\boldsymbol{\mu}_m$	The mean vector of Gaussian \mathcal{G}_m
\mathbf{O}_t	Observed vector MFCC+ Δ + $\Delta\Delta$ at frame t
\mathbf{Q}	The mapping matrix from unsupervisedly discovered patterns to ground truth
$\boldsymbol{\Sigma}_m$	The covariance matrix of Gaussian \mathcal{G}_m
τ	The number of spaced frames when defining co-occurrences
\mathbf{U}	The adjacency matrix of a graph
\mathbf{V}	The data matrix to be factorized in NMF
\mathbf{W}	The matrix whose columns represent patterns in bag-of-features
\mathbf{X}_t	Representation of \mathbf{O}_t by its posterior probabilities on Gaussians

Notations for HMM:

A	The emission matrix of a DDHMM, the Gaussian weight matrix of a CDHMM and the association matrix of the observation symbols and the hidden states in KLDHMM
O ⁽ⁿ⁾	The <i>n</i> -th observation sequence in the data
π	The initial state distribution of a HMM
Q ⁽ⁿ⁾	The sequence of hidden states aligned for the <i>n</i> -th observation sequence
S _{<i>k</i>}	The hidden state with index <i>k</i>
T	The transition matrix of a HMM

Contents

Abstract	iii
Abbreviations	v
List of Symbols	vii
Contents	ix
List of Figures	xv
List of Tables	xxiii
List of Algorithms	xxvii
1 Introduction	1
1.1 Vocabulary Acquisition	1
1.1.1 Vocabulary Acquisition as a Part of Language Acquisition	2
1.1.2 Grounding for Vocabulary Acquisition	2
1.1.3 Why not Using a Speech Recognizer for Vocabulary Acquisition?	3
1.1.4 Our Focus: Speech Representation for Vocabulary Acquisition	4
1.1.5 Related Fields	5

1.2	Preliminaries in Speech Processing	7
1.2.1	Mel-Frequency Cepstrum Coefficients	7
1.2.2	Hidden Markov Models	10
1.3	Methods for Vocabulary Acquisition	12
1.3.1	Recurring Segment Mining and Clustering	12
1.3.2	Using a Phone Recognizer	13
1.3.3	Latent Variable Models	14
1.3.4	Unsupervised Training of Hidden Markov Models	15
1.4	Goals and Motivations	18
1.4.1	Goal 1: Learning from Continuous Speech	19
1.4.2	Goal 2: Data-driven Models	19
1.4.3	Goal 3: Layered Architectures with Reusable Units	20
1.5	Outline of the Thesis	21
2	An NMF Approach for Vocabulary Acquisition	25
2.1	A General Description of Non-negative Matrix Factorization	25
2.1.1	NMF: Metrics and Algorithms	25
2.1.2	Relations to Other Methods	28
2.2	<i>Bag-of-Features</i> Representation of Speech	32
2.2.1	Mapping a Sequence to a Vector	32
2.2.2	Linear Operations	35
2.3	NMF for Vocabulary Acquisition	35
2.3.1	Training	35
2.3.2	Evaluation Methods	37
2.3.3	Advantages and Disadvantages of the NMF Model	38
3	Advanced BoF Representation of Speech	41
3.1	Multiple Codebooks and Soft VQ	41

3.1.1	Frame Coding Methods	42
3.1.2	Word Acquisition Results	47
3.1.3	Discussion	49
3.2	Multiple Temporal Scales and Asynchronous Streams	51
3.2.1	Motivations and Related Work for Multiple/Variate Frame Rates	51
3.2.2	VCV Corpus and Articulatory Features	53
3.2.3	Consonants Recognition and AF Classification Results	54
3.2.4	Discussion	57
3.3	Multi-gram Models from Gaussian Posteriorgrams of Speech	61
3.3.1	Gaussian Posteriorgram Representation	61
3.3.2	BoF Representations Derived from Gaussian Posteriorgram	62
3.3.3	Computation, Results and Analysis	65
3.4	Conclusion	70
4	Graph Regularized NMF for Unsupervised Pattern Discovery	73
4.1	Unsupervised Pattern Discovery Using NMF	74
4.2	Graph Regularized NMF	76
4.2.1	Preliminaries of Graph Theory	76
4.2.2	Graph Regularization	77
4.2.3	Algorithms	79
4.3	Experiments and Results on Spoken Pattern Discovery	83
4.3.1	Construction of Adjacency Matrices	84
4.3.2	Evaluation Method	86
4.3.3	Parameter Setting and Results	87
4.3.4	Conclusion	88
4.3.5	Comparison of GNMF and L1GNMF	89
4.4	Evaluation of L1GNMF on Other Databases	93

4.4.1	Visual Object Discovery on Caltech256	93
4.4.2	Document Clustering on TDT2	98
4.5	How the Graph Regularization Works	101
4.5.1	Interpreting the Discovered Patterns	101
4.5.2	The Role of Graph Regularization	103
4.6	Conclusion	104
5	Learning of Sub-word Units with Non-negative Matrix Tri-Factorization	107
5.1	Hidden Markov Models of Spoken Words	107
5.1.1	Configuration and Parameters	108
5.1.2	The Role of Hidden States	109
5.2	Tri-Factorization Learning of Sub-word Units	110
5.2.1	Co-occurrence Statistics and Hidden States	110
5.2.2	NMF Learning of the Gaussian Co-occurrence Matrix	113
5.2.3	Non-negative Matrix Tri-Factorization (NMTF)	115
5.3	Speech Representation Using Sub-word Units	117
5.3.1	Matrix Embedding for Dimension Reduction	117
5.3.2	Sequential Labeling	119
5.4	Experiments on Vocabulary Acquisition	122
5.4.1	Experiments with Matrix Embedding	122
5.4.2	Experiments with Sequential Labeling and Blindly Clus- tered Gaussians	126
5.5	Conclusion	131
6	Unsupervised Training of HMMs Using Non-negative Tucker Decomposition	133
6.1	Unsupervised Training of HMMs for Sequential Pattern Discovery	134
6.1.1	HMM Topology and Acoustic Models	134

6.1.2	Training Algorithms	137
6.2	Non-negative Tucker Decomposition for HMM Training	142
6.2.1	Non-negative Tensor Representation of Speech	143
6.2.2	Structured Decomposition for HMM Learning	143
6.3	Methods for Joint Training of NTD and HMMs	146
6.3.1	NTD Regularized BW (NTD.Reg.BW)	146
6.3.2	Alternative Training of NTD and BW (NTD.Alt.BW)	149
6.4	Experiments on TIDIGITS	150
6.4.1	Data Preparation	150
6.4.2	Experiments Using DDHMM	151
6.4.3	Comparison of DDHMM, SCDHMM and KLDHMM	159
6.4.4	Learning from a Few of Labeled Examples and Many Unlabeled Continuous Utterances	163
6.5	Conclusion	166
7	Conclusion	167
7.1	Original Contributions	167
7.2	Future Research Directions	168
7.2.1	An Incremental Learning Framework	168
7.2.2	Hierarchical HMMs	170
A	Convergence of the Contraction Mapping in L1GNMF	173
B	Tensor Notations and Operations	177
C	Derivation of the Symmetric NMTF Algorithm	179
	Bibliography	181
	Curriculum Vitae	193

List of Publications**195**

List of Figures

1.1	An input utterance is composed of samples from its time-domain waveform. Overlapping windows are used to cut the utterance into frames. Every frame is subsequently processed by following the blocks in the figure to yield its cepstrum coefficients.	8
1.2	A three-state discrete density hidden Markov model with a left-to-right structure. A hidden state is modeled by a multinomial distribution over a set of pre-trained observation symbols. The spectrogram of an utterance is vector quantized into the observation symbols. Subsequently, per-frame likelihoods of hidden states are computed for HMM training and decoding.	11
1.3	A three-state continuous density hidden Markov model with a left-to-right structure. A hidden state is modeled by a multivariate distribution. The observations are usually MFCC+ Δ + $\Delta\Delta$ vectors derived from the spectrogram of an utterance. Subsequently, per-frame likelihoods of hidden states are computed for HMM training and decoding.	11
1.4	The DTW alignment between two utterances each of which only contain an English digit “one”. The matrix of local similarity between the feature vectors of the two utterances is shown (white = good match; black = poor match). The blue line shows the optimal alignment between the two utterances.	13
1.5	The speech representation in [89]. Every frame is represented by posterior probabilities of pre-trained phones. Vocabularies are acquired based on the phone-array-represented utterances. . . .	14

1.6	The speech representation in [113] and [98]. Directed arcs of acoustic events (VQ labels or phone identities) are utilized to represent the dynamic properties of speech.	15
1.7	Dynamic HMM net in [70]. The hidden states and their transitions can be added and removed during training.	16
1.8	HMMs of speech units in [54]. It has similar configurations with an ASR system, but with only three states each and trained unsupervisedly.	17
1.9	Unsupervised learning of phone units in [12]. A group of hidden states are firstly trained without considering their transitions. Then transition probabilities are estimated by bootstrapping the training data. Phone models are finally acquired by finding frequent state paths.	17
1.10	An incremental word learning model proposed in [3]. Both labeled and unlabeled data are used for initializing a CDHMM. The parameters will be re-estimated by bootstrapping the training data and a variance floor is estimated and implemented to avoid underestimation of the variances. Finally large margin discriminative training is utilized to improve the recognition performance.	19
2.1	Geometric illustration of PCA and NMF. PCA finds an orthogonal basis where the first principle vector \mathbf{P}_1 lies in the main direction of the data points. NMF tries to find solutions such that the majority of data points can be represented as their convex combinations.	29
2.2	The directed graph for probabilistic latent semantic analysis. The terms (t_i) are generated by hidden topics (z_k) which are further generated from documents (d_j). Here, the “generate” means sampling by following some multinomial distribution. . .	30
2.3	NMF and dictionary learning. A fundamental condition for NMF is the “low-rank” approximation, but dictionary learning does not have this constraint and could yield more bases than data samples.	32

2.4	The flowchart of transforming an utterance into its BoF presentation. MFCC vectors are first extracted and quantized into numbers by using a pre-trained codebook. With a parameter lag , co-occurrences are defined as features. Subsequently, the bag-of-features representation is computed for the utterances and is stored in a column of the data matrix \mathbf{V}	33
3.1	Diagram of HAC. Processing steps from top to bottom: (1) compute MFCC, Δ and $\Delta\Delta$, (2) vector quantization produces a label sequence, (3) compute frequency of co-occurrences of label pairs (HAC), (4) stack in a vector.	43
3.2	Codebooks and Voronoi regions partitioning the data space. Hard VQ can code two close data points by different codewords. With multi-codebooks, the two data points have a chance to be put in the same Voronoi region and thus to have the same codeword. Multi-codebooks can compensate for the accuracy loss in the hard decision with one codebook.	44
3.3	Tradeoff between accuracy and complexity (measured in required memory) for the coding methods on ACORNS-Y2-UK. “Baseline”: hard VQ with single codebook of increasing size. “Hard VQ”: increasing number of codebooks, each of size 250, 250 and 100 (for MFCC, Δ and $\Delta\Delta$). “ $K = 1, \dots, K = 5$ ”: soft VQ with increasing number of codebooks and with $K =$ number of retained softly assigned labels per frame. “Adaptive VQ”: soft VQ with increasing number of codebooks and a variable number of retained softly assigned labels per frame.	50
3.4	General form of a J -stream recognizer with anchor points between speech units (to force synchrony between different streams) [9]. The frame rate and model topology are not necessarily the same for the different streams.	52
3.5	Recognition accuracy with error bars of consonants and plosives. The mean and uncertainty values were computed on the 9 Folds of VCV.	57
3.6	Classification accuracy with error bars of the three articulatory features. The mean and uncertainty values were computed on the 9 Folds of VCV.	58
3.7	Constructing the multi-gram representations from Gaussian posteriorgram of an utterance.	62

- 4.1 The geometric explanation of NMF. Suppose the data is generated by non-negative combinations of “parts”, the “parts” are thus the true solutions as basis vectors. Unsupervised NMF tries to find basis vectors to cover the training data as much as possible. A good solution should also cover the testing data well. 75
- 4.2 The motivation to exploit the adjacency of features. Since the occurring order of the vectors disappears when constructing the BoF, the two different sequences will have the same BoF representation. 75
- 4.3 The illustration of minimizing $\mathcal{F}(\tilde{\mathbf{W}}_k)$ by using its auxiliary function $\mathcal{A}(\tilde{\mathbf{W}}_k, \tilde{\mathbf{W}}_k^t)$ where $\tilde{\mathbf{W}}_k^t$ is the solution from the previous iteration. Both functions are defined on the hyperplane $\|\tilde{\mathbf{W}}_k\|_1 = 1$. $\mathcal{A}(\tilde{\mathbf{W}}_k, \tilde{\mathbf{W}}_k^t)$ is usually chosen as a convex function whose stationary point $\tilde{\mathbf{W}}_k^{t+1}$ is easy to compute. It is straightforward to see the decreasing of $\mathcal{F}(\tilde{\mathbf{W}}_k)$ from $\tilde{\mathbf{W}}_k^t$ to $\tilde{\mathbf{W}}_k^{t+1}$ 80
- 4.4 Adjacency of features. In both figures, the black dots and circles are the features on which the adjacency is defined. \mathbf{X}_t is the posterior probabilities of frame \mathbf{O}_t on the Gaussians. p and P are parameters to define adjacency. Nbest and K are parameters to control sparsity. lag is the parameter to define Gaussians co-occurrences. 86
- 4.5 The effect of not normalizing the columns of the pattern matrix \mathbf{W} from GNMF with $\lambda=1000$ and oracle Gaussians. The mean values of the column sums of \mathbf{W} at each iteration become smaller and smaller with more iterations, which can produce trivial patterns $\mathbf{W}_k=0$ 90
- 4.6 The objective function values and the differences of the objective function values of GNMF and L1GNMF with $R=30$, $\lambda=10000$ and oracle Gaussians. With the normalization at every iteration, the monotonous decreasing of the objective function in GNMF is violated. L1GNMF does not have such a problem. 91
- 4.7 Parameter sensitivity analysis of GNMF and L1GNMF. (a) is with $R=30$, $\Gamma=300$ for speech. (b) is with $R=30$, $\Gamma=200$ for the oracle Gaussians. (c) is with $R=30$ for speech (top) and for image (bottom). (d) is with $R=30$, $\Gamma=200$ for image. This analysis provides evidences for the choices of the regularization parameters in GNMF and L1GNMF. 94

4.8	Definition of graph adjacency between SIFT features. SIFT points are extracted with different scales in (a), so the adjacency of SIFT points depends on the scales and the distance between the two centers in (b).	96
4.9	The mapping matrices obtained on the speech data by NMF and L1GNMF with the same initial \mathbf{W} and \mathbf{H} . The top panel on each figure links the discovered patterns to the digits, while the bottom panel interprets the patterns by genders. From the top panels, we observe that one digit can be linked to several patterns. The bottom panels show that the patterns linked to the same digit can be separated into male and female versions. L1GNMF gives much more clear links than NMF.	102
4.10	The mapping matrices obtained on the image data by NMF and L1GNMF with the same initial \mathbf{W} and \mathbf{H}	102
4.11	10 prototypical images and their corresponding scores for each of the selected patterns.	104
4.12	The comparison between NMF and L1GNMF of the number of images classified correctly per category.	105
4.13	The geometric explanation of graph regularized NMF. Compared to Figure 4.1, the graph regularization can adjust the NMF solutions by grouping adjacent points together. The obtained solutions \mathbf{W}'_k would perform better than the NMF solution \mathbf{W}_k in the sense of covering training and testing data points.	105
5.1	Comparison between a HMM and a HAC-NMF model of a spoken word. In a HMM, a <i>word</i> is modeled by <i>hidden states</i> while a <i>hidden state</i> is modeled by a GMM, i.e. <i>Gaussians and weights</i> where the Gaussians are the probability distributions on the observation frames. However, in HAC-NMF, though we have introduced <i>posterior probabilities on Gaussian pairs</i> as a bigram model in Chapter 3, the layer of hidden states is missing.	108
5.2	The relationship between Gaussian co-occurrences and hidden states. Every hidden state is modeled by a multinomial distribution on the Gaussians. Hence, the Gaussian co-occurrence is modeled by <i>the co-occurrences of hidden states</i> and <i>the emission of Gaussians from every hidden state</i>	111

- 5.3 The flowchart of the proposed learning methods. Two tracks are presented where *Track A* is for supervised learning of hidden units and *Track B* is for unsupervised learning of hidden units. 112
- 5.4 Learning of hidden states from the matrix \mathbf{W} learned by NMF. A column of \mathbf{W} is the bag of co-occurrences of features for a discovered pattern. \mathbf{W} can have multiple lags τ which inherited from the data matrix \mathbf{V} as presented in Eq.(2.21). Taking the block of \mathbf{W} corresponding to lag τ , $\mathbf{W}^{(\tau)}$, as an example, a column of $\mathbf{W}^{(\tau)}$, $\mathbf{W}_k^{(\tau)}$, is first reshaped to a co-occurrence matrix $\mathbf{C}^{(\tau,k)}$. NMTF is subsequently performed on the co-occurrence matrix to learn hidden states. 114
- 5.5 The graphical model between Gaussians and hidden states learned by the symmetric NMTF. The weights of the Gaussians for the hidden state S_k are learned and stored in a column vector $\mathbf{A}^{(r)}$. The vectors are stacked to form the full weight matrix. . 114
- 5.6 Transformation of the co-occurrences of Gaussians in utterance n with lag τ , $\mathbf{V}_n^{(\tau)}$, to co-occurrences of hidden units in $\mathbf{E}_n^{(\tau)}$. vec is the operator to reshape a matrix to a vector by stacking the columns while vec^{-1} is its inverse. 118
- 5.7 Sequential labeling using hidden units learned by NMTF. The frames of an utterance is labeled from left to right. Three types of labeling scores are merged: the forward transition, the observation and the backward transition. 119
- 5.8 The co-occurrences of the hidden units of digit “one”, $R_1=35$ units per digit, from \mathbf{W} with $K_1=5$ 125
- 5.9 The NMTF learning of HMM states and Gaussian weight matrices with $\tau=2$. $\mathbf{A}^{(\tau,r)}$, $\mathbf{B}^{(\tau,r)}$ and $\mathbf{D}^{(\tau,r)}$ are learned by NMTF: $\mathbf{C}^{(\tau,r)} \approx \mathbf{A}^{(\tau,r)}\mathbf{B}^{(\tau,r)}\mathbf{D}^{(\tau,r)}$. The example corresponds to digit “one”. 126
- 5.10 The comparison of performance on vocabulary discovery between Gaussians and sub-word units. 129
- 5.11 The comparison of performance on vocabulary discovery from TIDIGITS between Gaussians and sub-word units. Supervisory information is utilized to identify vocabulary patterns and garbage patterns, based on which 10 sub-word units are learned for the vocabulary patterns and 3 units are learned for the ones related to garbage or silence. 129

5.12	Representations of utterance “4625” in posterior probabilities on the learned hidden units. The corresponding spectrogram is also shown for comparison.	130
5.13	Posteriorgrams on learned hidden units for comparing the proposed sequential labeling and the Viterbi labeling.	131
5.14	Comparison of performance on vocabulary discovery between Gaussians, sequential labeling and Viterbi labeling of hidden units.	132
6.1	The HMM topology used for sequential pattern discovery. Each parallel branch is called a sub-HMM.	135
6.2	Non-negative Tucker decomposition of the data tensor \mathbf{X} to yield hidden states and sequential patterns modeled by sub-HMMs simultaneously.	144
6.3	Structured non-negative Tucker decomposition $\mathbf{X} \approx \mathbf{B} \times_1 \mathbf{A} \times_2 \mathbf{A} \times_3 \mathbf{H}^T$ in three stages: (top) NMF to extract recurrent sequential patterns $\{\mathbf{C}_{(3)}^{(r)}, r = 1, \dots, R\}$, (middle) NMTF is applied subsequently to learn a sub-HMM for each pattern r , (bottom) stacking the R emission matrices in one emission matrix (left) and expanding the R transition matrices with zeros and stacking these in the core tensor as mode-3 slices (right).	144
6.4	The alternating training framework of NTD.Alt.BW. The current HMM parameters are transformed to their tensor form by φ^{-1} to initialize the structured tensor decomposition. Then the tensor parameters are transformed back to HMM parameters by φ and α Baum-Welch iterations are performed. This process is repeated several times.	149
6.5	Convergence analysis. Top: log-likelihood of the training data for NTD.Alt.BW and initializing BW with only one NTD (NTD.Init.BW). Middle: recognition accuracy in % of the intermediate HMMs. Bottom: accuracy in % of the intermediate NMF models.	154
6.6	Mapping matrices between the sub-HMMs and the digits for four different methods initialized equally ($R = 30, L = 10$).	156
6.7	The performance of BW+SA and NTD.Alt.BW with respect to numbers of hidden states and number of sub-HMMs. *: BW+SA with 5 states, \circ : NTD.Alt.BW with 5 states, +: BW+SA with 10 states and \diamond : NTD.Alt.BW with 10 states.	159

6.8	The performance of NTD.Reg.BW with respect to the regularization parameter λ . The number of sub-HMMs is $R = 30$ and the number of states per sub-HMM is $L = 10$. The NTD.Reg.BW and BW are with 25 BW iterations, while the BW+SA is with 125 iterations.	160
6.9	Mapping matrices from the BW training and the NTD.Alt.BW training of SCDHMM. The BW training is with 25 EM passes, while the NTD.Alt.BW training methods are with 5 NTD passes and 5 BW passes.	162
6.10	Mapping matrices from the BW training and the NTD.Alt.BW training of KLDHMM. The BW training is with 25 EM passes, while the NTD.Alt.BW training methods are with 5 NTD passes and 5 BW passes.	163
6.11	Mapping matrices from the Viterbi training and the NTD.Alt.Vit training of KLDHMM. The parameters and initializations are the same as the BW training of SCDHMM and KLDHMM. The Viterbi training is with 25 EM passes, while the NTD.Alt.Vit training methods are with 5 NTD passes and 5 Viterbi passes.	164
7.1	Incremental word learning by adding sub-HMMs. New spoken patterns are first detected, subsequently modeled by a sub-HMM and added to the overall HMM as a new branch.	169
7.2	A hierarchical HMM for speech recognition. The model has multiple layers of hidden states. CDHMM and KLDHMM can be implemented for initializing the bottom layers. Supervision is only available for the top layer which is trained with transcribed data.	172
B.1	Tensor notations and operations.	178

List of Tables

3.1	EER (%) of NMF with multi-codebooks on ACORNS-Y2-UK	48
3.2	EER (%) of NMF with soft VQ on ACORNS-Y2-UK	49
3.3	EER (%) of NMF with adaptive VQ on ACORNS-Y2-UK	49
3.4	Indices of experiments on VCV	54
3.5	Accuracy (%) of consonant recognition and AF classification on Fold 1 of VCV	55
3.6	Average accuracy (%) of AF value classification on testset of <i>Fold1</i> of VCV. For single feature streams, only the ones yielding good performance (<i>SS,MM,LS</i>) are listed.	56
3.7	Training and recognition (accuracy in %) with different shift on testset of <i>Fold1</i> of VCV	60
3.8	Consonant recognition rates (%) on the 9 folds of VCV using multi-codebooks	60
3.9	UWER on TIDIGITS using the NMF unigram model	66
3.10	UWER on TIDIGITS using the NMF bigram model (FeatSel-1)	66
3.11	UWER on TIDIGITS using the NMF trigram model (FeatSel-1 and FeatSel-3). The NMF dimension R is 12. For each K , within the computational budget, we selected as small as a threshold to retain as much as information.	66
3.12	UWER of the NMF bigram model with pruning of Gaussian co-occurrences (FeatSel-1 and FeatSel-2)	67

3.13	UWER of the NMF trigram model with pruning of Gaussian co-occurrences (FeatSel-1 and FeatSel-2)	68
3.14	UWER of the Bigram+Trigram model with FeatSel-1, FeatSel-2 and FeatSel-3	69
3.15	Comparison of the bigram part and the trigram part of the learned pattern matrix	69
3.16	UWER of the NMF multi-gram models with 100 Gaussians from a CDHMM (FeatSel-1 and FeatSel-3)	70
4.1	UWER of NMF, GNMF and L1GNMF with the unigram model (500 Gaussians) on TIDIGITS.	88
4.2	UWER of NMF, GNMF and L1GNMF with the bigram model (200 Gaussians) on TIDIGITS.	88
4.3	UWER of NMF, GNMF and L1GNMF with oracle Gaussians on TIDIGITS.	90
4.4	Computation counts for each main iteration of NMF, GNMF and L1GNMF	92
4.5	Comparison of CPU time required for 500 iterations of NMF, GNMF and L1GNMF. The values (in seconds) were averaged over 50 random tests.	93
4.6	20 object categories selected from Caltech256	95
4.7	Classification error rates of NMF, GNMF and L1GNMF on Caltech256.	98
4.8	Performance of NMF,GNMF and L1GNMF on TDT2	100
5.1	Comparison of the oracle Gaussians and the learned hidden units by NMTF on word learning with the grounded NMF learning.	124
5.2	UWER (%) obtained by using the refined hidden units as features. The baselines from Gaussians are also shown. The overview on all N_1 's can be found in Figure 5.11. This table only shows the results when zooming into the tails of the two plots in Figure 5.11.	130

- 6.1 Evaluation of BW, BW+SA, NTD.Reg.BW and NTD.Alt.BW for unsupervised learning of discrete density HMMs: log-likelihood of the training data, segmentation accuracy in %, model purity in % and recognition accuracy in %. 153
- 6.2 Evaluation of the JLH method: 25 JLH iterations, initialized from the BW and NTD.Alt.BW models respectively ($R = 30, L = 10$). 157
- 6.3 Details of the two-tailed paired t -test: $H_0 : \mu_0 = \mu_1$ v.s. $H_1 : \mu_0 \neq \mu_1$. h is the result of the test: $h=0$ indicates that the null hypothesis cannot be rejected at the 5% significance level, while $h=1$ indicates that the null hypothesis can be rejected at the 5% level. p is the probability of observing the given result or a more extreme value, if the null hypothesis is true. Small values of p cast doubt on the validity of the null hypothesis. ci is a 95% confidence interval for the true mean, or of $\mu_1 - \mu_0$ for a paired test. 161
- 6.4 The performance on TIDIGITS of NTD.Reg.BW training of DDHMM. $M = 1000$ Gaussians, $R = 12$ sub-HMMs, $L=10$ states for each sub-HMM and 25 EM iterations. First column: Baum-Welch training, second and third column: NTD.Reg.BW with listed regularization weight. 165
- 6.5 The performance on TIDIGITS of NTD.Reg.BW training of SCDHMM. $M = 1000$ Gaussians, $R = 12$ sub-HMMs, $L=10$ states for each sub-HMM, 25 EM iterations. First column: Baum-Welch training, second and third column: NTD.Reg.BW with listed regularization weight. 165
- 6.6 The performance on TIDIGITS of NTD.Reg.BW training of KLDHMM. $M = 1000$ Gaussians, $R = 12$ sub-HMMs, $L=10$ states for each sub-HMM, 25 EM iterations. 166

List of Algorithms

4.1	Updating algorithm of L1GNMF	83
5.1	Pseudo code for asymmetric NMTF $\mathbf{C} \approx \mathbf{A}\mathbf{B}\mathbf{D}$	116
5.2	Pseudo code for symmetric NMTF $\mathbf{C}^{(\tau)} \approx \mathbf{A}\mathbf{B}^{(\tau)}\mathbf{A}^T$ with multiple contextual dependencies τ	117

Chapter 1

Introduction

1.1 Vocabulary Acquisition

Speech is believed to be the most natural and convenient way to communications between humans and in research on automatic speech recognition (ASR) we assume speaking to machines also has sufficient advantages to merit our efforts. However, there is still a large space for improvement towards smooth interaction between human and machines (like computers and robots) using speech. With today's technology, establishing of a communication between a human and a machine involves prior knowledge about *language* and the *context of the communication*. Language-specific knowledge includes the phone set and its acoustic realization, the vocabulary and its realization in terms of phones and finally the grammar or how words are placed in a sentence. These language specifics involve design and supervised training procedures. Apart from the generic language-specific knowledge, words and sentences have a meaning in the application-specific context. For instance, if you instruct a service robot to “fetch a Jupiler from the fridge”, the words “fetch”, “Jupiler” and “fridge” have a specific meaning in your home. The ability of understanding and responding to speech by a robot is actually based on application-specific computer programs, which seem difficult to adapt to new circumstances. Therefore, a cognitive robot should possess the ability of learning language autonomously, i.e. should have the ability of *language acquisition*.

1.1.1 Vocabulary Acquisition as a Part of Language Acquisition

Infants begin to acquire language without any prior knowledge about grammar, words or speech sounds when they are confronted with the first communicative utterances produced by their care givers. It is assumed by “Emergentist theories” that language acquisition in humans is a cognitive process that emerges from the interaction of biological pressures and the environment. “The proponents of these theories argue that general cognitive processes subserve language acquisition and that the end result of these processes are language-specific phenomena, such as *vocabulary acquisition* and *grammar induction*” [1] [69]. In the above example, to understand the instruction “fetch a Jupiler from the fridge”, the robot should first acquire the vocabulary, e.g. “fetch” is an action to go somewhere and carry something back, “Jupiler” is a kind of beer and “fridge” is something to store beers in. This ability is *vocabulary acquisition*. Based on acquired vocabularies, the robot should also understand the rules or grammars by which the words are organized to convey special meanings, which is *grammar induction*.

Inspired by the success of human language acquisition, computational models for vocabulary acquisition are proposed in [11][89][54][45][34][92]. Towards such kind of models, approaches to bridge sensory signals and meaningful language symbols (e.g. phones, words, phrases) play a dominant role. Specifically, since *words* are the elementary units of a spoken language in the sense of naming things and their relations, the first task in language acquisition is *vocabulary acquisition* on which *grammar induction* is based. Hence, out of the multiple aspects of language acquisition, we focus on the study of *vocabulary acquisition*.

1.1.2 Grounding for Vocabulary Acquisition

An important first step towards vocabulary acquisition from speech is discovering acoustic cues. However, this by itself cannot complete the task of vocabulary acquisition. A second important step is to find a meaning for each discovered acoustic cue. In this second step, the robot relates the cues with other input modalities, e.g. from its visual or tactile sensors. This process is referred to as *grounding* [11][113][22][72]. The grounding associations can be based on frequency of co-occurrences [11][113][35], or cross-entropy of two events [89]. For example, every time the robot hears the sound “Jupiler” when it sees a bottle of beer with a particular label, the connection between the sound and the visual object will be strengthened. This process repeats until the meaning of “Jupiler” is fully acquired when the sound is linked to its meaning.

Computational experiments with grounding imply computational models in other modalities, such as building object recognizer in images or motor control and proprioception. This is beyond the scope of this thesis which focuses on the computational aspects of speech. We will therefore idealize the object recognizers and label utterances with tags, i.e. use symbolic representations of the non-speech modalities. *Grounding of words* is then relating the acoustic word models with tags.

1.1.3 Why not Using a Speech Recognizer for Vocabulary Acquisition?

In computer science, the transcription process from spoken words into text is called automatic speech recognition (ASR) while a program to fulfill this process is called an automatic speech recognizer [52]. As is explained in section 1.1.2, the role of grounding is actually modeling the associations between acoustic inputs and other modalities to endow a sound some meaning. Now with a well-trained automatic speech recognizer, those associations seem to be able to be acquired via the textual format of the acoustic inputs. That is, the robot can first transcribe the sensory speech signals of the instruction to a sequence of textual words: /fetch/-/a/-/Jupiler/-/from/-/the/-/fridge/ by using an automatic speech recognizer. Then the output text /Jupiler/ will be associated with a particular bottle appearing in the scene. Certainly, the textual word “Jupiler” and its acoustic realizations should be *pre-defined* in the automatic speech recognizer, otherwise the recognizer will fail to transcribe this part of the speech signal. Normally, to set up an ASR system, the phone sets, vocabularies and grammars are supposed to be known beforehand and a sufficient amount of training utterances and their associated transcriptions (word sequences) should also be given. Under these conditions, the state-of-the-art ASR system has obtained quite promising results [125]. Models for word-meaning learning based on ASR have also been studied in [22][72][92].

ASR can be helpful to the learning of word meaning but by itself it is not suitable for vocabulary acquisition. The aims of the two problems are different where speech recognition is to recognize the *pre-defined* words while vocabulary acquisition is to learn *new* grounded words. In vocabulary acquisition the vocabularies which the data contains are not available beforehand, nor are the word boundaries in continuous speech, where the only available information might be that *an utterance starts and ends with silence (or non-speech)*. There could be a large amount of training data, but the part with word labels would be small. Also, sometimes grounding information as supervision may only occur from utterance-level labels stemming from detected events in other modalities

like video inputs [89] or word tags as a simplification of visual object detection [108].

However, ASR can serve vocabulary acquisition. Given the success of ASR over the past decades, the acoustic features, statistic models and learning architectures in ASR can still be modified and implemented in vocabulary acquisition as will be described in section 1.2 and section 1.3. Taking vocabulary acquisition as an incremental process, ASR would be an internal stage which happens after the robot acquires a couple of words. That is, the agent would first acquire some words, then do speech recognition to refine the acquired words and to detect and learn new words.

On the other hand, research on speech recognition can also benefit from vocabulary acquisition. For example, in the instruction “fetch a Jupiler from the fridge”, “Jupiler” may not be in the dictionary of a robot borrowed from your friend who prefers “Stella”. So the word “Jupiler” should be learned by the robot from your conversations involving “Jupiler”. For a conventional ASR system, new models have to be trained with sufficient transcribed data to cover the out-of-vocabulary words, which requires human intervention to generate hand labels [83, 129, 108, 56] and sometimes it is difficult to know the correct transcription labels. This difficulty is more salient in the analysis of speech in underresourced languages or dialects [130]. With the techniques of vocabulary acquisition, a few transcribed speech segments would be sufficient to aid the robot to acquire new words. Furthermore, vocabulary acquisition can also be useful in the analysis of untranscribed databases to complement the transcribed corpora [58][61][71].

1.1.4 Our Focus: Speech Representation for Vocabulary Acquisition

The grounding process itself deserves careful research towards successful word-meaning learning. However, the representation of input signals in each modality (e.g. audio or visual) provides fundamental information to be associated in the grounding stage. The main focus of the thesis is hence to design computational methods for vocabulary acquisition by investigating speech representation approaches. Word tags are utilized as grounding for the learning of acoustic representations of vocabularies. The tags can be replaced by input representations of other modalities like vision, so the tags can be called “pseudo-visual inputs” in [11][113]. Admittedly, this symbolic representation may not suffice to represent all meaning in language. Fuzzy class memberships, for one, may require more elaborate representations. Also, we will not deal with modeling *word order*, i.e. with learning the grammar that the acquired words

occur in. Our studies are limited to acquiring and grounding a few tens of words, where it is reasonable to assume discrete utterance-level tags suffice to represent meanings.

Applications of the investigated methods:

The motivation for studying speech representation with weak or no supervision is fourfold. Firstly, it can serve as fundamentals to model early language acquisition of infants where they acquire spoken units from exposure to the spoken language. These units are subsequently related to multi-modal observations which is provided by the grounding supervision. Secondly, a similar learning problem is created when one wants to communicate with robots in natural language to assign them a task. Such instructions are bound to contain environment-specific vocabularies that the robot needs to learn from interaction with humans. Thirdly, unsupervised learning does not require transcriptions of speech, so the huge human effort to generate hand labels is avoided. Finally, unsupervised vocabulary acquisition can be applied in modeling out-of-vocabulary words and in the analysis of speech in underresourced languages or dialects as discussed in section 1.1.3.

In a more general sense, the essential problem studied is sequential data analysis. The involved tasks are discovering sequential patterns, clustering, classification and segmentation of sequences. Therefore, the study on representation methods of sequential data would perform as general tools for processing any sequential data, not only speech. For example, mining topics from textual documents, discovering visual objects from images, gene finding from DNA sequences and unsupervised part-of-speech tagging of documents.

1.1.5 Related Fields

Data mining

The aim of vocabulary acquisition, i.e. *discovering words from speech*, is similar to data mining from speech databases whose target is *discovering knowledge from speech data*. Both of them involve data preprocessing, modeling, optimization, and post-processing of the found structures. Consequently, some techniques used in data mining can be applied to vocabulary acquisition. In contrast to general data mining techniques, the methods we have applied to vocabulary acquisition prefer *content-based* and *non-negative* representations of speech; while conventional data mining methods on speech signals do not have to.

- Content based representations

The content-based representation refers to storing the features related to a word, not only the features to discriminate one acoustic recording from others. For example, in the extraction of audio fingerprints in data mining, average zero crossing rate, estimated tempo, average spectrum, spectral flatness, prominent tones across a set of frequency bands can be selected to identify audio files [16]. However, they are not suitable for vocabulary acquisition because their temporal resolution is not adequate to represent the content, i.e. which words are used.

- Non-negative representations

The property of non-negativity originates from neural networks by virtue of two properties: the firing rates of neurons are never negative and synaptic strengths do not change sign. Non-negativity has been proven to be useful to obtain parts-based representations from data [64].

Furthermore, sufficient prior language knowledge (e.g. phone lattices) or an available speech recognizer can be applied to data mining from speech [109], but not to modeling human vocabulary acquisition.

Machine learning

The ultimate application-oriented goal of vocabulary acquisition is to teach a robot new spoken words for further language learning. Hence vocabulary acquisition is basically a process of machine learning.

- The property of little supervision or no supervision implies it can be classified as *weakly supervised learning* [28] or *unsupervised learning* [38][112][42].
- The acquired spoken terms should be related to other modalities. Thus it is *multi-modal learning*, i.e. joint training with the inputs of texts, images, etc. [4][126].
- Before joint training with (weak) supervisory information, the model can generate hierarchical units without supervision to reflect the underlying structure of speech data and thus to facilitate the upcoming supervised learning or accelerate the learning rate. This hierarchical learning architecture seems to relate to *deep learning* [46][6][127].
- Given the predefined words, ASR is a form of sequential pattern recognition. Unlike in ASR or other pattern recognition tasks, there are no predefined patterns in vocabulary acquisition, so the method has

to discover patterns before recognizing them. The process of pattern discovery and recognition can be organized in an iterative way: discover spoken patterns \rightarrow sequential pattern recognition \rightarrow interface with multi-modal grounding \rightarrow update the spoken patterns \rightarrow sequential pattern recognition \rightarrow ...

Besides the above research fields, vocabulary acquisition also shares common techniques with *signal processing* for the front-end time-frequency feature extraction. For example, blind source separation can be applied to identify different speakers, and their voice activities over time, etc.

Furthermore, a high-level vocabulary acquisition would be involved in *language models* for the back-end interpretation of the discovered spoken patterns. After acquiring a sufficient number of words, the robot would be able to learn the organizing rules or grammars of the words. Word order would be taken into account to express various meanings. Complex multi-modal information, such as actions, can therefore be learned.

In summary, vocabulary acquisition is to find meaningful spoken patterns with techniques from ASR, data mining, machine learning and signal processing.

1.2 Preliminaries in Speech Processing

Before discussing the related work, we will first describe the extraction of acoustic features and the HMM framework on which state-of-the-art systems are based.

1.2.1 Mel-Frequency Cepstrum Coefficients

The mostly utilized short-time representation of speech signals in ASR systems are the Mel-Frequency Cepstrum Coefficients (MFCC). Figure 1.1 describes the pipeline to extract MFCC where the blocks correspond to the following operations.

After sampling the time-domain waveform, the speech signal x_t is processed by a pre-emphasis filter, which results in a 6dB/octave increase in gain of the high frequencies. At a sampling frequency of 16k Hz, the filter $H(z) = 1 - 0.97z^{-1}$ is a suitable choice. This counteracts the attenuation effect at these frequencies owing to the glottal source in voiced speech and will make the average speech

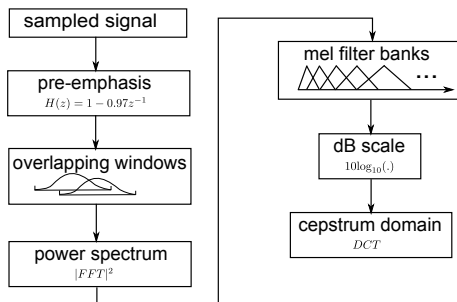


Figure 1.1: An input utterance is composed of samples from its time-domain waveform. Overlapping windows are used to cut the utterance into frames. Every frame is subsequently processed by following the blocks in the figure to yield its cepstrum coefficients.

spectrum roughly flat.

$$x_t \leftarrow x_t - 0.97x_{t-1} \quad (1.1)$$

Subsequently, the speech signal is segmented into overlapping *frames* to capture its time varying characteristics. The window length is normally selected around 20~30 ms during which speech is assumed to be quasi-stationary, and a frame shift 10 ms is applied for hopping the windows in the state-of-the-art ASR systems. Suppose the discrete samples in a frame are $\{x_t\}_{t=1}^T$. A windowing function (e.g. Hamming window, $w_t = 0.54 - 0.46\cos(2\pi\frac{t-1}{T})$) is operated on the frame to reduce the anomalies due to the framing boundaries.

$$x_t \leftarrow w_t x_t \quad (1.2)$$

The short-term power spectrum or *spectrogram* is computed by taking the squared modulus of its Discrete Fourier Transform (DFT).

$$|X_k|^2 = \left| \sum_{t=1}^T x_t e^{-i2\pi\frac{t-1}{N}k} \right|^2 \quad (1.3)$$

where N is the number of DFT frequencies which is usually a power of two.

A Mel-scaled filter bank is applied to the power spectrum to reflect the characteristics of the human hearing system. The frequency resolution of the human ear performs nearly linearly in the low-frequency area, but performs approximately logarithmic in the high frequency regions,

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (1.4)$$

where f is the frequency in Hz and m is the corresponding Mel-frequency. To model this property, a group of triangular amplitude responses is applied to wrap the energy distribution along the frequency axis. To obtain the filter banks, we first divide the Mel scale into D equally spaced bins. Each bin has a central Mel-frequency $m_d = \frac{d}{D+1}M_S, d = 1, \dots, D$ where $M_S = 2595 \log_{10}(1 + \frac{F_S/2}{700})$ and F_S is the sampling frequency. The corresponding central frequencies are thus $f_d = 700(10^{m_d/2595} - 1)$. So the d -th Mel filter bank is constructed as,

$$g(f) = \begin{cases} \frac{f-f_{d-1}}{f_d-f_{d-1}}, & f_{d-1} \leq f \leq f_d \\ \frac{f_{d+1}-f}{f_{d+1}-f_d}, & f_d < f \leq f_{d+1} \\ 0 & \text{elsewhere} \end{cases} \quad (1.5)$$

where $f_0 = 0$ and $f_{D+1} = F_S/2$. Subsequently, with the discrete version of $g(f)$,

$$g_k = g\left(\frac{k}{N}F_S\right), \quad (1.6)$$

the wrapping is made for each filter bank to get D Mel frequency spectrum coefficients.

$$S_d = \sum_{k=1}^{T/2} g_k |X_k|^2 \quad (1.7)$$

Log spectrum is applied to compress the scale of the Mel spectrum.

$$\log S_d = 10 \log_{10}(S_d) \quad (1.8)$$

Subsequently, the log power spectrum is decorrelated by a discrete cosine transformation (DCT).

$$y_s = \sum_{d=1}^D \log S_d \cos\left(2\pi s \frac{d-1}{D}\right) \quad (1.9)$$

where $s = 0, \dots, S-1$.

Hence, a frame \mathbf{x}_t is finally transformed to its MFCC vector \mathbf{y}_t . To reflect the dynamic characteristics of speech, besides the static MFCC vector \mathbf{y}_t , the velocity (Δ) and acceleration ($\Delta\Delta$) vectors are computed and stacked to form a compounded representation MFCC + Δ + $\Delta\Delta$ for each frame, called an observation vector \mathbf{O}_t .

$$\begin{aligned} \Delta \mathbf{y}_t &= \frac{1}{2 \sum_{\theta=1}^{\Theta} \theta^2} \sum_{\theta=1}^{\Theta} \theta (\mathbf{y}_{t+\theta} - \mathbf{y}_{t-\theta}) \\ \Delta\Delta \mathbf{y}_t &= \frac{1}{2 \sum_{\theta=1}^{\Theta} \theta^2} \sum_{\theta=1}^{\Theta} \theta (\Delta \mathbf{y}_{t+\theta} - \Delta \mathbf{y}_{t-\theta}) \end{aligned} \quad (1.10)$$

1.2.2 Hidden Markov Models

A hidden Markov model (HMM) is usually selected to describe a speech unit (e.g. a phoneme, word) as a sequence of parts with similar acoustic properties which are called *states* in the model. A HMM is a graphical model that generates a sequence of observation vectors while following a first order Markov assumption: the probability of being in a state q_t at frame t depends only on the previous occupation of state q_{t-1} at frame $t-1$. The transition probability between two states is given by $\Pr(q_t|q_{t-1})$.

The time-frequency observation \mathbf{O}_t is assumed to be generated by a stochastic emission process that is independent of the transition process and only depends on the current state q_t , i.e. is described completely by $f(\mathbf{O}_t|q_t)$ (the emission distribution). The term “hidden” refers to the fact that the underlying state can in general not be uniquely determined from the observations. In discrete density HMMs (DDHMM), observation vectors are first quantized into observation symbols and $f(\mathbf{O}_t|q_t)$ is a discrete distribution over the observation symbol set. In continuous density HMMs (CDHMM), $f(\mathbf{O}_t|q_t)$ is a multivariate distribution directly on the observation vectors. Examples of HMMs with a left-to-right topology together with its emission density functions and state transition probabilities are illustrated in Figure 1.2 for a discrete density HMM and in Figure 1.3 for a continuous density HMM.

The probability for the observations sequence $\mathbf{O} = \{\mathbf{O}_t\}_{t=1}^T$ given HMM Λ is,

$$\Pr(\mathbf{O}|\Lambda) = \sum_{q_1, \dots, q_T} \Pr(q_1) f(\mathbf{O}_1|q_1) \prod_{t=2}^T \Pr(q_t|q_{t-1}) f(\mathbf{O}_t|q_t) \quad (1.11)$$

where \mathbf{O}_t is the MFCC+ Δ + $\Delta\Delta$ vector at frame t and $\{q_t\}_{t=1}^T$ is any hypothesized state sequence.

A diversely utilized method to construct an emission distribution function $f(\mathbf{O}_t|S_k)$ for state S_k is the Gaussian mixture model (GMM) where the distribution function is the weighted sum of Gaussians. In the case where all hidden states share the same set of Gaussians $\{\mathcal{G}_1, \dots, \mathcal{G}_M\}$, the emission function is,

$$f(\mathbf{O}_t|S_k) = \sum_{m=1}^M \omega_{m,k} \Pr(\mathbf{O}_t; \mathcal{G}_m) \quad (1.12)$$

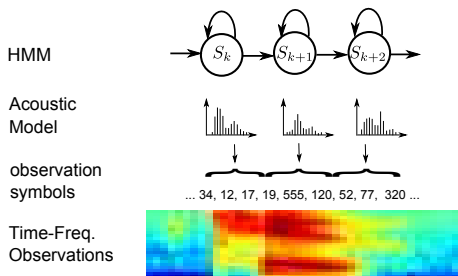


Figure 1.2: A three-state discrete density hidden Markov model with a left-to-right structure. A hidden state is modeled by a multinomial distribution over a set of pre-trained observation symbols. The spectrogram of an utterance is vector quantized into the observation symbols. Subsequently, per-frame likelihoods of hidden states are computed for HMM training and decoding.

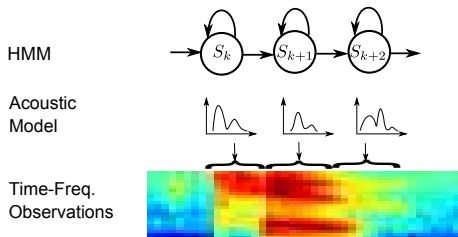


Figure 1.3: A three-state continuous density hidden Markov model with a left-to-right structure. A hidden state is modeled by a multivariate distribution. The observations are usually MFCC+ Δ + $\Delta\Delta$ vectors derived from the spectrogram of an utterance. Subsequently, per-frame likelihoods of hidden states are computed for HMM training and decoding.

where $\omega_{m,k}$ is the weight of Gaussian \mathcal{G}_m with the constraint $\sum_{m=1}^M \omega_{m,k} = 1$ and $\Pr(\mathbf{O}_t; \mathcal{G}_m)$ is the probability of observing \mathbf{O}_t from Gaussian \mathcal{G}_m . That is,

$$\Pr(\mathbf{O}_t; \mathcal{G}_m) = \frac{1}{\sqrt{(2\pi)^D \prod_d \sigma_{d,m}^2}} \exp\left(-\sum_d \frac{(O_{d,t} - \mu_{d,m})^2}{2\sigma_{d,m}^2}\right). \tag{1.13}$$

The covariance matrices of the Gaussians are usually taken to be diagonal, which will reduce the number of parameters with respect to the full diagonal choice. The DCT decorrelation in MFCC extraction ensures that despite the diagonal covariance, a low number of mixture components will result in a good fit of the emission densities. HMMs with GMM probability density where the

Gaussians are shared by all the hidden states are called semi-continuous density HMM (SCDHMM).

In the case where Gaussians are tied to some hidden state, the emission probability is,

$$f(\mathbf{O}_t|S_k) = \sum_{m=1}^{M_k} \omega_{m,k} \Pr(\mathbf{O}_t; \mathcal{G}_{m,k}) \quad (1.14)$$

where $\{\mathcal{G}_{m,k}, m = 1, \dots, M_k\}$ is the group of Gaussians tied to state S_k , $\omega_{m,k}$ is the weight of Gaussian $\mathcal{G}_{m,k}$ where $\sum_{m=1}^{M_k} \omega_{m,k} = 1$ and $\Pr(\mathbf{O}_t; \mathcal{G}_{m,k})$ is the probability of observation \mathbf{O}_t on Gaussian $\mathcal{G}_{m,k}$ as is defined in Eq.(1.13). This type of HMM is called CDHMM.

1.3 Methods for Vocabulary Acquisition

We classify the related work into different groups by their model configurations.

1.3.1 Recurring Segment Mining and Clustering

Different spoken realizations of the same word are supposed to have similar time-frequency structures in the spectrogram. MFCC vectors computed from the spectrogram are usually taken as the basic representations of speech. Hence by comparing the MFCC sequences of the utterances, recurring word patterns could be discovered and the patterns are modeled by a segment of MFCC vectors. However, speech signals vary in speed. A suitable metric for the comparison should be tolerant to duration variation, e.g. between a fast version and a slow version of the same word. For this purpose, dynamic time warping (DTW) is a proper choice.

DTW is a well-known technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions. Intuitively, the sequences are warped in a nonlinear fashion to match each other [91]. Figure 1.4 shows the distance matrix and the alignment of two utterances. The content of both utterances is digit “one” but from different speakers. Due to time variation in speech, the two versions of “one” have a different number of frames, 86 and 94 respectively in the figure. To apply DTW, a distance matrix of size 86 by 94 is first constructed by calculating the Euclidean distance between the frames of one utterance and the frames of the other utterance, where each frame is represented by its MFCC+ Δ + $\Delta\Delta$. The distance matrix is shown in Figure 1.4

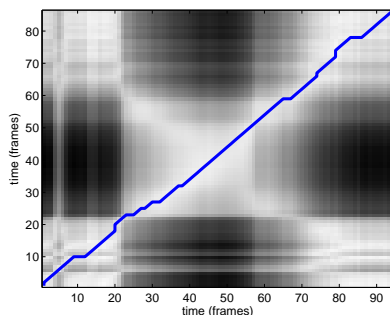


Figure 1.4: The DTW alignment between two utterances each of which only contain an English digit “one”. The matrix of local similarity between the feature vectors of the two utterances is shown (white = good match; black = poor match). The blue line shows the optimal alignment between the two utterances.

where the higher the distance, the darker its color. The thick curve connecting the point $(0,0)$ and the point $(86,94)$ is the aligned path with the lowest cost.

Based on DTW, several methods have been proposed to extract spoken segments, like the DP-ngram in [35] and segmental DTW in [83][129]. The extracted segments are linked to ground truth or clustered towards further interpretation. The methods can segment speech in an unsupervised manner and some of the extracted recurring segments show meaningful relations to the underlying words in the speech corpus. Since DTW favors short segments, a critical parameter in those methods is the minimum duration that a meaningful segment should have. In general, it is difficult to figure out a good parameter for both short and long words. Ad hoc choices are taken in the above publications. Another problem of DTW is its high computational complexity which is not only from the DTW itself but also from the combinatorial nature of the problem: sufficient pair-wise comparison between segments in utterances is necessary to extract recurring meaningful patterns.

1.3.2 Using a Phone Recognizer

A computational model, named cross-channel early lexical learning, was proposed in [89] to learn words from untranscribed acoustic and video input. Cross-channel means that the inputs derived from different sensors (audio and visual in this model) are integrated in an information-theoretic framework.

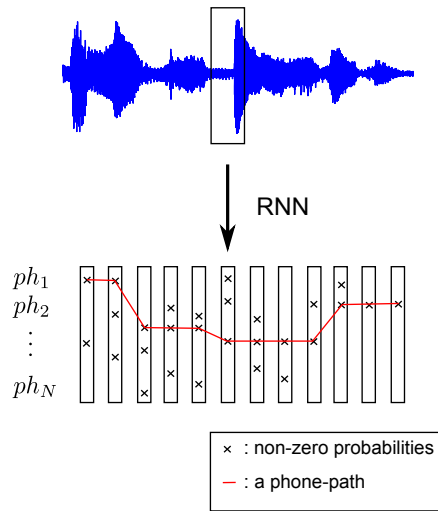


Figure 1.5: The speech representation in [89]. Every frame is represented by posterior probabilities of pre-trained phones. Vocabularies are acquired based on the phone-array-represented utterances.

Acquired words are represented in terms of associations between acoustic and visual sensory experience. The system employs a recurrent neural network to pre-train English phonemes on a database with hand-labeled phone segments. With the phone recognizer, the spoken utterances are represented as phone probability arrays as is shown in Figure 1.5. Meanwhile, visual inputs are also represented into some predefined formats. The mutual entropy between audio and visual representations is utilized to construct audio-visual associations as word meanings.

This method generalizes the research of word meaning learning by using a phone recognizer rather than a word recognizer. So the method assumes that the robot has acquired the phones of a language which seems not true in infant learning of language. Additionally, a method with this assumption cannot get rid of the shortcoming of language dependency, though with the phone set new vocabularies can be generated and acquired.

1.3.3 Latent Variable Models

Given the great success of bag-of-words representations of documents, topic models and probabilistic latent semantic analysis (PLSA) in the research on

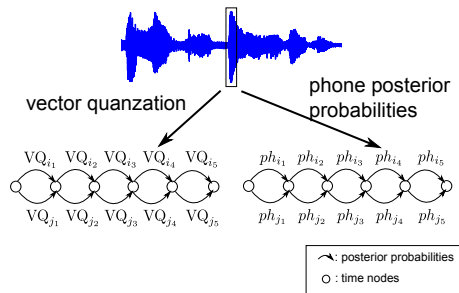


Figure 1.6: The speech representation in [113] and [98]. Directed arcs of acoustic events (VQ labels or phone identities) are utilized to represent the dynamic properties of speech.

information retrieval, a novel representation of utterances and spoken words has been proposed in [113]. An utterance is first described by directed co-occurrences of *acoustic* events over an analysis window of variable length as illustrated in Figure 1.6. The acoustic events are the occurrence of specific patterns, such as vector-quantized (VQ) full-band spectra (VQ_i in Figure 1.6) in [75] and phones (ph_i in Figure 1.6) in [98]. The co-occurrence events of an utterance are subsequently accumulated leading to a vectorial representation of high but fixed dimension called *histogram of acoustic co-occurrence (HAC)*. A HAC is a form of bag-of-features (BoF), which will be adopted as the baseline method of speech representation in this thesis and will be described in detail in Chapter 2. During training, recurring acoustic patterns are discovered and associated with words through non-negative matrix factorization (NMF). During testing, word activations are computed from the HAC-representation and their time of occurrence is estimated. Hence, words in a continuous utterance can be detected, ordered and located.

1.3.4 Unsupervised Training of Hidden Markov Models

Continuous density hidden Markov models (CDHMM) have been the standard model for speech recognition for decades. Gaussian mixture models (GMM) are mostly utilized as the distribution function of the acoustic features in the hidden states. The left-to-right configuration of HMMs describes the dynamic nature of speech. Unsupervised training of CDHMM seems to be a first good choice for vocabulary acquisition.

The first issue to study is the topology of the CDHMM. A dynamic hidden

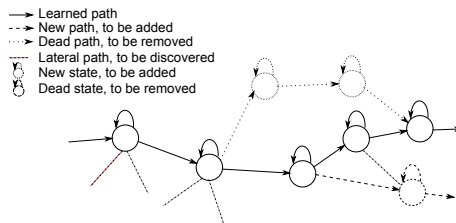


Figure 1.7: Dynamic HMM net in [70]. The hidden states and their transitions can be added and removed during training.

Markov net was developed in [70] for unsupervised online adaptive learning of spoken patterns as is shown in Figure 1.7. The network contains hidden states modeled by multivariate Gaussian functions. During training, new unseen patterns are detected and learned by adding new states and transitions to the network. To avoid explosion of the volume of the network, spurious states and transitions which are rarely visited will be removed gradually. Thus the network structure in Figure 1.7 can be changed gradually. Experiments on a small database of isolated spelled letters showed the model is capable of online learning and recognition. This training scheme succeeds in using a few hidden states to cover the whole training data, but it is unclear how the learned hidden states can be related to spoken words.

In [54], a low-order HMM is studied to model a speech unit which includes three states and allows for left-to-right transitions. HMMs for twenty short speech units were connected to one another to construct an overall HMM to model the data, in which transitions were allowed from the last states of the short HMMs to their first states, as is shown in Figure 1.8. All parameters of this overall HMM were learned using speech data approximately one minute in length without any phoneme transcriptions. After learning the overall HMM, the short HMMs it contains were separated from one another by deleting edges between them, and a set of short HMMs was constructed. Subsequently every spoken word was represented by connecting some of these short HMMs. From the last step, it is obvious that this method requires sufficient examples of *isolated spoken words* to build word HMMs from the learned short HMMs.

A layered framework with emphasis on unsupervised phone learning was presented in [12]. First, a few minutes of input speech are accumulated to give a sufficiently large training sample. Single state HMMs with a mixture of eight Gaussian components as emission probability function are estimated using k -means (top drawing in Figure 1.9). Thereby a transition matrix between the different single-state HMMs is estimated, see the second drawing

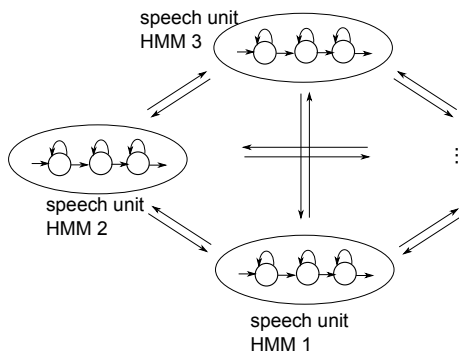


Figure 1.8: HMMs of speech units in [54]. It has similar configurations with an ASR system, but with only three states each and trained unsupervisedly.

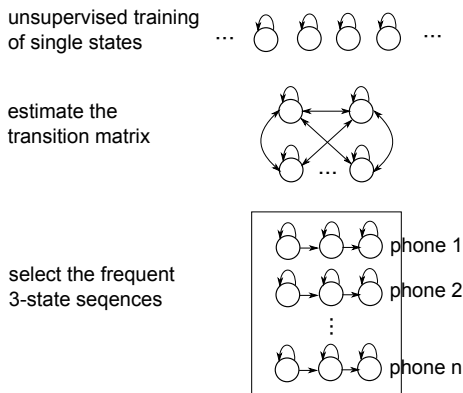


Figure 1.9: Unsupervised learning of phone units in [12]. A group of hidden states are firstly trained without considering their transitions. Then transition probabilities are estimated by bootstrapping the training data. Phone models are finally acquired by finding frequent state paths.

of Figure 1.9. To obtain the initial phone models, a Monte-Carlo-sampling governed by these transition probabilities is used. This gives out the most frequent state-sequences. The n most frequent state sequences are concatenated to 3-state phone-models with Bakis-topology (the third drawing of Figure 1.9). These initial phone models are further refined using Baum-Welch training [5]. The method takes a bottom-up approach and learns phone models in an unsupervised way. The parameters in the method to determine the *most frequent state sequences* and *frequent state paths* are tricky and ad hoc.

As can be seen from the above descriptions, the HMM training is unsupervised in the stage for learning hidden states and for obtaining phone models, where no human-labeled information about the content of the data is used. Based on the obtained hidden states and phone models, training utterances with their corresponding word labels are utilized to construct word models. Other grounding associations can also be imposed to guide the robot for word meanings. According to the task of vocabulary acquisition in section 1.1.1, the amount of labeled utterances or the utterances with grounding information is small, which is different from training a speech recognizer with a large amount of transcribed data as is discussed in section 1.1.3. So the problem of training HMMs for vocabulary acquisition is either *supervised learning with few labeled data* or *unsupervised learning*. It would make the training difficult if there are many unknown parameters to be estimated such as GMMs with a lot of Gaussian components. In other words, for supervised learning with few labeled data, the learning could be over-fitting to the training data and not generalize well to the unseen testing data. For unsupervised training, due to lack of supervisory transcriptions and the non-convexity of the problem, the training would produce poor local optima corresponding to meaningless patterns.

An incremental word learning system is proposed in [3] to cope with few training data samples. The flowchart is illustrated in Figure 1.10. For the training of GMM in HMM, the means and variances of the Gaussians are estimated from data samples where each sample is a frame in the training utterances. A variance estimated from only a few training samples might not be representative for the underlying distribution. The estimated variances are normally observed to decrease in each training iteration [39]. Variance flooring is used to avoid the decrease of the variance in each iteration. To re-estimate the parameters, large margin discriminative training is applied to enhance the generalization ability of the model with few training data. The method provides an incremental framework for word learning and is able to yield good generalization ability even with few training data. However, this approach is currently only valid for training with isolated word inputs, not for continuous speech.

1.4 Goals and Motivations

All the aforementioned models have attractive properties yet shortcomings. An ideal model for vocabulary acquisition should build on elementary speech representations, extract layer-wise spoken patterns and explain the learned patterns by human language knowledge. In this thesis, we are working towards computational models for vocabulary acquisition with the properties: learning from continuous speech, data-driven and yielding a layered model.

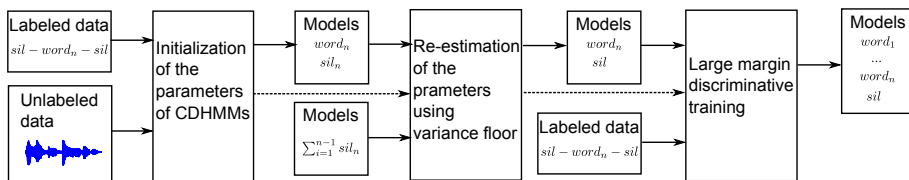


Figure 1.10: An incremental word learning model proposed in [3]. Both labeled and unlabeled data are used for initializing a CDHMM. The parameters will be re-estimated by bootstrapping the training data and a variance floor is estimated and implemented to avoid underestimation of the variances. Finally large margin discriminative training is utilized to improve the recognition performance.

1.4.1 Goal 1: Learning from Continuous Speech

It would be exhausting for a tutor to teach a robot vocabularies by repeating each specific word up to hundreds of times in isolation. A more natural way would be that a robot always listens to *continuous* speech, analyzes the speech autonomously and acquires the new patterns and updates the previously stored ones.

One may argue that, in engineering, speech segments containing words can be generated prior to vocabulary acquisition. Actually, speech segmentation and vocabulary acquisition are coupled and should be learned together. Vocabulary acquisition cannot rely on speech segments because we do not know what the words are until we have learned them.

1.4.2 Goal 2: Data-driven Models

Data-driven modeling is based on the analysis of the data characterizing the system under study. In contrast to the “knowledge-driven” models, a data-driven model can be defined on the basis of connections between the system state variables (input, internal and output variables) with only a limited number of assumptions about the “physical” behavior of the system. Hence, the prospective data-driven models for vocabulary acquisition would not make many assumptions about the speech, like the language being spoken, the grammar being used, and the lexical boundaries. This property corresponds to the tutoring effort made by care-givers to a child or robot. Humans acquire spoken vocabularies without being told the segmentation boundaries between words, the phone units a word contains nor the speakers who uttered the words.

However, it does not mean that no extra information can be applied to the data-driven models. The common properties for all languages, like layer-wise structures with different granularity, can still be applied, which will be explained in detail in the next section. Besides this, two more assumptions would also be useful for the training of data-driven models.

Goal 2.1: Repeated Patterns

In daily conversations, spoken units (phones, syllables and words) are repeated. There is evidence to show that an 8-month young child has already the ability to extract new words by identifying repeated words from the current speech [90][57]. Intuitively, to gain the attention of the robot, a pattern would either be repeated many times or occur together with other salient inputs. So in the unsupervised training solely with the speech input, repetition is a basic requirement for language learning.

Goal 2.2: Topological Structures of Speech Representation

Speech shows stationary properties on a short time scale. Speech perception is a multi-time resolution process, with perceptual analysis occurring concurrently on at least two time scales (approx. 20-80 ms, approx. 150-300 ms), commensurate with (sub)segmental and syllabic analysis, respectively [86]. Furthermore, the transient parts in speech - consonants - live at even shorter time scales. Speech thus contains information living at multiple time scales. A variate or multiple analysis window, rather than a window with fixed length for all parts in speech, seems to be suitable to reflect the various time-frequency structures of speech.

Speech is a kind of time-series, so the models of speech units should have *left-to-right structures* along the time axis, which is also demonstrated by the state-of-the-art results achieved by fitting left-to-right HMMs to training data for ASR.

1.4.3 Goal 3: Layered Architectures with Reusable Units

In language hierarchy, spoken words are built upon a small number of phones by their various combinations. Some variation of the phones is insignificant, so they are allophones of a phoneme¹. Other variation is significant and makes

¹For example, [p^h] as in pin and [p] as in spin are allophones for the phoneme /p/ in the English language because they cannot distinguish words (in fact, they occur in complementary

a new phoneme. We can only learn this from examples of words and observe that some variations produce a new word. Once the phonemes are learned, we can recognize them and use them to distinguish different other words.

In computational methods for vocabulary acquisition, the above process can be simulated as follows: we first learn some spoken words from speech, then decompose these words into sub-word units with finer granularity. During the learning process, no concrete form of the sub-word units would be supposed, like context-dependent and context-independent phone models shared by different words in ASR. The sub-word units depend on what the model discovers from the data while respecting the underlying structures in the data.

The sub-word units discovered from words can be quite useful to the learning of new words. For example, the agent has learned four English digits: “four”, “five”, “six” and “seven”. Now we have a new word “sive” to be learned. If no sub-word units are available, the agent has to learn “sive” from its acoustic-level representations. However, if the agent has abstracted the phone-like unit “s” from the examples of “six”, “seven” and many other words containing “s” and if it has also extracted the phone-like unit “i”-“ve” from “five” by contrasting “five”, “four” and many other words containing “f”, the learning of the new word can be based on the sub-word phone-like units which would certainly induce a faster learning rate. However, unlike in supervised phone learning with human-defined phone labels, sub-word units discovered unsupervisedly may not correspond to the phone labels exactly, while other explanations would also be possible. The sub-word units are useful as long as they can represent spoken words accurately.

1.5 Outline of the Thesis

Enlightened by the extensively studied techniques in ASR, we design computational models to discover and represent vocabularies from continuous speech. Starting with a recently proposed non-negative matrix factorization (NMF) approach to vocabulary acquisition, this work targets at vocabulary representations with high accuracy and fast learning rate. The dissertation is organized in five chapters.

- **Chapter 2** introduces the NMF framework for vocabulary acquisition and evaluation. A spoken utterance is represented by its bag-of-features (BoF), and the BoFs of the training utterances form the matrix to be

distribution). English speakers treat them as the same sound, but they are different: the first is aspirated and the second is unaspirated (plain) [60].

factorized. The cost function of NMF is Kullback-Leibler divergence (KLD). Since NMF is good at finding repeated parts from data, it is expected to discover recurring words as parts from continuous utterances, which fits the **Goal 1** and **Goal 2.1**. The conceptual comparison between NMF and other dimension reduction or clustering techniques is discussed, which explains why NMF is chosen in this thesis.

- **Chapter 3** improves the BoF representations of speech by overcoming the accuracy loss when making the BoF representation using vector quantization (VQ). Three techniques are experimented with multi-codebook, soft VQ and multiple time scales, where multi-codebook and soft VQ are basically clustering techniques and multiple time scales originate from the **Goal 2.2**. All of them are demonstrated to be useful for the improvement of vocabulary acquisition by the NMF model. The obtained accuracies approach those obtained with hidden Markov models (HMM) trained with transcribed data. However, the above improvements of the NMF model are at the expense of high computational complexity and require sufficient labeled data as supervision. These concerns are addressed in Chapter 4 and Chapter 5 as follows.
- **Chapter 4** imposes graph regularization on the original NMF model. The BoF representation used in the NMF learning framework presented in Chapter 2 has only a weak representation of temporal adjacency. In complex learning situations (long sequences, poor grounding information, ...), this might lead to erroneous vocabulary models that do not make temporal sense, e.g. that do not correspond to contiguous sounds. The graph model that will be introduced in this chapter reflects the temporal closeness of features when representing speech. The graph regularized NMF can preserve feature closeness in the solutions, hence can produce meaningful spoken patterns in unsupervised learning. Those properties correspond to **Goal 2**. With the regularization term, the optimization problem becomes ill-posed if not being normalized. Also new algorithms should be created to solve the new problem efficiently. Thus an element-wise updating algorithm is proposed to solve this large-scale (e.g. for a matrix with 1000,000 rows and 10,000 columns) graph regularized NMF with ℓ_1 normalization (L1GNMF). The proposed algorithm obtains superior performance with respect to the original NMF and a similar graph regularized NMF algorithm proposed in [15]. Beside its good performance on speech data, the L1GNMF also works on image pattern discovery by imposing spatial closeness between visual features.
- **Chapter 5** applies non-negative matrix tri-factorization (NMTF) to discover hidden sub-word units from the NMF-learned vocabulary models. The process is similar to the bootstrapping process of infants' learning

language. It is shown that the discovered sub-word units can be interpreted by multiple consecutive HMM states if using Gaussians of GMMs from supervised HMM training. The idea is subsequently extended to totally unsupervised cases where a sequential decoding scheme is applied to emphasize the sequential aspect of speech. Fast word learning rate, that is to obtain accurate vocabulary models by using as few labeled data as possible, is observed by employing the discovered sub-word units. This chapter addresses **Goal 3**.

- **Chapter 6** The sub-word units discovered in the NMTF learning of Chapter 5 have strong relations with hidden states of HMMs, so NMTF can be applied to training an HMM. A joint training paradigm of NMF, NMTF and HMM is proposed for the unsupervised training of HMMs. The basic idea is to create two views of an utterance: one is the observation sequence and the other is the bag-of-features representation. NMF provides the global view of the data by decomposing data into parts, while the HMM offers strict frame-by-frame modeling. The two views benefit from each other in the joint training framework. The methods are evaluated on the task of vocabulary discovery, segmentation and recognition on the TIDIGITS database and demonstrate good performance on discrete density HMM, continuous density HMM and a Kullback-Leiber divergence based HMM.
- **Chapter 7** concludes the work and discusses future directions for vocabulary acquisition.

Chapter 2

An NMF Approach for Vocabulary Acquisition

In this chapter, we apply non-negative matrix factorization (NMF) to vocabulary acquisition. The general formulation of NMF is first presented. Subsequently, the representation of spoken utterances using a vector-space model is given to model speech data with matrices. Finally the training and testing framework of NMF is described.

2.1 A General Description of Non-negative Matrix Factorization

2.1.1 NMF: Metrics and Algorithms

The NMF problem can be formulated as,

$$\mathbf{V}_{M \times N} \approx \mathbf{W}_{M \times R} \mathbf{H}_{R \times N} \quad (2.1)$$

where M, R and N are the dimensions of the matrices. Constructed from the training data, N multivariate M -dimensional data vectors are placed in the columns of an $M \times N$ matrix \mathbf{V} . This matrix is then approximately factorized into an $M \times R$ matrix \mathbf{W} and an $R \times N$ matrix \mathbf{H} . In the above process, the model *extracts R factors* (columns of \mathbf{W}) from N observed samples (columns

of \mathbf{V}). R is chosen to be smaller than N to obtain the low-rank approximation of the original data matrix. All the elements involved are non-negative. [65]

Objective functions

To evaluate the approximation between \mathbf{V} and its reconstruction \mathbf{WH} in Eq.(2.1), the Frobenius norm in Eq.(2.2) or the Kullback-Leibler divergence (KLD) in Eq.(2.3) are usually employed.

$$\|\mathbf{V} - \mathbf{WH}\|_F^2 = \sum_{m,n} (V_{m,n} - (WH)_{m,n})^2 \quad (2.2)$$

$$\text{KLD}(\mathbf{V} \parallel \mathbf{W}, \mathbf{H}) = \sum_{m,n} V_{m,n} \log \frac{V_{m,n}}{(WH)_{m,n}} - V_{m,n} + (WH)_{m,n} \quad (2.3)$$

The Frobenius norm originates from the Gaussian noise assumption, and it is suitable to model data like power spectra and gray scale images. Let $\mathcal{V}_{m,n}$ be the random variable of the scale of feature m in sample n and assume it follows $\mathcal{N}((WH)_{m,n}, \sigma)$. Then the probability density function is,

$$\Pr(\mathcal{V}_{m,n} = V_{m,n}; (WH)_{m,n}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(V_{m,n} - (WH)_{m,n})^2} \quad (2.4)$$

Given a set of observations $V_{m,n}$, the maximum likelihood estimation (MLE) of $(WH)_{m,n}$ is equivalent to maximizing the objective function,

$$\sum_{m,n} \log \Pr(\mathcal{V}_{mn} = V_{m,n}; (WH)_{m,n}, \sigma) = \sum_{m,n} -\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}(V_{m,n} - (WH)_{m,n})^2 \quad (2.5)$$

Without considering the additive constant $-\log(\sqrt{2\pi}\sigma)$ and the scaling factor $\frac{1}{2\sigma^2}$, the maximization of Eq.(2.5) is equivalent to the minimization of the Frobenius norm in Eq.(2.2).

KLD is based on the Poisson noise assumption and fits well to count data. Let $\mathcal{V}_{m,n}$ be the random variable of the count of feature m in sample n and assume

it follows $\text{Poisson}((WH)_{m,n})$. Then the probability density function is,

$$\Pr(\mathcal{V}_{m,n} = V_{m,n}; (WH)_{m,n}) = \frac{((WH)_{m,n})^{V_{m,n}}}{V_{m,n}!} e^{-(WH)_{m,n}} \quad (2.6)$$

Given a set of observations $V_{m,n}$, the MLE of $(WH)_{m,n}$ is equivalent to maximizing the objective function,

$$\sum_{m,n} \log \Pr(\mathcal{V}_{m,n} = V_{m,n}; (WH)_{m,n}) = \sum_{m,n} V_{m,n} \log (WH)_{m,n} - (WH)_{m,n} \quad (2.7)$$

Except for a constant term $\sum_{m,n} V_{m,n} \log V_{m,n} - V_{m,n}$, the maximization of the above objective function is equivalent to the minimization of the KLD in Eq.(2.3).

Algorithms

The difficulty to solve the NMF problem lies in preserving the non-negativity of the elements of \mathbf{W} and \mathbf{H} while optimizing the objective functions. So the conventional gradient descent method has to be modified for this purpose. The projective gradient method is a possible choice [67]. In our work, for the NMF optimization, we use the multiplicative update algorithm proposed by Lee and Seung in [65] where one can find the details of the derivation of the algorithm and the proof of convergence. We list the related algorithms for NMF based on Frobenius norm in Eq.(2.8) and KLD based NMF in Eq.(2.9). In both versions, \mathbf{W} and \mathbf{H} are updated iteratively.

$$H_{r,n} \leftarrow H_{r,n} \frac{(W^T V)_{r,n}}{(W^T W H)_{r,n}}, \quad W_{m,r} \leftarrow W_{m,r} \frac{(V H^T)_{m,r}}{(W H H^T)_{m,r}} \quad (2.8)$$

$$H_{r,n} \leftarrow H_{r,n} \frac{\sum_i W_{i,r} V_{i,n} / (WH)_{i,n}}{\sum_i W_{i,r}}, \quad W_{m,r} \leftarrow W_{m,r} \frac{\sum_j H_{r,j} V_{m,j} / (WH)_{m,j}}{\sum_j H_{r,j}} \quad (2.9)$$

Properties of the algorithms:

- **Element-wise updating** The operations are matrix-vector multiplications and element-wise arithmetics, so the algorithms are quite efficient

for parallel computations. With this property, the sparsity in the data matrix \mathbf{V} will reduce the computation complexity by using sparse operations from $O(MNR)$ to $O(pNR)$ where p is the maximal number of activated features in the columns of \mathbf{V} .

- **Multiplicative updating** The updated variable is generated by the product of its previous value and another term. With this multiplicative updating, the zero locations in the original \mathbf{W} or \mathbf{H} will remain zero. The property is called *zero-locking*. It is useful to preserve the structure appointed in the initialization stage and is also useful to obtain sparse solutions.
- **Scaling ambiguity** Since the objective functions compare \mathbf{V} and the product \mathbf{WH} , any scaling of \mathbf{W} by a non-negative matrix \mathbf{S} with non-negative inverse, \mathbf{WS} , and re-scaling of \mathbf{H} with $\mathbf{S}^{-1}\mathbf{H}$, will not change the objective function value. When \mathbf{S} is a diagonal matrix, this phenomenon can be called *scaling ambiguity*. It can be solved with column-wise normalization of \mathbf{W} as suggested in [64].

2.1.2 Relations to Other Methods

***k*-means clustering and vector quantization**

In the view of clustering, the columns of \mathbf{W} of NMF can be considered as clustering centers [23]. Assuming $\mathbf{V} = [\mathbf{V}_1, \dots, \mathbf{V}_N]$, $\mathbf{W} = [\mathbf{W}_1, \dots, \mathbf{W}_R]$ and $\mathbf{H} = [\mathbf{H}_1, \dots, \mathbf{H}_N]$, the NMF with Frobenius norm is to minimize,

$$\|\mathbf{V} - \mathbf{WH}\|_F^2 = \sum_n \|\mathbf{V}_n - \sum_r \mathbf{W}_r H_{rn}\|_2^2 \quad (2.10)$$

In clustering and vector quantization (VQ), each data sample should be assigned to one cluster, so H_{rn} is binary and holds $\sum_r H_{rn} = 1$. Thus the above equation becomes,

$$\sum_n \sum_r H_{rn} \|\mathbf{V}_n - \mathbf{W}_r\|_2^2 \quad (2.11)$$

which is the within-cluster distance to be optimized in *k*-means clustering. Once the cluster centers are obtained, VQ can be applied to the data samples to compress the data. The difference between NMF and VQ is that NMF can learn parts from the data while the *k*-means and VQ can only yield *averages* of the data samples, as illustrated in [64].

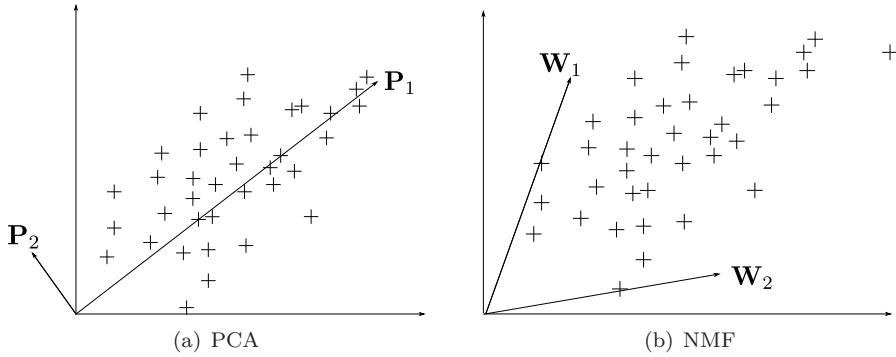


Figure 2.1: Geometric illustration of PCA and NMF. PCA finds an orthogonal basis where the first principle vector \mathbf{P}_1 lies in the main direction of the data points. NMF tries to find solutions such that the majority of data points can be represented as their convex combinations.

Principle component analysis

Principle component analysis (PCA) is another technique for dimension reduction by matrix factorization. It is usually accomplished by singular value decomposition (SVD) as is shown in Eq.(2.12).

$$\mathbf{V}_{M \times N} \approx \mathbf{P}_{M \times R} \mathbf{S}_{R \times R} \mathbf{B}_{N \times R}^T \tag{2.12}$$

Being different from NMF, the entries in \mathbf{P} and \mathbf{B} do not have to be non-negative. The columns of \mathbf{P} are the eigenvectors of $\mathbf{V}\mathbf{V}^T$, so they are orthogonal to each other. The columns of \mathbf{B} are the eigenvectors of $\mathbf{V}^T\mathbf{V}$. \mathbf{S} is a diagonal matrix whose diagonal elements are the square roots of the eigenvalues of $\mathbf{V}\mathbf{V}^T$ or $\mathbf{V}^T\mathbf{V}$. By only taking the eigenvectors corresponding to the top R eigenvalues, the data matrix \mathbf{V} is expressed as the product of the “thin” matrices as is expressed in Eq.(2.12) where $R \leq \min\{M, N\}$.

The geometric explanation of NMF and PCA is illustrated in Figure 2.1. The pluses denote the data samples. PCA finds orthogonal *main* directions (the columns of \mathbf{P} : \mathbf{P}_1 and \mathbf{P}_2 in the figure) of the data, while NMF constructs a convex cone to hold the data where the convex cone is configured by the non-negative bases (\mathbf{W}_1 and \mathbf{W}_2 in the figure).

Probabilistic latent semantic analysis

The NMF model with Kullback-Leibler divergence is equivalent to probabilistic semantic analysis (PLSA) [37][25]. In text-based document analysis, PLSA is a method to find latent topics \mathbf{z} from the mutual relations between two other statistical variables, text words \mathbf{f} and documents \mathbf{d} . PLSI has been further extended to a more general latent Dirichlet allocation (LDA) model [8]. By modeling the relations between \mathbf{f} and \mathbf{d} with the conditional probabilities, the mathematical expression is,

$$\Pr(f_m|d_n) \approx \sum_r \Pr(f_m|z_r)\Pr(z_r|d_n) \tag{2.13}$$

The corresponding graphical model is given in Figure 2.2 where an arrow from variable a to b means that b is generated from a with probability $\Pr(b|a)$.

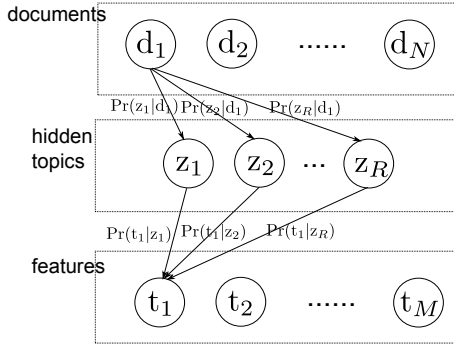


Figure 2.2: The directed graph for probabilistic latent semantic analysis. The terms (t_i) are generated by hidden topics (z_k) which are further generated from documents (d_j) . Here, the “generate” means sampling by following some multinomial distribution.

With the notation $V_{m,n} := \Pr(f_m|d_n)$, $W_{m,r} := \Pr(f_m|z_r)$ and $H_{r,n} := \Pr(z_r|d_n)$, we explain the equivalence between the solution of PLSA and that of NMF. The maximum likelihood solution of PLSA is derived using an EM algorithm in Eq.(2.14) [48].

$$\begin{aligned} \Pr(f_m|z_r) &\leftarrow \frac{\sum_n V_{m,n} \Pr(z_r|d_n)}{\sum_m \sum_n V_{m,n} \Pr(z_r|d_n)} \\ \Pr(z_r|d_n) &\leftarrow \frac{\sum_m V_{m,n} \Pr(z_r|d_n)}{\sum_r \sum_m V_{m,n} \Pr(z_r|d_n)} \end{aligned} \tag{2.14}$$

where $\Pr(z_r|f_m, d_n) = \frac{\Pr(f_m|z_r)\Pr(z_r|d_n)\Pr(d_n)}{\sum_r \Pr(f_m|z_r)\Pr(z_r|d_n)\Pr(d_n)}$. It is easy to see that $\Pr(z_r|f_m, d_n) = \frac{W_{m,r}H_{r,n}}{\sum_r W_{m,r}H_{r,n}}$. So Eq.(2.9) becomes,

$$\begin{aligned} W_{m,r} &\leftarrow \sum_n V_{m,n} \frac{W_{m,r}H_{r,n}}{\sum_r W_{m,r}H_{r,n}}, & W_{m,r} &\leftarrow \frac{W_{m,r}}{\sum_m W_{m,r}} \\ H_{r,n} &\leftarrow \sum_m V_{m,n} \frac{W_{m,r}H_{r,n}}{\sum_r W_{m,r}H_{r,n}}, & H_{r,n} &\leftarrow \frac{H_{r,n}}{\sum_m H_{m,n}} \end{aligned} \quad (2.15)$$

With the column-wise normalization of \mathbf{V} and \mathbf{W} , Eq.(2.9) is equivalent to Eq.(2.14).

Dictionary learning

In NMF, each data vector \mathbf{V}_n is modeled by a linear combination of the columns of \mathbf{W} , i.e. $\mathbf{V}_n \approx \mathbf{W}\mathbf{H}_n$ where n is the index of the data vector. The columns of \mathbf{W} are usually called *bases* or *atoms*, and the set of the columns of \mathbf{W} is a *dictionary*. However, NMF constrains the bases in \mathbf{W} and the activations in \mathbf{H}_n to be non-negative where no subtraction is allowed. Besides this, the number of bases, R , should be smaller than the size M and N of the original data matrix \mathbf{V} . But the dictionary learning does not limit the number of bases, e.g. an over-complete dictionary can have more bases than the number of data samples, that is $R \geq N$, as is illustrated in Figure 2.1.2.

In summary, given the comparison of NMF with respect to other methods, KLD based NMF has the advantage of *parts-of-whole* learning, non-negative representations and a solid probabilistic interpretation. It has therefore been selected as our method for vocabulary acquisition. Dictionary learning also provides promising properties which can help to improve the original NMF model. For example, incremental learning of a large dictionary together with non-negative representations would benefit our task on large vocabulary cases. Suppose the robot has acquired some words in $\mathbf{W}^{(\text{old})}$, it can learn new words from observations \mathbf{V} by estimating $\mathbf{W}^{(\text{new})}$ and fixing $\mathbf{W}^{(\text{old})}$, i.e. by minimizing Eq.(2.16) w.r.t. $\mathbf{W}^{(\text{new})}$, $\mathbf{H}^{(\text{new})}$ and $\mathbf{H}^{(\text{old})}$.

$$\text{KLD} \left(\mathbf{V} \parallel [\mathbf{W}^{(\text{old})} \ \mathbf{W}^{(\text{new})}], \begin{bmatrix} \mathbf{H}^{(\text{old})} \\ \mathbf{H}^{(\text{new})} \end{bmatrix} \right) \quad (2.16)$$

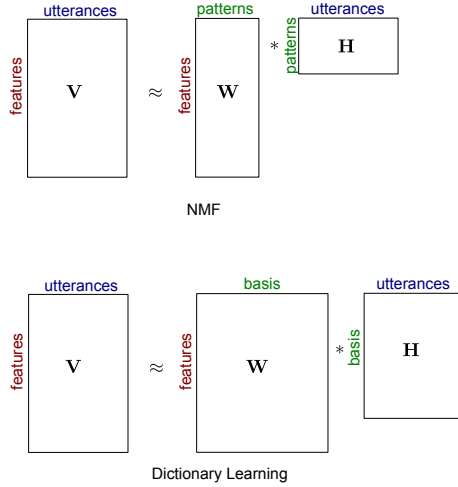


Figure 2.3: NMF and dictionary learning. A fundamental condition for NMF is the “low-rank” approximation, but dictionary learning does not have this constraint and could yield more bases than data samples.

2.2 Bag-of-Features Representation of Speech

The basic representation of data samples in NMF is a vector. Without segmentation of continuous speech, one utterance would be taken as a data sample. An utterance will first be represented by its *bag-of-features* vector as is illustrated in Figure 2.4. This vector reflects the occurrence frequency in the utterance of each of the features. In the simplest case, the features can be taken as clustering centers of MFCCs [75].

2.2.1 Mapping a Sequence to a Vector

As is explained in section 1.2, an input utterance is represented by a series of MFCC vectors, say $\{\mathbf{O}_t, t = 1, \dots, T\}$, where \mathbf{O}_t is the observation vector of frame t . Each frame vector is subsequently labeled by codewords. In speech coding, codewords are obtained as cluster centers during a process where similar training data points are grouped into clusters. The set of codewords is called a *codebook*. Here we assume we have already obtained the codebook with codewords, $\{\mathcal{C}_i, i = 1, \dots, I\}$ where I is the number of codewords or the codebook size. The details of training codebooks for speech representation will

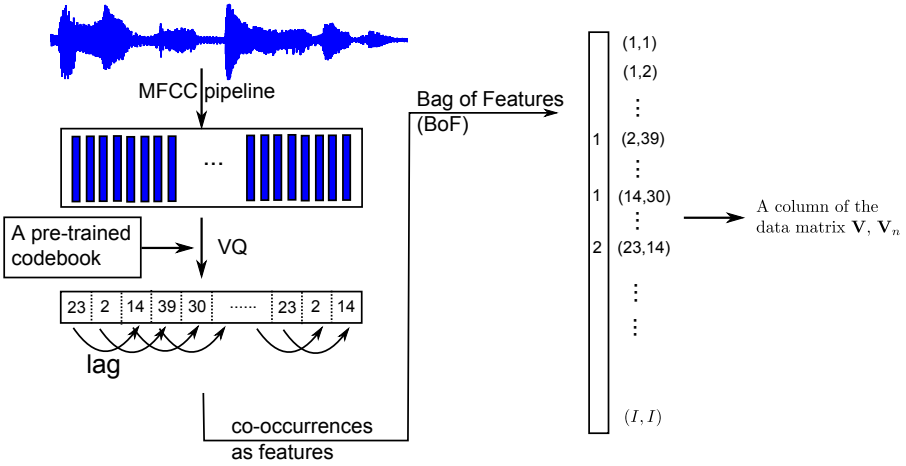


Figure 2.4: The flowchart of transforming an utterance into its BoF presentation. MFCC vectors are first extracted and quantized into numbers by using a pre-trained codebook. With a parameter lag , co-occurrences are defined as features. Subsequently, the bag-of-features representation is computed for the utterances and is stored in a column of the data matrix \mathbf{V} .

be discussed in section 3.1.1. The simplest way to label \mathbf{O}_t by the codebook, is *winner-takes-all* by only using the closest codeword in Euclidean distance.

$$\operatorname{argmin}_i \|\mathbf{O}_t - \mathbf{C}_i\|_2 \tag{2.17}$$

In this stage, the input is an utterance with $D \times T$ MFCC vectors, and the output is a $1 \times T$ sequence of codeword numbers, where D is the dimension of MFCC and T is the number of frames in the utterance. We can reformulate the $1 \times T$ sequence to a $I \times T$ vector sequence by putting ones only at the labeled code words as is shown in Eq.(2.18).

$$\Pr(x_t = \mathbf{C}_i) = \begin{cases} 1, & i = \operatorname{argmin}_{i'} \|\mathbf{O}_t - \mathbf{C}_{i'}\|_2 \\ 0, & \text{elsewhere} \end{cases} \tag{2.18}$$

By using codewords as features, the *bag-of-features* (BoF) representation is

$$V_{in}^{(S)} = \sum_{t=1}^{T_n} \Pr(x_t = \mathbf{C}_i) \tag{2.19}$$

where S means the *static* stream MFCC and T_n is the length in number of frames of utterance n .

The name “bag-of-features” describes that $V_{in}^{(S)}$ characterizes the input speech as an occurrence frequency of a set of features, $\{x_t\}$. The order in which the features occur is not represented. It is as if we observe all occurring features and throw them in a bag, after which the order in which they occurred cannot be reconstructed. This nomenclature is inspired from the “bag-of-words” representation in document retrieval [48].

Though the dynamic property of speech could be modeled by the Δ and $\Delta\Delta$ features in each frame, the long-term trajectory will be lost when summing over the frames to compute the BoF representation. Hence, the co-occurrences of codewords can be utilized as features when computing the BoF to reflect the dynamic characteristic of speech. A parameter τ is taken as a temporal lag between two frames to count the co-occurrences of codewords. This kind of BoF representation is named *histogram of acoustic co-occurrences* (HAC) in [113].

$$V_{i+i'*I,n}^{(S,\tau)} = \sum_{t=1}^{T_n} \Pr(x_t = \mathcal{C}_i, x_{t+\tau} = \mathcal{C}_{i'}) \quad (2.20)$$

where $\Pr(x_t = \mathcal{C}_i, x_{t+\tau} = \mathcal{C}_{i'})$ can be approximated by $\Pr(x_t = \mathcal{C}_i) * \Pr(x_{t+\tau} = \mathcal{C}_{i'})$.

Similar operations can be applied to the Δ stream and $\Delta\Delta$ stream with their own codebooks. For each stream, the HAC representation of an utterance is a $I^2 \times 1$ vector where I is the number of code words in the codebook of that stream. The lag parameter τ can also have multiple values to represent contextual information at different temporal scales. We use $\mathbf{V}^{(S,\tau_1)}$ to denote the data matrix generated with lag τ_1 from the MFCC (static) vectors. By stacking the data matrices with different streams (static (S), velocity (V) and acceleration (A)) and multiple temporal lags (τ_1, τ_2), we obtain a comprehensive data matrix in Eq.(2.21) where each column represents an utterance in different aspects.

$$\begin{bmatrix} \mathbf{V}^{(S,\tau_1)} \\ \mathbf{V}^{(V,\tau_1)} \\ \mathbf{V}^{(A,\tau_1)} \\ \mathbf{V}^{(S,\tau_2)} \\ \mathbf{V}^{(V,\tau_2)} \\ \mathbf{V}^{(A,\tau_2)} \end{bmatrix} \quad (2.21)$$

2.2.2 Linear Operations

The BoF representation is basically a transformation to convert a sequence of variable length to a vector with fixed dimension. Given an utterance containing T words, the following additive property of the BoF operation holds.

$$\text{BoF}(\text{word}_{u_1}, \text{word}_{u_2}, \dots, \text{word}_{u_T}) \approx \sum_{l=1}^L \omega_l \text{BoF}(\text{word}_l) \quad (2.22)$$

where u_t ($1 \leq u_t \leq L$) is the word index, L is the total number of words and ω_l is the occurrence frequency of word l in the utterance. With this property, it is clear that the whole data (BoF of an utterance) are composed of parts where each part refers to a BoF of word. This property offers the foundation for NMF to be applicable to the task of vocabulary acquisition.

However, the above condition requires the BoF representation of a word to be *invariant* to possible pronunciation variations. The invariance here means for the same word pronounced in two different circumstances, e.g. pronunciation durations, spoken by different speakers or with different contextual words, should lead to similar BoF representations. To achieve the invariance property, techniques should be considered to improve the BoF representation, which is one of the motivations for the work in the following chapters.

2.3 NMF for Vocabulary Acquisition

The two motivations to use NMF for vocabulary acquisition are first that NMF is good at unsupervised/semi-supervised learning and second that NMF can learn parts from data without segmentation. The two properties would be suitable to extract vocabulary patterns from continuous speech with no or little human effort.

2.3.1 Training

Above we described how to make the data matrix using BoF representations of utterances. In principle, the BoF word representations could be found in unsupervised NMF learning of the data matrix \mathbf{V} . But it is difficult to explain the patterns without side information. In language acquisition, the meaning of the (word-like) patterns can be derived from other modalities together with speech. For example if a baby often hears the pattern “daddy” upon seeing his father, a semantic link is established between them [30][89]. The information

from other modalities is referred to as *grounding* information. In our task, to focus the research on speech, we will assume that all information from other modalities than speech is modeled perfectly.

Grounding

In this work, the grounding information is also represented as a bag-of-features. In child language acquisition, this corresponds to the situation that the child can recognize independent objects such as “daddy”, “bear” or “ball”. Each object, described by a set of key words, appearing in the visual scene triggers a feature in the BoF representation to be present. Since there is no sequential order in the visual scene, a bag-of-features description is appropriate. The grounding information is thus represented as follows: for the training set, if the n -th utterance is known to contain K_n key words from a set of L with indices m_1, m_2, \dots, m_{K_n} ($1 \leq m_k \leq L$), we can construct the $L \times N$ grounding matrix \mathbf{G} with accumulated ones in its m_k -th row and n -th column and zero elsewhere, where $k = 1, 2, \dots, K$ and N is the number of utterances in the training set. In other words, $G_{l,n}$ is the number of occurrences of the l -th keyword in the n -th utterance.

With the grounding matrix, the training process is as follows where the cost function is the Kullback-Leibler divergence. To balance the impact of the grounding part \mathbf{G} and the data matrix \mathbf{V} , a scaling factor is imposed on the data matrix: $V_{i,j} \leftarrow (\sum_{l,n} G_{l,n} / \sum_{i,n} V_{i,n}) V_{i,n}$ prior to the factorization.

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{V} \end{bmatrix} \approx \begin{bmatrix} \mathbf{W}^{(g)} \\ \mathbf{W} \end{bmatrix} \mathbf{H} \quad (2.23)$$

We call the columns of \mathbf{W} , *patterns*. The learned patterns are in their BoF representation. The matrix $\mathbf{W}^{(g)}$ links the patterns to the grounding words. The matrix \mathbf{H} contains the activations of the patterns in the utterances of the training set. The number of patterns R should be equal to or larger than the number of grounding words, i.e. $R \geq L$. That is to use some extra columns in $\mathbf{W}^{(g)}$ and \mathbf{W} to accommodate the information unrelated to grounding words in the data, e.g. the carrier sentence, silence and word boundaries. We will call these extra patterns, *garbage* patterns.

Initialization

To obtain our expected vocabulary patterns, $\mathbf{W}^{(g)}$ or \mathbf{H} is cleverly initialized. $\mathbf{W}^{(g)}$ can be initialized by stacking an identity matrix with size $L \times L$ and a

non-negative random matrix with size $L \times (R - L)$. \mathbf{H} can be initialized by stacking the grounding matrix \mathbf{G} with size $L \times N$ and a non-negative random matrix with size $(R - L) \times N$. The zero-locking properties of NMF updating will direct the solution to the desired activation pattern.

2.3.2 Evaluation Methods

In the stage of recognition, we first compute the activation probability matrix \mathbf{H}' of the learned patterns,

$$\min_{\mathbf{H}'} \text{KLD}(\mathbf{V}' || \mathbf{W}\mathbf{H}') \quad (2.24)$$

where \mathbf{V}' is the BoF matrix of the utterances of the test set, i.e. \mathbf{H}' is estimated from the acoustic BoF representation, without using the grounding information. Only \mathbf{H}' needs to be estimated while \mathbf{W} is obtained from the trained model. The estimated activation matrix $\hat{\mathbf{G}}'$ of the grounding words is subsequently computed for the testing utterances:

$$\hat{\mathbf{G}}' = \mathbf{W}^{(g)}\mathbf{H}' \quad (2.25)$$

Two evaluation metrics are available to compare the estimated $\hat{\mathbf{G}}'$ and the true grounding matrix \mathbf{G}' of the test set.

Equal error rate (EER)

By thresholding these word activations in $\hat{\mathbf{G}}'$, we can detect words in the utterances. The threshold value will trade off false alarm rate for missed detection rate and is chosen independent of the keyword. In this evaluation, we always choose the operating point where both error rates have the same occurrence frequency, i.e. we report the *equal error rate* of word detection.

The metric is questionable when the number of true positives and the number of true negatives are seriously unbalanced, e.g. in a database with $L = 50$ keywords and every utterance only contains 2 keywords (2 positives versus 48 negatives). Taking the equal error rate will yield around 20 times more false alarms than missed detections. However, it can still reflect the relative performance of the models and algorithms. Compared to the following unordered word error rates, no significant difference was observed from equal error rates when drawing conclusions for the model evaluations in the previous experiments [27].²

²In this thesis, only the results in section 3.1.2 use this metric.

Unordered word error rate (UWER)

Since $\hat{\mathbf{G}}'$ indicates the presence of the words in each test utterance without ordering them in time, a performance metric that can be adopted here is an *unordered* error rate. Suppose that the number D_u of different keywords occurring in the u -th test utterance is given, the D_u candidates with highest activation are retained in the u -th column of $\hat{\mathbf{G}}'$. The word error rate is then defined as the sum of the number of incorrect words (only substitution), divided by the sum of D_u over the complete test set [76]. By transforming $\hat{\mathbf{G}}'$ and \mathbf{G}' into binary matrices, the computation of the unordered word error rate is,

$$\text{UWER} = \frac{\sum_{kn} |\hat{G}'_{kn} - G'_{kn}|}{2 \sum_{kn} G'_{kn}} \quad (2.26)$$

2.3.3 Advantages and Disadvantages of the NMF Model

As is mentioned above, the NMF learning model is suitable for discovering word-like parts from continuous speech without additional segmentation. The KLD fits well to the count data in the BoF representation. Promising results of the NMF model have been reported on vocabulary learning in the ACORNS project [11]. However, there are several shortcomings of the model which have to be overcome towards better performance and larger data sets.

Accuracy loss in VQ

In the learning system, to apply the BoF representation, the input utterances have to be symbolized into sequences of codeword numbers. The VQ process transforms a vector into an integer, which causes a loss of discrimination between the observations. The situation is similar to early speech recognition systems, where VQ was applied to extract observations as inputs to an HMM system. In the improved ASR systems that followed, VQ was replaced by either VQ with multiple codebooks or density functions (e.g. Gaussian mixture models) which corresponds to (semi-) continuous density HMMs [53][125].

Correspondingly, in Chapter 3, we will improve the VQ-based NMF learning framework in a similar way by giving each frame a more precise description.

Local optima of NMF

The convergence of the algorithm to solve the NMF problem has been proved in [65], but no global optimum is guaranteed due to the non-convexity of the optimization problem with respect to $\{\mathbf{W}, \mathbf{H}\}$. So the algorithm can converge to a local optimum depending on its initialization [26]. Grounding supervision can help the model find solutions which match the vocabularies concerned. However, in unsupervised learning where grounding supervision is absent, with the high dimensionality of the features, the solution would probably be trapped at poor local optima.

Additional constraints can be applied to regularize the NMF learning towards better solutions than the original model. In Chapter 4, we model the acoustic features in a graph adjacency matrix and subsequently use the adjacency matrix to regularize the NMF.

High dimensionality

The above NMF learning models each word by its acoustic features directly which is a high dimensional vector. By following this framework, new vectors will be added to model new words. Therefore the computational budget is very high. With the high dimensional representation, the number of parameters to be estimated is also large.

Thus to improve the NMF model, dimension reduction techniques which impose structures in word descriptions are proposed in Chapter 5. By using non-negative matrix tri-factorization, sub-word units like hidden states are discovered as intermediate levels in the representation of words. The sub-word units with rich speech structures show better performance than acoustic codewords.

No word order in decoding

The BoF representation loses sequential information by putting all the features in a vector. With the obtained vocabulary patterns in the BoF form, unordered word error rates can be obtained easily. The unordered word error rate is a suitable metric to evaluate the quality of word representations, but not for normal speech decoding which aims at word sequences. The sliding window approach in [113] can partially overcome this problem. However, unlike HMMs, the sliding window approach has a low temporal resolution (window length) and it is difficult to detect repeated words.

In Chapter 6, we bridge the above gap between NMF and HMM by non-negative matrix tri-factorization. With a joint training framework, the final outputs are improved HMM parameters which can certainly be utilized for sequential decoding.

Chapter 3

Advanced BoF Representation of Speech

In Chapter 2, an NMF learning framework based on a bag-of-features (BoF) representations was presented. The features adopted in the BoF representations are co-occurrences of codewords. The vector quantization (VQ) process involved causes loss of description accuracy of the MFCC features. In this chapter, we improve the BoF representation of speech by looking for more precise frame coding than VQ, such as multiple codebooks and soft VQ as is also used in semi-continuous HMMs for ASR [53]. Asynchronous time scales to analyze the speech signal and multiple contextual dependencies like bigram and trigram language models, are also experimented with within the NMF framework. Some of the ideas will increase the computational complexity, and especially the memory usage, which will be monitored.

3.1 Multiple Codebooks and Soft VQ

As is presented in Chapter 2, the method to form the BoF representation is based on the co-occurrence frequencies of prototypical short-term speech spectra (histogram of acoustic co-occurrences, HAC), where finding a close prototype for a given spectrum of a frame involves VQ. The VQ process forces us to make compromises on the recognition accuracy that can be obtained with the NMF approach. While it has been shown in [113] that the approach can produce an accuracy 94.43% that is comparable to that 96.25%

of discrete density HMMs (DDHMM), the question naturally arises of how to generate accuracies that are comparable to those of continuous density HMMs (CDHMM). An obvious first approach is to increase the codebook size, such that the quantization error can be reduced. However, since the HAC representation used in the NMF model is based on co-occurrence statistics, we reckon that the amount of data required for training would scale quadratically with the codebook size, which is even worse than the linear scaling one observes with discrete density HMMs.

In this section, we report on our continued efforts to search for alternative representations that mitigate the loss due to quantization errors [99]. By exploiting the property that NMF can easily combine information from multiple streams, we look for ways to encode the spectral information with greater accuracy without increasing the complexity as much as one would by merely increasing the codebook size. Moreover, we keep in mind that our task is “recognition” and does not end at “coding”: recognition requires generalization to avoid over-learning.

We first review the procedure to make a HAC representation [113] of an utterance as illustrated in Figure 3.1. HAC is a special kind of BoF representation where the features are acoustic co-occurrences. An input utterance is processed as is common in speech recognition by hopping an analysis window (e.g. 25ms length) over the utterance by advancing it over regular frame shifts (e.g. 10ms) and computing a short term spectrum which is transformed to Mel Frequency Cepstral Coefficients (MFCCs), yielding a sequence of *static* (S) or MFCC stream vectors, one per frame. To emphasize the dynamics in speech, the first (*velocity* - V) and second order (*acceleration* - A) difference of this sequence, i.e. Δ and $\Delta\Delta$, is computed as well. Each of the three HAC-representation streams is quantized by its own codebook. The frequency with which two codewords of a stream co-occur at a fixed time difference or lag τ is then counted over the utterance. The number of bins in the resulting HAC representation equals the square of the codebook size. When multiple utterances are available, the HAC representations are stacked in a matrix \mathbf{V} : one column per utterance.

3.1.1 Frame Coding Methods

The performance of the NMF model depends on the quality of the HAC representation of speech, while in HAC the codebooks play an important role. Hence we first report how a codebook can be trained and then design methods to improve this kind of representation.

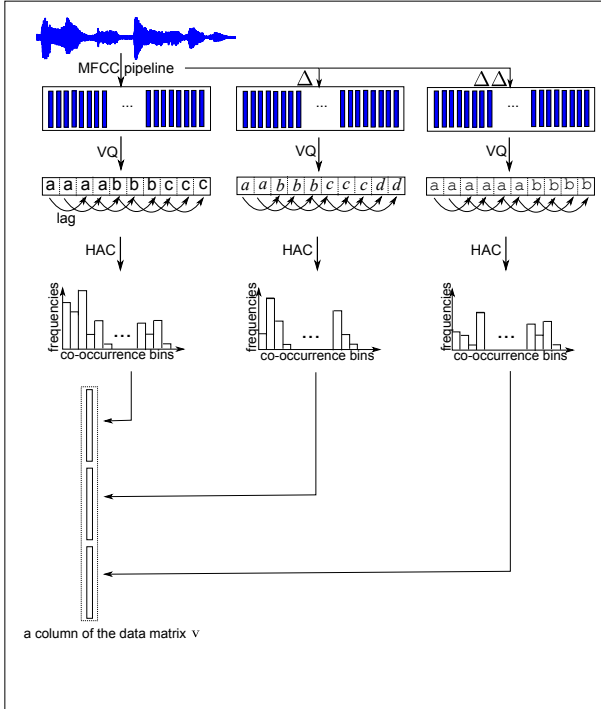


Figure 3.1: Diagram of HAC. Processing steps from top to bottom: (1) compute MFCC, Δ and $\Delta\Delta$, (2) vector quantization produces a label sequence, (3) compute frequency of co-occurrences of label pairs (HAC), (4) stack in a vector.

Observations of a given stream (MFCC, Δ or $\Delta\Delta$) can be encoded by a codebook pre-trained for the respective stream. A codebook contains a group of codewords where the codewords are k -means cluster centers of training data vectors. Since the results of k -means depend both on the data and on the initialization, we can train different codebooks for the same stream by every time randomly selecting a part of the whole training data and randomly initializing the k -means algorithm. As is shown in Figure 3.2, the consequence of training different codebooks is obtaining different Voronoi regions covering the training data. For example, the two points denoted by squares in Figure 3.2, are given completely different VQ labels from codebook 1. However, they are close to each other. Hence, an additional codebook could compensate for this situation, e.g. codebook 2, where the two neighboring points can obtain the same label. Hence, the hard decisions involved in vector quantization that limit the accuracy also have an arbitrary aspect, which can actually be used to our

advantage, as outlined below.

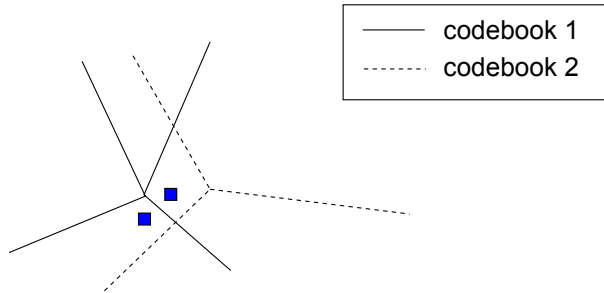


Figure 3.2: Codebooks and Voronoi regions partitioning the data space. Hard VQ can code two close data points by different codewords. With multi-codebooks, the two data points have a chance to be put in the same Voronoi region and thus to have the same codeword. Multi-codebooks can compensate for the accuracy loss in the hard decision with one codebook.

Multi-codebooks

A first technique that we explore is to use multiple codebooks with different Voronoi regions on the same information stream, a technique which has also been applied to HMMs [59]. Taking the MFCC stream as an example, suppose we train two different codebooks as explained above. Each codebook has M codewords and will yield a data matrix $\mathbf{V}^{(S,\tau,\text{cdb}_i)}$ with lag τ where $1 \leq i \leq 2$ is the codebook index. The data matrix of the MFCC stream will be expanded to $[(\mathbf{V}^{(S,\tau,\text{cdb}_1)})^T (\mathbf{V}^{(S,\tau,\text{cdb}_2)})^T]^T$.

The process to make a HAC representation with multiple codebooks is identical for the MFCC, Δ or $\Delta\Delta$ stream, for different lags. We therefore call the HAC computed by a particular choice of signal analysis parameters and lag and applying a given codebook to quantize a stream, a **view**. So in an example with 3 streams all with 10ms frame shift and a 25 ms window, each quantized with 2 codebooks, but using 3 different temporal lag values τ , there will be $3 \times 3 \times 2 = 18$ views. The data matrices made from Q different views, $\mathbf{V}^{(q)}$, $q = 1, 2, \dots, Q$, can be concatenated to yield the integrated acoustic data matrix for the training or the testing set. Here q denotes view, i.e. a choice for each

of the parameters: stream, lag and codebook. The data matrix is therefore,

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}^{(1)} \\ \mathbf{V}^{(2)} \\ \vdots \\ \mathbf{V}^{(Q)} \end{bmatrix}, \quad (3.1)$$

where each column of \mathbf{V} represents an utterance.

All HAC representations share that the observed histogram counts are the sum of co-occurrence frequencies of the words that make up the utterance. By stacking them into a supervector, we can obtain a more accurate description of the utterance. This way, the set of spectra that are mapped to the same quantized representation is reduced and hence quantization effects are reduced. However, each of the codebooks can remain small and the data requirements are not increased as we add more codebooks. Indeed, every view adds more parameters to be estimated, but also more data. Admittedly, such a representation is redundant and not optimal from the perspective of efficient coding. However, in this context, where the data requirements scale quadratically with codebook size and where NMF has been shown to be able to exploit information from redundant acoustic sources [75], this is not a major concern.

Soft VQ

A second technique that is explored here is soft VQ, i.e. a spectral data vector characterized by its proximity to multiple prototypes. A prototype is described probabilistically, i.e. instead of Voronoi regions with hard decision boundaries, the extent of a cluster is described by a probability density function. Proximity of a data point to the clusters is measured as the posterior probability of a collection of clusters, much like in semi-continuous HMMs [52]. To keep the NMF problem computationally feasible, we require that the data matrix is sufficiently sparse, which translates into the requirements that each spectral vector can only be characterized by its proximity of a “limited” set of prototypes.

For each stream, a codebook of I clusters was constructed with k -means clustering:

- cluster centers: $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_I$
- covariance matrix of the cluster: $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \dots, \boldsymbol{\Sigma}_I$

Each cluster is modeled by a multivariate Gaussian \mathcal{C}_i whose mean is $\boldsymbol{\mu}_i$ and whose full covariance matrix is $\boldsymbol{\Sigma}_i$. With this Gaussian assumption, the likelihood of the stream's data vector \mathbf{O}_t (at analysis frame t) on codeword i becomes,

$$\Pr(\mathbf{O}_t; \mathcal{C}_i) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_i|}} \exp\left\{-\frac{1}{2}(\mathbf{O}_t - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{O}_t - \boldsymbol{\mu}_i)\right\} \quad (3.2)$$

where D is the dimension of the stream. When labeling a frame, we assign a membership score proportional to its likelihood. To retain sparsity, only the top K -ranking clusters are retained:

$$\Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,1}}), \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,2}}), \dots, \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,K}}) \quad (3.3)$$

where $\mathcal{C}_{i_{t,1}}, \mathcal{C}_{i_{t,2}}, \dots, \mathcal{C}_{i_{t,K}}$ are the K Gaussians with the highest likelihoods for the frame \mathbf{O}_t . The normalized scores used for computing the co-occurrence probabilities are derived as the Gaussian posterior Gaussian probabilities. Let x_t denote the random variable for coding frame t as a cluster label. The distribution to characterize the frame \mathbf{O}_t is hereby,

$$\Pr(x_t = \mathcal{C}_{i_{t,k}}) = \frac{\Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,k}})}{\sum_{k'=1}^K \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,k'}})} \quad (3.4)$$

In the HAC-representation, the contribution to the co-occurrence of $\{\mathcal{C}_{i_{t,k}}, \mathcal{C}_{i'_{t+\tau,k'}}\}$ is given in Eq.(3.5), where t and $t + \tau$ denote the time index of the two frames \mathbf{O}_t and $\mathbf{O}_{t+\tau}$ which are separated by τ frames.

$$\Pr(x_t = \mathcal{C}_{i_{t,k}}, x_{t+\tau} = \mathcal{C}_{i_{t+\tau,k'}}) = \Pr(x_t = \mathcal{C}_{i_{t,k}}) \Pr(x_{t+\tau} = \mathcal{C}_{i_{t+\tau,k'}}) \quad (3.5)$$

Hence, the data matrix of the stream with lag τ should be constructed like,

$$\mathbf{V}_{i_{t,k}+i_{t+\tau,k'}*I,n} = \sum_{t=1}^{T_n-\tau} \Pr(x_t = \mathcal{C}_{i_{t,k}}, x_{t+\tau} = \mathcal{C}_{i_{t+\tau,k'}}) \quad (3.6)$$

where T_n is the number of frames in utterance n .

Adaptive VQ

A third technique, adaptive VQ, is proposed to enhance sparsity and to reduce model complexity as well as retaining the accuracy of frame coding with soft VQ. The number of Gaussians, K , used to label each frame is defined adaptively based on the observation likelihoods. The frames near the centroids will use a small number of Gaussians while the ones near the boundaries will get many activations on the nearby Gaussians. Two methods are proposed in this section.

One is to select K according to the differences of the sorted likelihoods. For an observation frame \mathbf{O}_t , suppose its decreasing likelihoods are,

$$\Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,1}}), \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,2}}), \dots, \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,I}}) \quad (3.7)$$

Their differences are,

$$\delta\Pr(\mathbf{O}_t, s) = \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,s}}) - \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,s+1}}), s = 1, 2, \dots, I - 1 \quad (3.8)$$

Intuitively, for each frame \mathbf{O}_t , we look for the *break point* K_t where the sorted likelihood scores of the clusters decrease abruptly from “important” to “less important” clusters:

$$K_t = \min(\operatorname{argmax}_s \delta\Pr(\mathbf{O}_t, s), 10) \quad (3.9)$$

Then K_t codewords are used for labeling frame t . K_t is selected adaptively for each frame. 10 is used to keep the sparsity of the coding, that is we select at most 10 codewords for each frame.

Another approach is setting a threshold, e.g. 1/10 of the largest likelihood,

$$\gamma_t = \frac{\Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,1}})}{10} \quad (3.10)$$

Then K_t is selected by the following formula.

$$K_t = \min(\operatorname{argmin}_s \Pr(\mathbf{O}_t; \mathcal{C}_{i_{t,s}}) > \gamma_t, 10) \quad (3.11)$$

where 10 is used to maintain the sparsity of the coding.

The following procedures to construct the data matrix are the same as in soft VQ.

3.1.2 Word Acquisition Results

The experiments were made on the ACORNS-Y2-UK database which is the English part of the ACORNS database where ACORNS (Acquisition of COmmunication and RecogNition Skills) is a project to research methods for computational language acquisition [11][10]. With this aim, 50 English words are selected as keywords from the list of words infants of about 12-15 months old are reported to understand. Every utterance in the data is constructed by organizing several keywords and carrier words (e.g. pronouns, prepositions etc.) in correct grammar. The number of keywords per utterance is between one and four. There are 9998 utterances in the training set and 3300 utterances in the test set. Each of the 50 keywords occurs at least 50 times across the entire

database to guarantee sufficient repetition. The database contains speech from 10 different speakers, six male and four female. This data was recorded at a sample rate of 44,100 Hz. For our experiments, however, we down-sampled it to 16,000 Hz.

The window length for spectral analysis was 25ms and the frame shift was 10ms. The MFCC extraction used 30 Mel-filter banks from which 12 MFCC coefficients are computed plus the frame’s log-energy. The codebook sizes for streams MFCC, Δ and $\Delta\Delta$ were 250, 250 and 100 respectively. 3% of the utterances were selected randomly to train a codebook for each stream. The lags between frames to define co-occurrences were 20, 50 and 90 ms (i.e. $\tau=2, 5, 9$ given the 10ms shift). The factorization dimension of NMF was $R=75$, which was larger than the number of keywords (50). This allows to model the non-keywords (filler words) occurring in the utterances. However, the total number of words in the database is far more than 75, so the model does not allow an accurate description of the filler words. NMF requires an iterative algorithm which was initialized as described in section 2.3 of Chapter 2.

To avoid the singularity of the covariance matrix of each cluster in soft VQ, principal direction bisection was used to ensure that every cluster has at least $10 \times D$ data samples (i.e. observations vectors) where $D=13$ is the dimension of each stream (MFCC, Δ or $\Delta\Delta$).

By thresholding these keyword activations, we can detect words in the testing utterances. The threshold value will trade off false alarms for missing detections. In our evaluation, we always choose the operating point where both error types have the same occurrence frequency, i.e. we report the *equal error rate (EER)* of word detection.

Since the NMF algorithm is not guaranteed to find the global minimum of its cost function, we always made 5 training attempts and report the mean error rate and the standard deviation. The mean values and standard deviations are shown in Table 3.1 to Table 3.3, the error rates versus the memory required for the NMF models are plotted in Figure 3.3.

Table 3.1: EER (%) of NMF with multi-codebooks on ACORNS-Y2-UK

# Codebooks	1	3	5	10	15
EER (%)	1.87±0.04	1.60±0.04	1.56±0.06	1.55±0.06	1.55±0.04

Table 3.2: EER (%) of NMF with soft VQ on ACORNS-Y2-UK

# Codebooks	1	2	3	5
$K = 1$	1.65 ± 0.04	1.50 ± 0.02	1.37 ± 0.04	1.39 ± 0.04
$K = 3$	1.33 ± 0.07	1.29 ± 0.06	1.25 ± 0.06	1.21 ± 0.08
$K = 5$	1.33 ± 0.08	1.26 ± 0.07	1.22 ± 0.05	1.21 ± 0.03

Table 3.3: EER (%) of NMF with adaptive VQ on ACORNS-Y2-UK

# Codebooks	1	3	5	10
difference-based	1.46 ± 0.06	1.36 ± 0.05	1.33 ± 0.04	1.32 ± 0.04
threshold-based	1.29 ± 0.05	1.24 ± 0.03	1.22 ± 0.03	1.26 ± 0.06

3.1.3 Discussion

The multi-codebook method successfully decreases the error rate with increasing number of codebooks in Table 3.1. Different codebooks were trained by using random data samples generated separately, hence the ability of generalization of the model was improved. However, the accuracy levels off around 5 to 10 codebooks. That’s probably because increasing the number of codebooks would also bring disagreements between the representations of different codebooks, given the limited training data.

By modeling each cluster (codeword) as a full-covariance Gaussian and making a soft assignment of MFCCs on the codewords, the performance of the NMF model was further improved as shown in Table 3.2.

Adaptive VQ can keep the good performance of soft VQ as shown in Table 3.3 while using a lower amount of memory than soft VQ by pruning the labels with small scores for each frame, as is shown in Figure 3.3.

Figure 3.3 shows the change in error rates with respect to the required memory. Representations with more views means a larger \mathbf{V} matrix and hence heavier computational load. Compared to the baseline in Figure 3.3, where we merely increase the codebook size, we do succeed in decreasing the error rates significantly. We also tried very large codebooks with 1500 codewords for stream S , 1500 codewords for stream V and 1000 codewords for stream A . The peak memory required was around 50 gigabytes but only yielded a poor performance with unordered word error rate 3.13 ± 0.12 . In the figure, we also find that applying soft VQ and adaptive VQ is a better compromise of required memory versus accuracy. $K = 3$ is a good choice for the number of codewords to be retained for each frame. We also checked the average number of labels

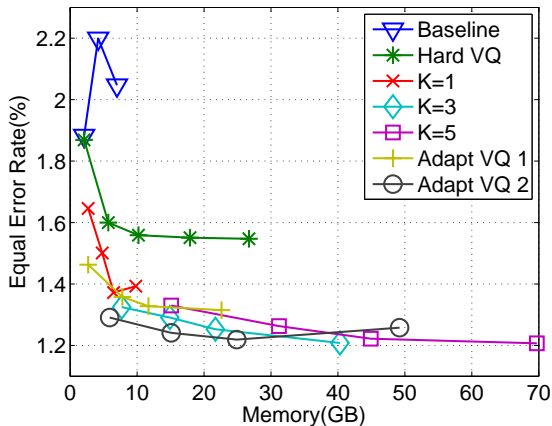


Figure 3.3: Tradeoff between accuracy and complexity (measured in required memory) for the coding methods on ACORNS-Y2-UK. “Baseline”: hard VQ with single codebook of increasing size. “Hard VQ”: increasing number of codebooks, each of size 250, 250 and 100 (for MFCC, Δ and $\Delta\Delta$). “ $K = 1, \dots, K = 5$ ”: soft VQ with increasing number of codebooks and with $K =$ number of retained softly assigned labels per frame. “Adaptive VQ”: soft VQ with increasing number of codebooks and a variable number of retained softly assigned labels per frame.

applied for each frame in the two adaptive VQ techniques. For the first one, the average number is 1.2, for the second one, it is 1.8. These small values suggest that the sparse representation of speech would be plausible and sufficient for word learning.

Notice that the criteria for labeling frames is different between hard VQ with 1 codebook in Table 3.1 and soft VQ with $K = 1$ in Table 3.2. The first one uses Euclidean distances, while the second one is tantamount to Mahalanobis distance.

With the proposed representation of speech and the NMF learning framework, and with the grounding supervision, recurring vocabularies can be discovered successfully. The accuracy has been improved by 35% with respect to the baseline by using the techniques of multiple codebooks, soft VQ and adaptive VQ. The ratio between the “gain” (the error rates) and the “pain” (the required memory) was also raised.

However, the performance levels off as we scale up the present approaches to more complex models with more codebooks. We can see that the error rates

will not decrease so much when using more than 5 codebooks (Figure 3.3). For further improvement of the NMF learning model, new acoustic features besides the codewords from MFCCs should be considered to overcome the difficulties of huge memory and possible disagreements between multi-view representations [17].

3.2 Multiple Temporal Scales and Asynchronous Streams

In the previous section, we observed the leveling-off of the performance with increasing number of codebooks both in hard VQ and soft VQ. It seems that 3 codewords to code one frame and using 3~5 codebooks would be sufficient to yield lower bound of error rates of the NMF learning model given the acoustic observations (MFCC, Δ and $\Delta\Delta$). However, all those representations are based on a fixed signal analysis design, e.g. 25ms window length and 10ms frame shift. Due to the non-stationary property of speech, it makes sense to investigate better representations with finer time scales, especially for the transient parts in speech - consonants. To validate this idea, we examine the recognition of consonants with multiple asynchronous analysis windows and shifts [100].

3.2.1 Motivations and Related Work for Multiple/Variate Frame Rates

Listeners outperform automatic speech recognition (ASR) systems at every level of speech recognition, including the very basic level of consonant recognition. However, what is not clear is where this human advantage originates from [93]. The transient parts in speech need to be given more attention because human perceptual mechanisms are more sensitive to those parts [44]. So finer time scales than the conventional analysis window could be better for the recognition of transient-like sounds in speech. Plosives for instance require an analysis at a finer time scale to separate the closure from the burst phase. However, short analysis windows are suboptimal for other consonants such as nasals, for these are characterized by (anti-)resonances in their spectrum, which are best analyzed with a window of a few pitch periods. If treating each phone class with its most suitable time scale, the recognition framework needs to deal with features computed at multiple time scales, possibly asynchronously.

Variable frame rate processing has been proposed to emphasize the dynamic aspects of speech [13]. The technique is to prune in the time domain and to discard the slowly changing frames. Hence, the transient part in speech will be emphasized. It is reported that, by decreasing the number of frames and by keeping the number of hidden states in HMMs, the recognition rate can be improved by reducing the insertion errors. However, meanwhile, the deletion errors can be increased. So an optimal threshold to balance the insertion and deletion errors should be found in the training and tuning procedure [73]. It could also lose useful information by throwing away the slowly changing frames. An alternative to treat the dynamic property of speech is using multiple temporal scales as is discussed below.

Multiple stream features with different temporal scales can be integrated by modifying the structure of the conventional HMM recognizer [9]. The proposed framework of a J -stream recognizer is illustrated in Figure 3.4. The system will extract several information streams from the incoming speech signal, each representing different properties of the speech signal and being treated independently up to some recombination point (\otimes in Figure 3.4), e.g. at the syllable level. In this context, the different streams are not restricted to the same frame rate and the underlying HMM models associated with each stream do not have to have the same topology. The construction of the system introduces much more parameters to be estimated during training than those in conventional HMMs, e.g. the weights of different streams and recombination criteria. Furthermore, during training it requires prior knowledge for dividing speech into multiple streams.

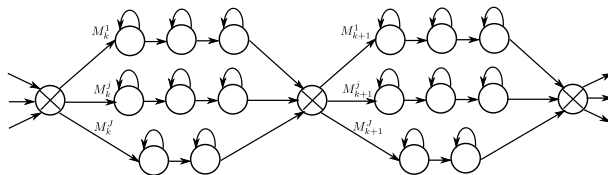


Figure 3.4: General form of a J -stream recognizer with anchor points between speech units (to force synchrony between different streams) [9]. The frame rate and model topology are not necessarily the same for the different streams.

In the previous work [75] and the last section, NMF has been proved to discover recurring word-sized units in utterances without relying on additional segmentation information. In the NMF model, an utterance is represented into a super-vector with multiple views. By treating the HAC vector generated with a *fixed window length and frame shift* as a *view* in Eq.(3.1), the data matrix can easily integrate information with multiple asynchronous time scales.

3.2.2 VCV Corpus and Articulatory Features

A Vowel-Consonant-Vowel (VCV) corpus [20] was used to evaluate the NMF approach on the recognition of smaller units than words: consonants (esp. plosive), by using analysis windows with multiple time scales. We investigate the recognition performance of input representations with different frame lengths and shifts and their combination.

NMF is supposed to be able to integrate features of different time scales [75]. This might be of particular importance for consonant classification (e.g. plosives). The VCV corpus is designed for evaluation of consonant recognition and classification and human performance on this task is shown. Hence it is our chosen database for evaluating the performance of NMF with speech representations using multiple temporal scales.

Corpus and Tasks

The dataset contains 24 consonants, *b, ch, d, dh, dj, f, g, h, k, l, m, n, ng, p, r, s, sh, t, th, v, w, y, z, zh*, in nine vowel contexts consisting of all possible combinations of the three vowels, *aa, iy, uu*, from 24 (male and female) speakers. The training set consists of 6,664 clean VCV utterances, while the clean test set has 384 utterances. For more information about the corpus, see [20].

The consonant recognition task is to find out which consonant is in each utterance. Specifically, the recognition of plosives was also made to test the performance of the model with high temporal resolution. The consonant in each utterance was also classified in terms of its articulatory features (AF). The AF considered are *manner, place* and *voicing*. *manner* has six values: *plosive, fricative, affricate, nasal, glide, liquid*, *place* has six values as well: *labial, alveolar, velar, palatal, dental, glottal*, and *voicing* has two values: *voiced (+voice)* and *unvoiced (-voice)*. The classification of AF is also tested.

Model Configuration

We take *consonant* recognition as an example to describe the model. The NMF models for the consonant/plosive recognition and articulatory feature classification share a similar structure.

Like in section 2.3 of Chapter 2, supervisory information is cast into the grounding matrix \mathbf{G} , as outlined below. For the training set, if the n -th utterance is known to contain the l -th consonant, construct the $L \times N$ grounding matrix \mathbf{G} with 1 in its l -th row and n -th column and 0 elsewhere, where L is

the total number of different consonants (or plosives, or AF values) and N is the number of training utterances. \mathbf{G}' denotes the grounding matrix of the test set. The oracle information described in \mathbf{G} helps to extract patterns that are associated with the ground truth.

The data matrix \mathbf{V} is constructed by stacking HAC representations of the utterances with one view or multiple views where each view here corresponds to an analysis scale (the pre-defined window length and frame shift).

3.2.3 Consonants Recognition and AF Classification Results

Three different values were used for the signal analysis window length (10ms, 20ms and 25ms) and three different values for the frame shift (2ms, 5ms and 10ms). The experiment indexes and the parameters are listed in Table 3.4, where $S, M, L, *$ denotes *Short, Medium, Long and all* respectively. So SM means *Short* window length (10ms) and *Medium* frame shift (5ms). The $*S$ means the combination of SS, MS, LS .

Table 3.4: Indices of experiments on VCV

index	SS	MS	LS	SM	MM	LM	SL	ML	LL
window	10ms	20ms	25ms	10ms	20ms	25ms	10ms	20ms	25ms
shift	2ms	2ms	2ms	5ms	5ms	5ms	10ms	10ms	10ms
lags, τ	[10,25,45]			[4,10,18]			[2,5,9]		

To make the results comparable, we used the same codebook for the experiments with the same window length. Hence, the co-occurrences at 10ms shift were a subset of that at 2ms shift. The lags τ were [10, 25, 45] for 2ms shift (in frames), [4, 10, 18] for 5ms shift and [2, 5, 9] for 10ms shift, so the actual elapsed time was the same in all cases: 20ms, 50ms and 90ms. Because plosives are short lived, using only the shorter time spacing of 20ms and 50ms (2 lags) was also evaluated and reported in Table 3.5. The codebook size for the *Static, Velocity and Acceleration* streams is 150, 100 and 50 respectively where smaller number of codewords were trained for the streams with less variations. The extracted feature vector is a 12-dimensional MFCC plus 1-dimensional energy computed from 30 MEL-filter banks. The factorization dimension was $R = 45$ for consonants, plosives and vowels, $R = 10$ for *manner*, $R = 10$ for *place* and $R = 3$ for *voice*. The initialization of \mathbf{W} and \mathbf{H} was done in the same way as in section 2.3.

To provide an idea of the significance of the differences, a 9-fold cross-validation was made. We took the original training/testing partition as *Fold 1*, then

Table 3.5: Accuracy (%) of consonant recognition and AF classification on Fold 1 of VCV

Features	Consonant	Plosive (3 lags)	Plosive (2 lags)	Manner	Place	Voice
SS	72.3	85.0	83.5	83.1	69.0	87.9
SM	77.7	80.4	83.5	86.6	72.0	90.1
SL	75.0	81.9	81.0	85.8	71.6	89.4
MS	76.5	82.5	83.3	82.9	71.4	88.8
MM	76.3	84.6	85.2	86.5	72.8	90.0
ML	73.1	73.3	69.6	84.4	71.4	89.3
LS	75.7	82.7	84.0	83.7	72.5	88.8
LM	75.1	80.0	79.8	85.8	71.6	89.4
LL	70.0	69.2	74.8	84.4	69.4	88.7
*S	76.0	85.6	86.9	82.6	71.5	89.6
*M	78.0	84.4	84.8	86.7	72.1	89.9
*S+*M	77.7	85.0	87.7	85.5	72.1	90.0
HMM+ ^[93] SVM	-	-	-	91.7	82.1	95.8
CDHMM ^[20]	88.5	-	-	-	-	-
Human ^[20]	93.8	-	-	-	-	-

divided the training set into 8 subsets with disjoint speakers. Each subset contains a male and a female speaker. In folds 2 through 9, the test set of fold 1 plus 7 subsets were used for training and the remaining subset was used for testing. The use of multiple folds also allows us to obtain an estimate of the accuracy of our experiments.

For the comparison with baselines, the results of consonant recognition and AF classification on the fold 1 are listed in Table 3.5 where each value is the mean of five NMF attempts using different random initializations. For the AF value classification, only the results with 5ms shift and the combined features are shown in Table 3.6. The performance of HMM+SVM is from [93] where HMMs were first trained using HTK for extracting the consonant segments and support vector machines (SVMs) were subsequently trained as classifiers on these segments. The number of HMMs was 30 where 24 of them modeled consonants and 6 of them modeled three vowels (one to model the initial and one to model the final vowel context of the VCV). Each HMM consisted of 3 emitting states with 32 Gaussian mixtures, while the silence model used 64 mixtures. The SVM training, development and test data sets were created by replacing the frame-level phonemic labels (from the above

Table 3.6: Average accuracy (%) of AF value classification on testset of *Fold1* of VCV. For single feature streams, only the ones yielding good performance (*SS, MM, LS*) are listed.

AF value	Accuracy(%)							# Test Utts.
	SS	MM	LS	*S	*M	*S+*M	HMM+SVM	
manner								384
plosive	81.9	84.0	83.5	82.3	88.5	91.7	96.9	96
fricative	83.8	90.4	80.6	77.8	94.4	77.8	96.5	144
affricate	86.3	88.8	91.3	90.6	90.6	90.6	87.5	32
nasal	83.8	75.0	83.3	83.3	87.5	85.4	85.4	48
glide	81.3	85.6	92.5	93.8	78.1	84.4	87.5	32
liquid	77.5	90.0	76.3	78.1	81.3	78.1	81.3	32
place								384
labial	71.3	83.8	77.9	68.8	75.0	80.2	94.8	96
alveolar	47.3	51.3	59.4	76.0	45.8	56.3	83.3	96
velar	78.3	72.9	80.0	68.8	79.2	85.4	81.3	48
palatal	81.9	86.5	84.8	76.0	94.8	84.4	90.6	96
dental	56.9	45.6	46.3	56.3	53.1	53.1	53.1	32
glottal	91.3	76.3	83.8	87.5	81.3	87.5	68.8	16
voicing								384
+voice	87.7	93.5	91.3	86.8	86.1	86.8	96.7	144
-voice	87.5	85.8	86.0	89.6	92.5	90.4	95.8	240

HMMs) with the corresponding canonical AF values. The RBF kernel was taken and the numbers of support vectors for each AF were also reported in [93]. The performance on consonant recognition using CDHMM is taken from [20] which used a 24-mixture CDHMM system with 3 state monophone models, based on the “standard” 39-dimensional MFCC_0_Z_D_A³ features.

The mean values and the error bars on the 9 Folds are shown in Figure 3.5 and Figure 3.6. The uncertainties in the two figures and in Table 3.8 are the empirical standard deviations of the mean accuracies over the 9 Folds. A detailed discussion of those figures and tables is provided below.

³In the configuration, “0” means the log-energy term, “Z” means cepstral mean subtraction and “D” and “A” refers to the delta and acceleration (delta+delta) features [125].

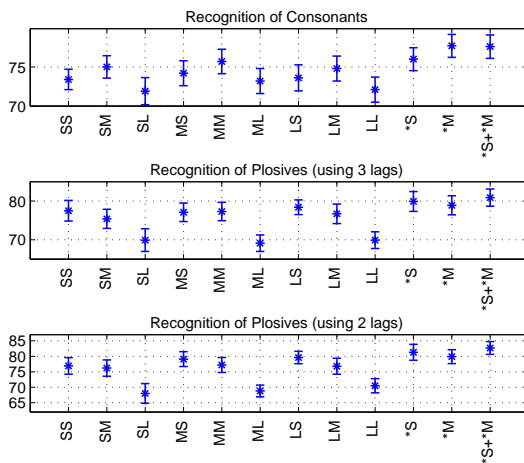


Figure 3.5: Recognition accuracy with error bars of consonants and plosives. The mean and uncertainty values were computed on the 9 Folds of VCV.

3.2.4 Discussion

The best performance on consonant recognition of the proposed model is 78%, while the human performance is 93.8% for native listeners [20]. The performance on the three AFs is, *voice>manner>place* (see Table 3.5 and Figure 3.6), which is the same as in [93]. But our results are 5% worse for *manner*, 10% worse for *place* and 6% worse for *voice* than those in [93], which is probably because we are not exploiting the segmentation information separating the consonant part from the vowel as in [93]. It is an advantage of the NMF model to achieve such recognition rates without prior segmentation information. However, for some AF values, *affricate, nasal, glide, liquid, velar, palatal, dental, glottal*, the model’s performance is better than that in [93]. Since the performance figures of [93] are only available on fold 1, the statistical significance of such a comparison is limited. Some further comments are given below.

Frame Shift

As can be seen from the tables, 10ms shift is always worse than 2ms and 5ms shift. 5ms shift seems to make a good compromise. At 10ms, the amount of training data is strongly reduced. For a given frame shift at recognition

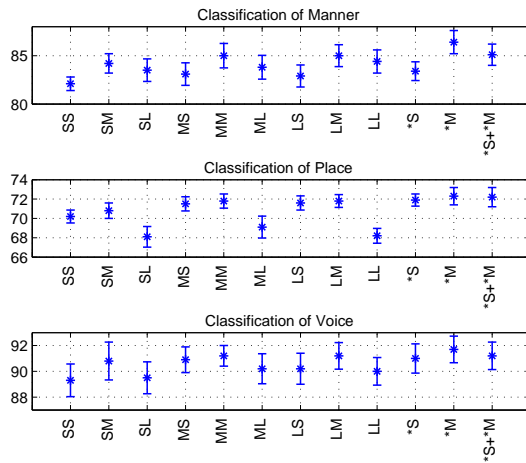


Figure 3.6: Classification accuracy with error bars of the three articulatory features. The mean and uncertainty values were computed on the 9 Folds of VCV.

time, one could argue that training with shorter frame shifts should yield better results, since it generates more training data: the co-occurrence data at the larger shift plus the interleaved data. Indeed, the co-occurrence symbol pairs observed with a frame shift 2β ms are a subsequence of the symbol pairs generated with a frame shift β ms. Hence, the 10ms frame shift data is just one sample out of 5 w.r.t. the 2ms frame shift data. From Table 3.7, for *fold1*, we can see matched shifts are in fact better. The “matched” here means that the training and testing used the same frame shift. Hence, the NMF model also captures the relations between the co-occurrence events, which is broken in unmatched conditions. Another interesting observation is that the top right results in Table 3.7 are worse than the bottom left ones, i.e. making the test data more sparse (by decimation) is worse than making the training data more sparse on the VCV dataset.

Window Length

The best results on consonant recognition are obtained with a 10ms or 20ms window in Table 3.5 and Figure 3.5. We expected to get the best performance on plosive recognition using short window lengths, but it is not always true as the medium window length also performs well with short and medium frame

shifts in Table 3.5 and Figure 3.5. Shorter lags do work better as expected. Notice also that the comparison is a difficult one since different window lengths also lead to different codebooks. Thus we suggest to use 2ms or 5ms frame shifts in combination with the optimal windows (10ms,20ms).

Combinatorial information

From Table 3.5, Figure 3.5 and Figure 3.6, it was observed that combining all window sizes improves the accuracy over single time scale analysis. Combining short and medium frame shifts also does not hurt accuracy. Moreover, the medium shift does provide a new sample of co-occurrence. We hence advise to use all available information sources.

Does the gain of combinatorial features come from *multi-codebooks*?

One thing one should notice is that by using the $*S$, $*M$ and $*S+*M$ features we actually used multi-codebooks with different time scales implicitly because at each time scale the model uses a different codebook. The technique of multi-codebooks is making an acoustic matrix $\mathbf{V}^{(q)}$ by each codebook with id q and concatenating them to form a new acoustic matrix $\mathbf{V} = [\mathbf{V}^{(1)}; \dots; \mathbf{V}^{(Q)}]$ as is explained in section 3.1.1 where multi-codebook representations also yielded better performance than the single-codebook one.

Experiments were made on the 9 Folds to check if the improvement of combinatorial features comes from *multiple time scales* or *multi-codebooks*. The results are in Table 3.8 where $3 \times SS$ means 3 codebooks with *Short* window length and *Short* frame shift. So the acoustic feature matrix of 3 (6) codebooks has the same size as the ones of $*S$ and $*M$ ($*S+*M$). There are indeed improvements from one codebook to 3 or 6 codebooks for all the features, but not as large as that by using features with multiple time scales. Thus we conclude that the gain comes from the combined information at different time scales.

In this section, using the NMF model, different window lengths and shifts were compared on the recognition of consonants and plosives and the classification of articulatory features on the VCV corpus. There is still a lot of room for improvement of the recognition results. The AF classification results were not as good as what is presented in [93], which can be attributed to the fact that we do not rely on segmentation of the consonant parts in the utterances. However, the performance on some AF values outperformed that of the HMM+SVM model. The 10ms frame shift that has settled as a compromise for the HMM

Table 3.7: Training and recognition (accuracy in %) with different shift on testset of *Fold1* of VCV

Train \ Test		MS	MM	ML
	MS	manner	84.2±2.0	69.6±1.6
place		70.6±1.6	66.3±0.8	55.4±1.0
voice		89.0±1.3	86.6±1.9	72.4±4.6
MM	manner	79.2±2.3	85.7±1.7	76.5±1.5
	place	70.5±2.8	73.2±1.7	67.3±2.8
	voice	87.7±1.0	90.5±0.7	88.3±0.2
ML	manner	77.8±3.2	83.5±2.6	85.2±1.4
	place	69.8±2.5	72.7±1.8	71.4±2.6
	voice	83.3±6.3	88.5±1.0	88.6±1.6

Table 3.8: Consonant recognition rates (%) on the 9 folds of VCV using multi-codebooks

	1×SS	1×MS	1×LS	1×SM	1×MM	1×LM	1×ML	1×LL
Mean	73.4	74.2	73.6	75.0	75.7	74.8	73.2	72.1
Unc.	1.3	1.6	1.7	1.1	1.6	1.6	1.6	1.6
	3×SS	3×MS	3×LS	3×SM	3×MM	3×LM	*S	*M
Mean	74.6	74.7	74.4	75.8	76.2	75.6	76.0	77.7
Unc.	1.4	1.4	1.6	1.3	1.7	1.6	1.1	1.1
	6×SS	6×MS	6×LS	6×SM	6×MM	6×LM	*S+*M	
Mean	74.3	74.3	74.1	75.5	75.8	75.6	77.6	
Unc.	1.3	1.5	1.7	1.2	1.7	1.6	1.5	

recognizer [59] is better replaced by 5ms in the NMF method for consonant recognition. The multiple window length and frame shift idea seemed to work indeed: its performance is always better than its components because it probably incorporates the complementary features of the different analyses. Furthermore, the NMF model is capable of exploiting asynchronous multi-scale information.

While this analysis focused on the representation of speech, we should be aware that a lot of parameters can affect the actual recognition results, e.g. codebook sizes, parameters in MFCC extraction, temporal lags, factorization dimensions and the NMF initial values. We controlled these parameters in the experiments as exploring the parameters would require more data.

3.3 Multi-gram Models from Gaussian Posteriorgrams of Speech

In the previous section, we studied the speech representation using multiple analysis time scales. The performance of NMF has been improved with respect to those with a single analysis scale. But they are still not as good as CDHMM recognizers. In this section we make a more direct comparison of the performance of NMF models and CDHMM recognizers. A priori, it is not clear if the difference comes from the codewords used in the speech representation (HAC, probability density function, etc.), from the learning framework or from the supervision information utilized for training. In this section, to compare our model with a CDHMM recognizer, we apply the Gaussians trained from this CDHMM as codewords in our NMF framework, such that this factor is removed and the model and learning method remain. Also by choice of recognition task and database, we will minimize the differences in supervisory information, as will be further explained in section 3.3.3.

3.3.1 Gaussian Posteriorgram Representation

In a state-of-the-art CDHMM recognizer, the observation of a frame is described by its MFCC+ Δ + $\Delta\Delta$ denoted by \mathbf{O}_t and a state is modeled by a Gaussian mixture model (GMM) which is a weighted combination of a couple of Gaussians. In this section, we use the Gaussians gathered from the GMMs of the pre-trained CDHMM. The introduction of CDHMM was given in section 1.2.2 where different Gaussians are utilized to model different hidden states. In this section, the Gaussian weights in each state and the state transitions are disregarded.

The Gaussian \mathcal{G}_m is utilized as a codeword here, whose centroid is $\boldsymbol{\mu}_m = [\mu_{1,m}, \dots, \mu_{D,m}]^T$ and whose diagonal covariance matrix is $\boldsymbol{\Sigma}_m = \text{diag}(\sigma_{1,m}^2, \dots, \sigma_{D,m}^2)$. The likelihood of the D dimensional frame \mathbf{O}_t on Gaussian \mathcal{G}_m is,

$$\Pr(\mathbf{O}_t; \mathcal{G}_m) = \frac{1}{\sqrt{(2\pi)^D \prod_d \sigma_{d,m}^2}} \exp\left(-\sum_{d=1}^D \frac{(O_{d,t} - \mu_{d,m})^2}{2\sigma_{d,m}^2}\right) \quad (3.12)$$

A frame is therefore encoded by the posterior probabilities on the Gaussians.

$$\Pr(\mathcal{G}_m | \mathbf{O}_t) = \frac{\Pr(\mathbf{O}_t; \mathcal{G}_m)}{\sum_{m'=1}^M \Pr(\mathbf{O}_t; \mathcal{G}_{m'})}, \quad (3.13)$$

where M is the total number of Gaussians. By using

$$\mathbf{X}_t = [\Pr(\mathcal{G}_1|\mathbf{O}_t), \dots, \Pr(\mathcal{G}_M|\mathbf{O}_t)]^T \quad (3.14)$$

to denote the posterior probabilities of a frame, the new representation $\{\mathbf{X}_t, t = 1, \dots, T\}$ is called the *posteriorgram* of the utterance. Like in soft VQ, sparsity of the representation can be obtained by retaining only the likelihoods of the top K Gaussians and setting the rest to zero.

3.3.2 BoF Representations Derived from Gaussian Posteriorgram

The illustrations of creating multi-gram models from the Gaussian posteriorgram are given in Figure 3.7.

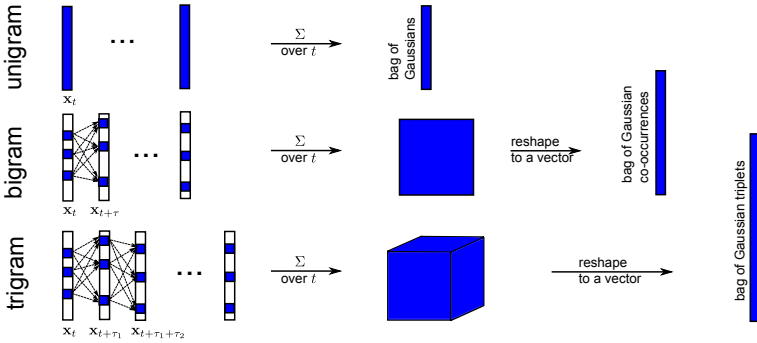


Figure 3.7: Constructing the multi-gram representations from Gaussian posteriorgram of an utterance.

Gaussian unigram

In the unigram model, an utterance is represented by the accumulated posterior probabilities of Gaussians which is called bag of Gaussians. The accumulated probabilities are stored in a $M \times 1$ vector where M is the number of Gaussians. So the corresponding column of the n -th utterance in the acoustic matrix \mathbf{V} are,

$$\mathbf{V}_n = \sum_{t=1}^{T_n} \mathbf{X}_t \quad (3.15)$$

where T_n is the number of frames of the utterance, $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_{T_n}]$ is the Gaussian posteriorgram. The process is depicted in the row of “unigram” in Figure 3.7.

Bigram: Gaussian co-occurrences

In the bigram model, an utterance is represented by the accumulated probabilities of Gaussian co-occurrences, i.e. bag of Gaussian co-occurrences. Hence the row size of \mathbf{V} becomes M^2 . Each frame \mathbf{O}_t is first represented by its posterior probabilities on the Gaussians like in Eq.(3.14). To retain sparsity, only a limited number of Gaussians, say top- K , are selected to label the frame. Let \mathbf{I}_t denote the indices of the retained Gaussians of frame t . This technique for feature selection is called **FeatSel-1** in this work.

$$\mathbf{V}_{\mathbf{I}_t \otimes \mathbf{I}_{t+\tau}, n} \leftarrow \mathbf{V}_{\mathbf{I}_t \otimes \mathbf{I}_{t+\tau}, n} + \mathbf{X}_{\mathbf{I}_t} \otimes \mathbf{X}_{\mathbf{I}_{t+\tau}} \quad (3.16)$$

where \otimes is the Kronecker product, τ is the lag parameter between two frames and $\mathbf{V}_{\mathbf{I}_t \otimes \mathbf{I}_{t+\tau}, n}$ is a sub-vector of \mathbf{V}_n with indices $\mathbf{I}_t \otimes \mathbf{I}_{t+\tau}$ and \mathbf{V}_n is the n -th column of the data matrix \mathbf{V} . Eq.(3.16) should be operated over t to yield \mathbf{V}_n . The process is depicted in the row labeled “bigram” in Figure 3.7 where the Kronecker product of two vectors is a Gaussian \times Gaussian matrix which is further flattened to a column of the data matrix.

Besides the above technique of only taking the top activated Gaussians for each frame, a second technique for feature selection is taking the top activated Gaussian co-occurrences according to their total activations in the whole training data. The sum of the columns of the data matrix, $\sum_n V_{i,n}$ or the sum of the columns of a learned pattern matrix, $\sum_r W_{i,r}$ from the NMF of Chapter 2, can work for this purpose. We will use a more advance method, however. From Chapter 2, we know that with grounding supervision some of the columns of \mathbf{W} correspond to keywords while others do not. In this section, we use the columns of \mathbf{W} corresponding to keywords to prune the Gaussian co-occurrences because the NMF learning process can help to remove the possible co-occurrences which are not related to keywords. Therefore the indices of the selected Gaussian co-occurrences, by sorting

$\sum_{k \in \{\text{word patterns}\}} W_{i,k}$ w.r.t. i and truncating, are stored in $\bar{\mathbf{I}}_1$ and $\bar{\mathbf{I}}_2$ which are the indices of the first and second Gaussians of the selected Gaussian co-occurrences. Let M' denote the number of Gaussian co-occurrences selected. So for frame t and $t + \tau$ the retained Gaussian co-occurrences are with first indices,

$$\hat{\mathbf{I}}_t \leftarrow \mathbf{I}_t \cap \bar{\mathbf{I}}_1, \quad (3.17)$$

and with the corresponding second indices,

$$\hat{\mathbf{I}}_{t+\tau} \leftarrow \mathbf{I}_{t+\tau} \cap \bar{\mathbf{I}}_2. \quad (3.18)$$

The histogram of co-occurrences of Gaussians for frame-pair $(\mathbf{X}_t, \mathbf{X}_{t+\tau})$ is then computed by Eq.(3.16) using $\hat{\mathbf{I}}_t$ and $\hat{\mathbf{I}}_{t+\tau}$. The BoF representation of the utterance is obtained by sliding over all the frame-pairs and accumulating the co-occurrences. This technique is called **FeatSel-2**.

Finally, a threshold γ_v can be taken to remove the rows of \mathbf{V} with tiny values,

$$\mathbf{V}_{\mathbf{I}_v, n} = 0, \quad (3.19)$$

where the set of row indices is $\mathbf{I}_v = \{i | \sum_n V_{i,n} < \gamma_v\}$. The final one is named by **FeatSel-3**.

Trigram: Gaussian triplets

In the trigram model, an utterance is represented by the accumulated probabilities of Gaussian triplets. Hence the row size of \mathbf{V} is M^3 . Again, using **FeatSel-1**, top- K Gaussians are retained to label the frame. The data matrix is constructed by,

$$\mathbf{V}_{\mathbf{I}_t \otimes \mathbf{I}_{t+\tau_1} \otimes \mathbf{I}_{t+\tau_2}, n} \leftarrow \mathbf{V}_{\mathbf{I}_t \otimes \mathbf{I}_{t+\tau_1} \otimes \mathbf{I}_{t+\tau_2}, n} + \mathbf{X}_{\mathbf{I}_t} \otimes \mathbf{X}_{\mathbf{I}_{t+\tau_1}} \otimes \mathbf{X}_{\mathbf{I}_{t+\tau_2}}, \quad (3.20)$$

where \mathbf{I}_t denotes the indices of the retained Gaussians of frame t , τ_1 is the lag parameter between the first frame and the second frame, τ_2 is the lag parameter between the second frame and the third frame and $\mathbf{V}_{\mathbf{I}_t \otimes \mathbf{I}_{t+\tau_1} \otimes \mathbf{I}_{t+\tau_2}, n}$ is a sub-vector of the n -th column \mathbf{V}_n . By operating Eq.(3.20), we obtain \mathbf{V}_n for the trigram model. The process is depicted in the row of “trigram” in Figure 3.7 where the Kronecker products of three vectors with time space τ_1 and τ_2 is a Gaussian \times Gaussian \times Gaussian tensor which is further transformed to be a column of the data matrix.

To make feature selection based on the retained Gaussian co-occurrences in **FeatSel-2**, we first use $\bar{\mathbf{I}}_1, \bar{\mathbf{I}}_2$ to construct the triplets of Gaussians. The Gaussian in the middle of a triplet should be the intersection of $\bar{\mathbf{I}}_1$ and $\bar{\mathbf{I}}_2$ to ensure that the chosen Gaussian can be both head and tail in the bigram model to connect the first Gaussian and the third Gaussian. That is,

$$\bar{\mathbf{I}}_2 \leftarrow \bar{\mathbf{I}}_1 \cap \bar{\mathbf{I}}_2. \quad (3.21)$$

The indices of the third Gaussians $\bar{\mathbf{I}}_3$ in valid triplets are thus derived from $\bar{\mathbf{I}}_1$ and the updated $\bar{\mathbf{I}}_2$. The indices of the retained Gaussian triplets for frame t ,

$t + \tau_1$ and $t + \tau_1 + \tau_2$ are updated as follows.

$$\begin{aligned} \hat{\mathbf{I}}_t &\leftarrow \mathbf{I}_t \cap \bar{\mathbf{I}}_1, \\ \hat{\mathbf{I}}_{t+\tau_1} &\leftarrow \mathbf{I}_{t+\tau_1} \cap \bar{\mathbf{I}}_2, \\ \hat{\mathbf{I}}_{t+\tau_1+\tau_2} &\leftarrow \mathbf{I}_{t+\tau_1+\tau_2} \cap \bar{\mathbf{I}}_3. \end{aligned} \tag{3.22}$$

The histogram of triplets of Gaussians for frame-triplets $(\mathbf{X}_t, \mathbf{X}_{t+\tau_1}, \mathbf{X}_{t+\tau_1+\tau_2})$ is then computed by Eq.(3.20) using $\hat{\mathbf{I}}_t$, $\hat{\mathbf{I}}_{t+\tau_1}$ and $\hat{\mathbf{I}}_{t+\tau_1+\tau_2}$. The BoF representation of the utterance is obtained by sliding over all the frame-triplets and accumulating the probabilities.

Similarly, a threshold γ_v can also be taken to remove rows of \mathbf{V} with only tiny values as in Eq.(3.19), which is **FeatSel-3** for the trigram model.

3.3.3 Computation, Results and Analysis

The experiments were made on the TIDIGITS database which contains 11 English digits with 8438 training utterances and 1001 testing utterances. The motivation to evaluate the models on this database is twofold. Firstly, unlike in the ACORNS-Y2-UK database, TIDIGITS has no garbage or filler words, i.e. all words are keywords to be learned. With the presence of filler words, the comparison between NMF and HMM would give an advantage to HMM for which the training will have the information on the filler words. Secondly a sufficient number of speakers is required to test speaker independence of the model. The VCV database is only suitable for the evaluation of consonant recognition and there is insufficient data for word learning. TIDIGITS is thus our database of choice in the following experiments.

Before starting, $M=3628$ Gaussians are gathered from a CDHMM model trained by HTK [125] on the TIDIGIT database where the basic frame-level representations are MFCC+ Δ + $\Delta\Delta$ vectors with dimension $D = 39$. The evaluation of the CDHMM recognizer is word error rate which is 0.16% here. To be comparable to the evaluation of the NMF models on vocabulary discovery, we transform the word error rate to unordered word error rate firstly. By only counting and accumulating the substitution of unique digits in one utterance, the corresponding CDHMM baseline in unordered WER is 0.15%.

The computational complexity of NMF depends on the sizes of the matrices. The positive entries in \mathbf{W} and \mathbf{H} are bounded by $\# \text{ features} \times R$ and $R \times N$ respectively. However, due to the sparsity of the data matrix \mathbf{V} , the number of positive entries can vary in a wide range which is roughly $\# \text{ features} \times \text{sparsity} \times N$. In the following experiments, we use *memory* as a

measurement of complexity. The memory is taken as the peak value observed during execution of the program.

Results of NMF models with soft assignment of Gaussians

In the NMF models with soft assignment of Gaussians, each frame is labeled by the top K posterior probabilities on the Gaussians. The unordered word error rates (UWER) with respect the model parameters are shown in Table 3.9, Table 3.10 and Table 3.11. For the results of unigram and bigram models, the reported UWER are mean values and standard deviations of five NMF attempts.

Table 3.9: UWER on TIDIGITS using the NMF unigram model

NMF Dimension R	11	12	15	25
UWER (%)	1.09±0.00	1.10±0.01	1.31±0.16	5.06±0.84

Table 3.10: UWER on TIDIGITS using the NMF bigram model (FeatSel-1)

NMF Dim. R	# Gau. per frame K	lags τ	UWER(%)	Memory (GB)
12	3	[2,3,5]	0.85±0.02	26
12	3	[2,3,5,7]	0.77±0.01	36
12	3	[2,5,9]	0.73±0.03	26
15	3	[2,5,9]	0.88±0.02	26
15	5	[2,5,9]	0.70±0.12	36

Table 3.11: UWER on TIDIGITS using the NMF trigram model (FeatSel-1 and FeatSel-3). The NMF dimension R is 12. For each K , within the computational budget, we selected as small as a threshold to retain as much as information.

# Gau. per frame K	threshold γ_v	lags τ_1	lags τ_2	UWER (%)	Memory (GB)
2	2.0e-1	[2,5]	[5,9]	7.60	30
2	1.5e-1	[2,5]	[5,9]	7.81	42
3	8.0e-1	[2,5]	[5,9]	6.35	35
3	5.0e-1	[2,5]	[5,9]	5.45	46
4	1.0e0	[2,5]	[5,9]	12.55	50

The factorization dimension R affects the performance of the model. The dimension being equal or slightly larger than the number of digits seems to be a good choice. Additional columns of \mathbf{W} seem not to be necessary and

probably harmful in this NMF learning model with grounding supervision and the special initialization described in section 2.3.

The performance improves from the unigram model to the bigram model as expected. From Table 3.10, we can see that retaining a larger number of Gaussians per frame (from $K = 3$ to $K = 5$) and richer contextual dependencies (from $\tau = [2, 3, 5]$ to $\tau = [2, 3, 5, 7]$) help to obtain better results. However, both of them increase the computational complexity too.

The trigram model did not yield good results given the reported parameter settings. With an increasing number of Gaussians retained per frame (K) and with a decreasing of the threshold γ_v , the model’s performance becomes better. However, with the current model and computation load it cannot obtain as good a performance as that the bigram model offers.

Results of NMF models with pruning of Gaussian co-occurrences

Given the heavy complexity in the bigram and trigram experiments, it is necessary to prune the Gaussian co-occurrences. The pattern matrix \mathbf{W} learned by the NMF bigram model with $K = 5$ was applied in FeatSel-2, i.e. to select the top M' activated Gaussian co-occurrences by sorting $\{i | \sum_{k \in \{\text{word patterns}\}} W_{i,k}\}$. The related results are reported in Table 3.12 and Table 3.13.

With the pruning of Gaussian co-occurrences, we can use the same computational budget to tackle a data matrix generated with larger K values, i.e. to retain more (important) Gaussians per frame-pair based on the selected Gaussian co-occurrences. Improvement is observed with respect to the bigram model in Table 3.10. The trigram model still does not work very well. We will analyze the possible reasons in the next section.

Table 3.12: UWER of the NMF bigram model with pruning of Gaussian co-occurrences (FeatSel-1 and FeatSel-2)

# of Gau. per frame, K	All	5	5	10	20
# of Gau. Co-occ., M'	10000	100000	400000	1000000	400000
UWER(%)	6.36	0.84	0.69	0.58	1.09
Memory(GB)	1	1	4	36	45

Table 3.13: UWER of the NMF trigram model with pruning of Gaussian co-occurrences (FeatSel-1 and FeatSel-2)

# of Gau. per frame, K	5	5
# of Gau. Co-occ., M'	10000	100000
UWER(%)	35.26	6.91
Memory(GB)	2G	22G

Combination of the unigram model, the bigram model and the trigram model

The first try was to combine the unigram model and the bigram model. The model of *unigram+bigram* performs similarly (UWER 0.79 ± 0.02) with the *bigram* model (UWER 0.73 ± 0.03). There seems no added value of combining the two. Subsequently we examined that if the combination of the bigram model and the trigram model performs better than any single model.

The trigram model did not produce promising results in the above experiments. We therefore check if it can help to improve the bigram model in a combined model by offering supplementary information. Suppose that the data matrix constructed by the bigram model is $\mathbf{V}^{(b)}$ and the matrix constructed by the trigram model is $\mathbf{V}^{(t)}$, we can merge the two matrices into a big one in the NMF learning framework in Eq.(3.23).

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{V}^{(b)} \\ \mathbf{V}^{(t)} \end{bmatrix} = \begin{bmatrix} \mathbf{W}^{(g)} \\ \mathbf{W}^{(b)} \\ \mathbf{W}^{(t)} \end{bmatrix} H \quad (3.23)$$

However, the performance is not improved as is listed in Table 3.14. By looking into the obtained pattern matrix $\mathbf{W} = [\mathbf{W}^{(b)}; \mathbf{W}^{(t)}]$, the possible reason could be that, much fewer trigram features than bigram features are retained if a small threshold γ_v is taken as in the attempts 1,3,4 in Table 3.15. The trigram features have almost vanished. On the other hand, a large threshold made the bigram features too rare as in attempt 5 in Table 3.15 which completely ruins the performance of the model. Given this fact, the distribution of the activations over the Gaussian triplets are probably highly non-uniform where only a few triplets corresponding to silence or garbage get high values in the data matrix \mathbf{V} , while the triplets corresponding to meaningful word patterns are too rare to be selected as features. The rarity of useful triplets could be due to the large thresholds, but the current computational budget can not pay for the cases with small enough thresholds. Another reason could be that

the triplets of Gaussians are not good acoustic cues for speech representation because the same speech unit perhaps has different realizations by Gaussian triplets given the variation in speakers and contexts.

Table 3.14: UWER of the Bigram+Trigram model with FeatSel-1, FeatSel-2 and FeatSel-3

Exp. ID	Bigram Part			Trigram Part			threshold	UWER
	M'	K	τ	K	τ_1	τ_2	γ_v	(%)
1	400K	5	[2,5,9]	1	[2,5]	[5,9]	1.0e-10	0.69
2	400K	5	[2,5,9]	-	-	-	-	0.69
3	1M	3	[2,5,9]	1	[2,5]	[5,9]	1.0e-10	0.62
4	1M	10	[2,5,9]	1	[2,5]	[5,9]	1.0e-10	0.58
5	1M	10	[2,5,9]	2	[2,5]	[5,9]	5.0e-1	4.91
6	1M	10	[2,5,9]	-	-	-	-	0.58

Table 3.15: Comparison of the bigram part and the trigram part of the learned pattern matrix

Exp. ID	threshold γ_v	Bigram		Trigram	
		# retained rows in $\mathbf{W}^{(b)}$, M'	activation $\sum_{ik} W_{ik}^{(b)}$	# retained rows in $\mathbf{W}^{(t)}$	activation $\sum_{ik} W_{ik}^{(t)}$
1	1.0e-10	1,200,000	6.1e0	48817	3.1e-6
3	1.0e-10	2,679,225	6.2e0	48817	2.0e-6
4	1.0e-10	2,999,989	6.0e0	48817	1.3e-5
5	5.0e-1	12,960	4.5e-1	2,168,681	5.6e0

Small number of Gaussians

The above experiments tell us that the *computational complexity* and the *rarity of features* prevent us from exploiting the rich contextual information by using the Gaussian triplets as features in BoF representations. To demonstrate that a multi-gram model with sufficiently retained amount of triplets can perform better than its unigram and bigram counterparts, we apply all the above unigram, bigram and trigram models on a small set of 100 Gaussians trained by CDHMM again using HTK [125]. 100 is chosen as the largest number of Gaussians whose experiments using the trigram model are feasible on our machine. The unordered word error rates are listed in Table 3.16.

Table 3.16: UWER of the NMF multi-gram models with 100 Gaussians from a CDHMM (FeatSel-1 and FeatSel-3)

	# Gau. per frame, K	threshold, γ_v	UWER (%)
unigram	100	0	5.89
bigram	10	0	1.42
trigram	5	1.0e-1	0.94
HMM	-	-	0.55

Given the large decrease of UWER from the unigram model to the bigram model with both 100 Gaussians and 3628 Gaussians, we can see that the bigram model contains very important contextual information. The trigram model with 100 Gaussians comes even closer to the HMM baseline. However, higher number of Gaussians and lower threshold may optimize this result further. Note that the supervision provided to the CDHMM and the NMF is different, where the first one is the transcriptions of the training utterances with word orders while the second one only indicates which words are in the utterances. To put rich contextual information into the NMF model, we have to pay attention to the rarity of features and the computational complexity.

By comparing the performance gain from the unigram model to the bigram model and that from the bigram model to the trigram model, we conjecture that there is also a limit on the model’s performance when increasing the contextual dependencies even without the problem of feature rarity and given sufficient computational resources.

3.4 Conclusion

In this chapter, we have applied the techniques of multiple codebooks, multiple asynchronous analysis temporal scales and rich contextual dependencies to the proposed NMF learning framework. The first two have been proved useful to improve the baseline of NMF with hard VQ and single analysis scale. The positive effect on performance suggest the methods can be valid choices to improve speech representation in the future work.

A question we tried to answer was if the relatively worse performance of NMF w.r.t. HMMs (on consonant recognition and keyword recognition) comes from the model structure or from the speech representation. This was tested by using Gaussians from a pre-trained CDHMM as codewords. Towards rich contextual representations, besides the unigram model with Gaussian occurrence and the bigram model with Gaussians co-occurrences, the trigram model with Gaussian

triplets was also evaluated. Significant improvement was observed from the unigram model to the bigram model. However, due to the huge computational load required, the trigram idea can only give promising results with a small number of Gaussians. The performance of NMF was shown to get close to the HMM baseline with supervised training.

Chapter 4

Graph Regularized NMF for Unsupervised Pattern Discovery

The limitations of directly modeling vocabularies by multi-view acoustic representations reported in Chapter 3, necessitate a multi-layer model with reusable units as intermediate layers to bridge the acoustic observations (bottom layer) and the grounded vocabulary (top layer). However, these intermediate units are not known explicitly to the learner. Supervisory information can only work at the top layer, while the intermediate layers need to work unsupervisedly. In this chapter, we explore the unsupervised discovery of spoken patterns using NMF.

Unsupervised NMF learning with BoF representations has been successfully applied to miscellaneous research areas. However, BoF representations describe feature relations by frequency distributions in which all features “are put in a bag”, regardless of e.g. the order they have occurred. Noisy observations and the non-convexity in the NMF optimization can yield poor local optima or generate meaningless patterns. In this chapter, we thus propose to impose additional constraints, like temporal proximity, to constrain unsupervised NMF. Learning is unsupervised and grounding information will only be used for evaluating the patterns discovered. The construction of an overall layered model is future work.

4.1 Unsupervised Pattern Discovery Using NMF

Based on the matrix representation \mathbf{V} of the training data, unsupervised NMF in Eq.(4.1) is applied to discover repeated patterns by the low-rank approximation of \mathbf{V} in Eq.(4.1)

$$\min_{\mathbf{W}, \mathbf{H}} \mathcal{F}_0(\mathbf{V} || \mathbf{W}, \mathbf{H}) \quad (4.1)$$

where \mathcal{F}_0 is the cost function (Kullback-Leibler divergence in our task, or KLD in short), \mathbf{W} is the pattern matrix with size $M \times R$, \mathbf{H} with size $R \times N$ is the weight matrix which are the coefficients of the obtained vocabulary patterns in the training utterances, M, N are the numbers of the features and utterances respectively and $R (\ll M, N)$ is the number of patterns or the reduced dimension of the original data matrix. Interesting properties follow from the *non-negativity* of all entries of \mathbf{V} , \mathbf{W} and \mathbf{H} . With this property, the recurring *non-negative patterns* often have a solid physical meaning: each utterance, a column of \mathbf{V} , is modeled as an additive combination of the columns of \mathbf{W} which are the patterns.

As is illustrated in Figure 4.1, given a set of training data, the goal of NMF is to find some basis vectors \mathbf{W}_k (patterns) whose convex combinations (only with positive weights) should *cover* the training data as much as possible. Subsequently, the learned \mathbf{W}_k 's are evaluated on the test data. They are correct and useful patterns if they are also able to *cover* the testing data.

However, in the BoF representation, a sequence is *flattened* to a vector, which disregards possible dependencies or relations between features which reflect the structure of the data. As illustrated in Figure 4.2, two different sequences result in the same BoF representation.

In real-word problems, feature (textual word) orders are lost in the topic models of document retrieval [119]. Different features (visual words) could represent similar visual contents in images [63]. Audio signals can be represented by linear combinations of features (spectral vectors) in a learned dictionary, but the temporal structure of the features are disregarded [80]. The speech representation in [113] is a bag of features which are short-term acoustic co-occurrences, so the long-term temporal order would be lost with a BoF representation of an utterance.

To address the structural information in the data, a lot of techniques have been proposed that regularize the original NMF model by imposing relations between features, patterns or utterances (the columns of the data matrix \mathbf{V}). The differences between these models and the original NMF are the additional constraint terms as is discussed below. Let α , β and λ denote the regularization parameters to weight the constraint terms. Statistical priors are

used to constrain the solutions, e.g. the sparseness constraints: $\alpha \sum_{i,r} W_{i,r} + \beta \sum_{r,n} H_{r,n}$ in [49] or a multivariate Laplace distribution $\mathcal{L}(\mathbf{H}|\lambda, \mathbf{J})$ with a full covariance matrix \mathbf{J} in [74] to model the dependencies of the entries of \mathbf{H} . Slow changes of the activations of patterns between temporally close utterances, $\lambda \sum_{n=2}^N (H_{r,n} - H_{r,n+1})^2$, are assumed in [117]. Note that this constraint only works for consecutive utterances, e.g. short successive utterances cut by a

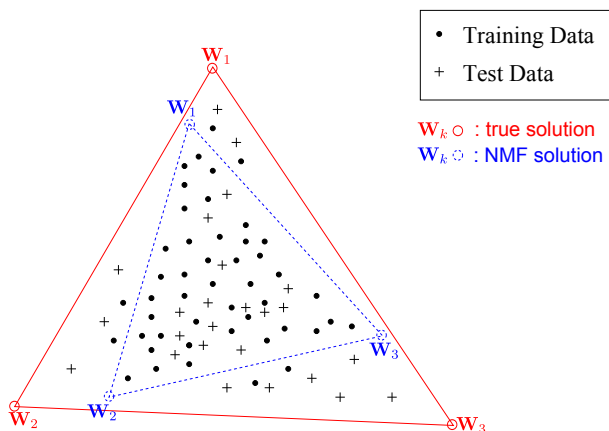


Figure 4.1: The geometric explanation of NMF. Suppose the data is generated by non-negative combinations of “parts”, the “parts” are thus the true solutions as basis vectors. Unsupervised NMF tries to find basis vectors to cover the training data as much as possible. A good solution should also cover the testing data well.

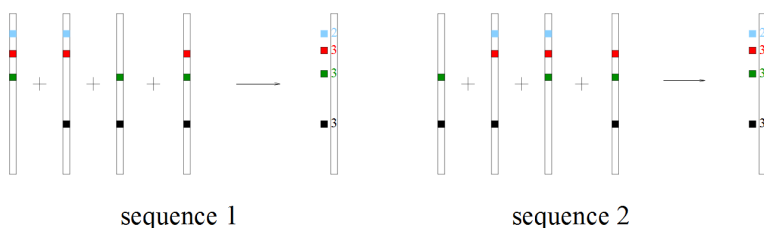


Figure 4.2: The motivation to exploit the adjacency of features. Since the occurring order of the vectors disappears when constructing the BoF, the two different sequences will have the same BoF representation.

sliding window from a long audio file. A constraint derived from the gradients of the low-rank reconstruction \mathbf{WH} , i.e. $\lambda|\nabla\mathbf{WH}|^2$, is used to preserve local topological structures in [128]. Mutual similarity or dissimilarity among the learned patterns is deployed to obtain sparse coding, such as the symmetric Kullback-Leiber divergences between any two columns of \mathbf{W} in [96] and a quadratic function of the cosine distances between any two columns of \mathbf{W} in [110]. Spatially localized and parts-based representations are studied by considering orthogonality constraints of \mathbf{W} and \mathbf{H} in [66] which resulted in the constraint: $\alpha\sum_{i,r}W_{i,r} - \beta\sum_{r,n}H_{r,n}$. Adjacency graphs of data samples are used to preserve the closeness of the samples in the projected low-rank space spanned by the patterns in [15, 36, 122, 120, 40], which in general have a form like $\lambda\text{Tr}(\mathbf{HLH}^T)$ where Tr is the trace of a matrix and \mathbf{L} is the graph Laplacian and will be given in details in section 4.2.1.

Among these techniques, the graph description of the adjacency of features seems to be suitable to provide information on the temporal closeness of code words in the task of spoken pattern discovery using NMF. High-order contextual dependencies besides co-occurrences, like triplets of features, were expected to reflect the long-term temporal relations. However, the experiments in Chapter 3 show that the NMF learning with triplets suffers from heavy computational loads. In this chapter, we explore a different method to express stronger temporal relations: graph adjacency. Additionally, this formulation provides a generic framework to express feature relations other than temporal.

4.2 Graph Regularized NMF

In this section, we will formulate the graph regularized NMF problem and derive an efficient algorithm to solve it [105].

4.2.1 Preliminaries of Graph Theory

A graph is composed of *vertices* and *edges*. The edges reflect the adjacency between vertices: if two vertices are adjacent, the weight of the edge between them is positive; otherwise, the weight is zero. Hence, the graph can be depicted uniquely by a matrix \mathbf{U} whose rows and columns refer to the vertices and the element $U_{i,j}$ is the weight from vertex i to vertex j . $U_{i,j}$ is a non-negative value. The larger it is, the more similar the vertex i and the vertex j are. A graph can be directed or undirected. In the directed graphs, $U_{i,j}$ is usually not equal to $U_{j,i}$ that is $j \rightarrow i$ is different from $i \rightarrow j$. In the undirected graphs, $U_{i,j} = U_{j,i}$ always holds, i.e. \mathbf{U} is symmetric. Normally self loops are removed to only

consider the relations between different vertices. So the diagonal elements of \mathbf{U} are set to zeros. In this work, we use a symmetric graph adjacency matrix.

The degree of a vertex reflects how strongly it is connected to other vertices and it is computed by accumulating the adjacencies of the vertex. A diagonal matrix \mathbf{D} is used to store the degrees of the vertices with elements $D_{i,i} = \sum_j U_{i,j}$. Finally, a variable called *graph Laplacian* is created to summarize the adjacency matrix and the degree matrix: $\mathbf{L} = \mathbf{D} - \mathbf{U}$.

4.2.2 Graph Regularization

The role of graph regularization is to ensure the adjacent features have nearly the same activations in a pattern \mathbf{W}_k (the k -th column of \mathbf{W}). Suppose we obtain the symmetric adjacency matrix \mathbf{U} of the features from some side information and represent them as the undirected graph Laplacian \mathbf{L} . Define the function of the graph constraint of column k as in Equation (4.2) [18].

$$\mathbf{W}_k^T \mathbf{L} \mathbf{W}_k = \sum_{u,v} (W_{u,k} - W_{v,k})^2 U_{u,v} \tag{4.2}$$

where u, v are the indices of features.

The objective function for the graph regularized NMF model is given in Eq.(4.3). \mathcal{F}_0 is the Kullback-Leibler divergence, a common objective function for NMF [64], especially in the situation with high dimensional count data and Poisson noise [94][31].

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{H}} \quad & \mathcal{F}_0(\mathbf{V} || \mathbf{W}, \mathbf{H}) + \lambda \mathcal{F}_1(\mathbf{W}) \\ \text{s.t.} \quad & \mathbf{W} \geq 0, \mathbf{H} \geq 0, \sum_i W_{i,k} = 1 \end{aligned} \tag{4.3}$$

where $\mathcal{F}_1(\mathbf{W}) = \text{Tr}(\mathbf{W}^T \mathbf{L} \mathbf{W}) = \sum_{i,j,k} (W_{i,k} - W_{j,k})^2 U_{i,j}$ is the graph regularization cost.

The work in [15] also proposes models for graph regularization on NMFs. One is using a *graph constraint with Euclidean distance* to regularize NMF based on *Frobenius norm* as is shown below.

$$\min_{\mathbf{W}, \mathbf{H}} \frac{1}{2} ||\mathbf{V} - \mathbf{W}\mathbf{H}||_F^2 + \lambda \text{Tr}(\mathbf{H}\mathbf{L}\mathbf{H}^T) \tag{4.4}$$

The other is using a *graph constraint with symmetric Kullback-Leibler divergence* to regularize NMF based on *Kullback-Leiber divergence* as is shown below.

$$\min_{\mathbf{W}, \mathbf{H}} \mathcal{F}_0(\mathbf{V} || \mathbf{W}, \mathbf{H}) + \lambda \sum_k \sum_{n,n'} \left(H_{k,n} \log \frac{H_{k,n}}{H_{k,n'}} + H_{k,n'} \log \frac{H_{k,n'}}{H_{k,n}} \right) U_{n,n'} \tag{4.5}$$

Within a transposition to convert the graph regularization from the space spanned by the rows of \mathbf{W} (in $\mathbf{V} \approx \mathbf{WH}$) to the space spanned by columns of \mathbf{H} (in $\mathbf{V}^T \approx \mathbf{H}^T \mathbf{W}^T$), our model in Eq.(4.3) is similar to Eq.(4.5). However, our work differs additionally in the following three respects:

- A different expression for the regularization cost.
It is easy to see that Eculidean distance in $\mathcal{F}_1(\mathbf{W})$ is well-defined for zero or tiny entries in \mathbf{W} , but the symmetric KLD is not. Numerically zero entries are not allowed in the symmetric KLD case. A tiny value has to be added as an ad hoc solution to avoid zeros. However, a small change for the tiny value $W_{i,k}$ would bring a large variation in the graph regularization term containing $W_{i,k}$, like $W_{i',k} \log(W_{i',k}/W_{i,k})$ when $W_{i',k}$ is not zero. Hence the scale of the regularization term is difficult to bound or to weight properly. As a practical solution in [15], Taylor expansions were made to replace the log terms under the condition that $W_{i,k} \approx W_{i',k}$ when $U_{i,i'} > 0$. But the condition only holds when the solution of \mathbf{W} is already close to some solution respecting the graph adjacency. If so, one could wonder what the added value of such regularization is.
- The normalization of the solutions.
This claim is relevant to handle the problem of scaling ambiguity or trivial solutions. The objective function of NMF is a metric which compares \mathbf{WH} against \mathbf{V} . Any scaling of the columns of \mathbf{W} by a diagonal matrix \mathbf{S} , \mathbf{WS} , can be compensated by the scaling of \mathbf{H} , $\mathbf{S}^{-1}\mathbf{H}$, such that \mathbf{WH} remains unchanged in Eq. (4.3) [64]. So such a scaling can be used to minimize $\mathcal{F}_1(\mathbf{W})$ (i.e. with tiny \mathbf{S}) while \mathcal{F}_0 remains unchanged, which implies that \mathcal{F}_1 could vanish and is no longer effective to express the structure of graph adjacency. The problem is referred to as the *scale transfer problem* in [40]. An ad-hoc solution is to impose an additional normalization of \mathbf{W} in each NMF iteration. We will explicitly take the constraint into account.
- Different iterative update formula, leading to an efficient algorithm for large scale computation.
For the algorithm used to minimize Eq. (4.3), we have to consider at least three properties: non-negativity, convergence and efficiency. The gradient descent algorithm with a modified step-size $\eta = \frac{\mathbf{W}^t}{\nabla^+ \mathbf{W}^t}$ to convert the additive update $\mathbf{W} = \mathbf{W}^t - \eta(\nabla^+ \mathbf{W}^t - \nabla^- \mathbf{W}^t)$ into a multiplicative one $\mathbf{W} = \frac{\nabla^- \mathbf{W}^t}{\nabla^+ \mathbf{W}^t} \mathbf{W}^t$, may fail. Here $\nabla^+ \mathbf{W}$ and $\nabla^- \mathbf{W}$ are the positive and negative parts of the derivative of the objective function w.r.t. \mathbf{W} . Non-negativity is always preserved, but convergence is not, since with the regularization term, the update may not coincide with the fixed point of some auxiliary function as in Lee and Seung’s original derivation [65].

As an alternative method, one can select a proper step size adaptively to ensure both non-negativity and non-increase of the objective function values [117, 128, 41]. Some techniques use auxiliary functions to derive proper updating algorithms, most of which originate from [65]. There are an infinite number of auxiliary functions for a given objective function, but it is not straightforward to come up with an adequate one. The suitability of an auxiliary function depends on its convexity and its tractability to compute a fixed point.

4.2.3 Algorithms

In this section we derive the non-negative updating algorithm of \mathbf{W} to solve problem (4.3), as well as ensuring the convergence and the ℓ_1 normalization of the columns of \mathbf{W} . The update algorithm for the unregularized term \mathbf{H} is the same as in [65].

Construction of the auxiliary function

Definition 4.1: $\mathcal{A}(\mathbf{W}, \mathbf{W}^t)$ is an auxiliary function for $\mathcal{F}(\mathbf{W})$ if

$$\mathcal{A}(\mathbf{W}, \mathbf{W}^t) \geq \mathcal{F}(\mathbf{W}), \quad \mathcal{A}(\mathbf{W}^t, \mathbf{W}^t) = \mathcal{F}(\mathbf{W}^t) \tag{4.6}$$

where $\mathcal{F}(\mathbf{W}) = \mathcal{F}_0(\mathbf{V}||\mathbf{W}, \mathbf{H}) + \lambda\mathcal{F}_1(\mathbf{W})$. If $\mathcal{A}(\mathbf{W}, \mathbf{W}^t)$ is an auxiliary function for $\mathcal{F}(\mathbf{W})$, then $\mathcal{F}(\mathbf{W})$ is not increasing under the update $\mathbf{W}^{t+1} = \operatorname{argmin}_{\mathbf{W}} \mathcal{A}(\mathbf{W}, \mathbf{W}^t)$ [65].

We firstly define the objective function w.r.t. $\tilde{\mathbf{W}}$ in Eq. (4.7), where $\tilde{W}_{ik} = W_{ik} / \sum_i W_{ik}$ is the normalized \mathbf{W} .

$$\mathcal{F}(\mathbf{V}||\tilde{\mathbf{W}}\mathbf{H}) = \mathcal{F}_0(\mathbf{V}||\tilde{\mathbf{W}}\mathbf{H}) + \lambda\mathcal{F}_1(\tilde{\mathbf{W}}) \tag{4.7}$$

The auxiliary function of the above objective function $\mathcal{F}(\mathbf{V}||\tilde{\mathbf{W}}\mathbf{H})$ is constructed in Eq. (4.8) ([65][24]).

$$\mathcal{A}(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t) = \mathcal{A}_0(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t) + \lambda\mathcal{A}_1(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t) \tag{4.8}$$

where \mathcal{A}_0 and \mathcal{A}_1 are auxiliary functions of \mathcal{F}_0 and \mathcal{F}_1 respectively.

$$\begin{aligned} \mathcal{A}_0(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t) &= \sum_{i,j} (V_{ij} \log(V_{ij}) - V_{ij}) + \sum_{i,k,j} \tilde{W}_{ik} H_{kj} \\ &\quad - \sum_{i,k,j} \frac{V_{ij} \tilde{W}_{ik}^t H_{kj}}{\sum_l \tilde{W}_{il}^t H_{lj}} (\log \tilde{W}_{ik} H_{kj} - \log \frac{\tilde{W}_{ik}^t H_{kj}}{\sum_l \tilde{W}_{il}^t H_{lj}}) \end{aligned} \tag{4.9}$$

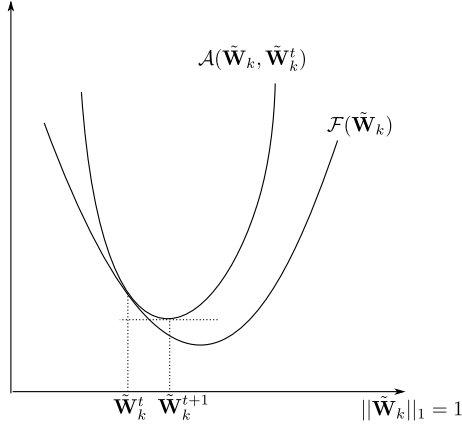


Figure 4.3: The illustration of minimizing $\mathcal{F}(\tilde{\mathbf{W}}_k)$ by using its auxiliary function $\mathcal{A}(\tilde{\mathbf{W}}_k, \tilde{\mathbf{W}}_k^t)$ where $\tilde{\mathbf{W}}_k^t$ is the solution from the previous iteration. Both functions are defined on the hyperplane $\|\tilde{\mathbf{W}}_k\|_1 = 1$. $\mathcal{A}(\tilde{\mathbf{W}}_k, \tilde{\mathbf{W}}_k^t)$ is usually chosen as a convex function whose stationary point $\tilde{\mathbf{W}}_k^{t+1}$ is easy to compute. It is straightforward to see the decreasing of $\mathcal{F}(\tilde{\mathbf{W}}_k)$ from $\tilde{\mathbf{W}}_k^t$ to $\tilde{\mathbf{W}}_k^{t+1}$.

$$\mathcal{A}_1(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t) = \sum_{i,k} D_{ii} \tilde{W}_{ik}^2 - \sum_{i,k,j} U_{ij} \tilde{W}_{ik}^t \tilde{W}_{jk}^t (1 + \log \frac{\tilde{W}_{ik} \tilde{W}_{jk}}{\tilde{W}_{ik}^t \tilde{W}_{jk}^t}) \quad (4.10)$$

It is obvious that $\mathcal{A}(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t) \geq \mathcal{F}(\mathbf{V} \|\tilde{\mathbf{W}}\mathbf{H})$ and $\mathcal{F}(\tilde{\mathbf{W}}^t, \mathbf{W}^t) = \mathcal{F}(\mathbf{V} \|\tilde{\mathbf{W}}^t\mathbf{H})$ from [65] and [24]. Thus $\mathcal{A}(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t)$ is an auxiliary function of $\mathcal{F}(\mathbf{V} \|\tilde{\mathbf{W}}\mathbf{H})$ w.r.t. the variable $\tilde{\mathbf{W}}$. The convexity of $\mathcal{A}(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t)$ w.r.t. $\tilde{\mathbf{W}}$ is also guaranteed from the convexity of $-\log(\cdot)$ and $(\cdot)^2$ w.r.t. $\tilde{W}_{i,k}$.

The above definitions can be explained as follows: any column of \mathbf{W} , say \mathbf{W}_k , is first projected onto the hyperplane $\|\mathbf{W}_k\|_1 = 1$ and then its objective function and auxiliary function are computed. We now extend $\mathcal{F}(\mathbf{V} \|\tilde{\mathbf{W}}\mathbf{H})$ in Eq. (4.7) and $\mathcal{A}(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t)$ in Eq. (4.8) (defined on the normalized variables $\tilde{\mathbf{W}}$'s) to $\tilde{\mathcal{F}}(\mathbf{V} \|\mathbf{W}\mathbf{H}) = \mathcal{F}(\mathbf{V} \|\tilde{\mathbf{W}}\mathbf{H})$ and $\tilde{\mathcal{A}}(\mathbf{W}, \tilde{\mathbf{W}}^t) = \mathcal{A}(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t)$ (defined on all \mathbf{W} 's). $\tilde{\mathcal{A}}(\mathbf{W}, \tilde{\mathbf{W}}^t)$ is still the auxiliary function of $\tilde{\mathcal{F}}(\mathbf{V} \|\mathbf{W}\mathbf{H})$. The convexity of $\tilde{\mathcal{A}}(\mathbf{W}, \tilde{\mathbf{W}}^t)$ is only guaranteed on the ℓ_1 contours $\|\mathbf{W}_k\|_1 = 1, \forall k$. $\tilde{\mathcal{A}}(\mathbf{W}, \tilde{\mathbf{W}}^t)$ will thus be minimized over \mathbf{W} along the ℓ_1 contours as is illustrated in Figure 4.3, which is achieved by the Jacobian matrix between $\tilde{\mathbf{W}}$ and \mathbf{W} in the next section.

Computing the fixed point

The gradient of $\tilde{\mathcal{A}}(\mathbf{W}, \mathbf{W}^t)$ with respect to \mathbf{W} is given in Eq. (4.11).

$$\frac{\partial \tilde{\mathcal{A}}(\mathbf{W}, \mathbf{W}^t)}{\partial W_{ik}} = \sum_s \frac{\partial \mathcal{A}(\tilde{\mathbf{W}}, \tilde{\mathbf{W}}^t)}{\partial \tilde{W}_{sk}} \frac{\partial \tilde{W}_{sk}}{\partial W_{ik}} \quad (4.11)$$

where $\frac{\partial \tilde{W}_{ik}}{\partial W_{sk}} = \frac{1}{\sum_t W_{tk}} (\delta_{is} - \tilde{W}_{sk})$, and δ_{is} is the Kronecker delta. Setting $\frac{\partial \tilde{\mathcal{A}}(\mathbf{W}, \mathbf{W}^t)}{\partial W_{ik}} = 0$ yields Eq. (4.12),

$$\left(-\frac{B_{ik}}{\tilde{W}_{ik}} + \sum_s B_{sk} + C_k (1 - \sum_s \tilde{W}_{sk}) + 2\lambda D_{ii} \tilde{W}_{ik} - 2\lambda \sum_s D_{ss} \tilde{W}_{sk}^2 - 2\lambda \frac{E_{ik}}{\tilde{W}_{ik}} + 2\lambda \sum_s E_{sk} \right) \frac{1}{\sum_s \tilde{W}_{sk}} = 0 \quad (4.12)$$

where $B_{ik} = \sum_j V_{ij} \frac{W_{ik}^t H_{kj}}{\sum_b W_{ib}^t H_{bj}}$, $C_k = \sum_j H_{kj}$ and $E_{ik} = (UW^t)_{ik} W_{ik}^t$. \mathbf{D} and \mathbf{U} are the degree matrix and the adjacency matrix of the graph respectively. Notice that with the constraint $\sum_s W_{sk} = 1$ and $\sum_s \tilde{W}_{sk} = 1$ (to make sure $\tilde{\mathcal{A}}$ is convex), all the terms including W_{sk} and the term $C_k (1 - \sum_s \tilde{W}_{sk}) \tilde{W}_{ik}$ are removed. For $\tilde{W}_{ik} > 0$, the equations are expressed in terms of the normalized $\tilde{\mathbf{W}}$ in Eq.(4.13).

$$2\lambda \left(\sum_s D_{ss} \tilde{W}_{sk}^2 \tilde{W}_{ik} + E_{ik} - \sum_s E_{sk} \tilde{W}_{ik} \right) + B_{ik} - \sum_s B_{sk} \tilde{W}_{ik} - 2\lambda D_{ii} \tilde{W}_{ik}^2 = 0 \quad (4.13)$$

By summing Eq. (4.13) over i , for any k , we get,

$$\left(\sum_s B_{sk} - 2\lambda \sum_s D_{ss} \tilde{W}_{sk}^2 + 2\lambda \sum_s E_{sk} \right) \left(\sum_i \tilde{W}_{ik} - 1 \right) = 0 \quad (4.14)$$

So the solution of Eq. (4.13) ensures the ℓ_1 normalization, $\sum_i \tilde{W}_{ik} = 1$, for almost all λ 's.

Iterative solution of Eq. (4.13)

An iterative algorithm is constructed in Eq.(4.15) to solve Eq. (4.13). The limit $\lim_{n \rightarrow \infty} \tilde{W}_{ik}^{(n)}$ would be the solution of Eq. (4.13) if the convergence of the iteration w.r.t. n holds, where $\tilde{W}_{ik}^{(0)} = \tilde{W}_{ik}^t$.

$$a_i(\tilde{W}_{ik}^{(n+1)})^2 + b_k^{(n)}\tilde{W}_{ik}^{(n+1)} - c_{ik} = 0 \quad (4.15)$$

Where $a_i = 2\lambda D_{ii}$, $b_k^{(n)} = \sum_s B_{sk} - 2\lambda \sum_s D_{ss}(\tilde{W}_{sk}^{(n)})^2 + 2\lambda \sum_s E_{sk}$, $c_{ik} = B_{ik} + 2\lambda E_{ik} \geq 0$.

We partition the row index set I of \mathbf{W} in two sets I_1 and I_2 where $a_i > 0$ and $a_i = 0$ respectively. For $i \in I_1$, the positive root of the quadratic equation is the updated $\tilde{W}_{ik}^{(n+1)}$.

$$\tilde{W}_{ik}^{(n+1)} = \frac{-b_k^{(n)} + \sqrt{(b_k^{(n)})^2 + 4a_i c_{ik}}}{2a_i} \quad (4.16)$$

For $i \in I_2$ (i.e. $D_{ii}=0$, which correspond to isolated vertices in the graph), Eq. (4.15) degrades to the linear Eq. (4.17) with solution $\tilde{W}_{ik}^{(n+1)} = \frac{c_{ik}}{b_k^{(n)}}$.

$$b_k^{(n)}\tilde{W}_{ik}^{(n+1)} - c_{ik} = 0 \quad (4.17)$$

When $b_k^{(n)} \leq 0$, the fixed point is not feasible as $W_{ik}^{(n+1)}$ would be non-negative. Thus the local optimum is at the boundary $W_{ik}^{(n+1)} = 0$ which means the isolated vertex is not grouped into any pattern or cluster in this case. This corresponds to the situation with a very large λ , i.e. the graph regularization dominates the optimization process, so $W_{ik}^{(n+1)} = 0$ is reasonable for isolated vertices. An additional normalization (as will be shown in Case 4 of the proof in the Appendix A) is required to shift the hyperplane from $\sum_{i \in I_1} W_{ik}^{(n+1)} = \text{Constant} > 1$ (as the primary $\sum_{i \in I_2} W_{ik}^{(n+1)} < 0$ and $\sum_{i \in I} W_{ik}^{(n+1)} = 1$) to $\sum_{i \in I_1} W_{ik}^{(n+1)} = \sum_{i \in I} W_{ik}^{(n+1)} = 1$.

In the case of $\lambda=0$, $b_k^{(n)} = b_k^{(0)} = \sum_s B_{sk}$, the solution is Eq. (4.18) which is exactly the same as in [65], where one first computes $W_{ik} = B_{ik}$ and then normalizes $\tilde{W}_{ik} = \frac{W_{ik}}{\sum_s W_{sk}}$.

$$\tilde{W}_{ik} = \frac{B_{ik}}{\sum_s B_{sk}} \quad (4.18)$$

To summarize, the algorithm is listed in Table 4.1, while the proof of the iterative solution of Eq.(4.13) is given in the Appendix A). The algorithm is called *L1GNMF* to distinguish from the algorithm in [15] which will be referred to as *GNMF* in the sequel. The operations involved are element-wise addition, multiplication \odot , division \oslash and square root which are efficient to compute. In our numerical experiments, at most $N_2=5$ iterations are sufficient to compute the fixed point within each NMF iteration.

```

input :  $\mathbf{V}, R, N_1, N_2, \lambda, \mathbf{D}, \mathbf{U}, \mathbf{W}, \mathbf{H}, t=0$ 
output:  $\mathbf{W}, \mathbf{H}$ 
 $\mathbf{X} = 2 * \lambda * \text{diag}(\mathbf{D})^T * \mathbf{1}_{1 \times R}$ ,
 $I_1 = \{i | X_{i,1} > 0\}, I_2 = \{i | X_{i,1} == 0\}$ ;
while  $t < N_1$  do
   $\mathbf{F} = \mathbf{V} \oslash (\mathbf{W} * \mathbf{H})$ 
   $\mathbf{H} \leftarrow (\mathbf{W}^T * \mathbf{F}) \odot \mathbf{H}$ 
   $\mathbf{F} = \mathbf{V} \oslash (\mathbf{W} * \mathbf{H})$ 
   $\mathbf{Y} = ((\mathbf{F} * \mathbf{H}^T) + 2 * \lambda * (\mathbf{U} * \mathbf{W})) \odot \mathbf{W}, n=0$ 
  while  $n < N_2$  do
     $\mathbf{Z} = \sum_s \mathbf{Y}_{s,:} - 2 * \lambda * \text{diag}(\mathbf{W}^T * \mathbf{D} * \mathbf{W})$ 
     $\mathbf{Z} \leftarrow \mathbf{1}_{M \times 1} * \mathbf{Z}$ 
     $\mathbf{W}_{I_1,:} \leftarrow \sqrt{\mathbf{Z}_{I_1,:}^2 + 4 * \mathbf{X}_{I_1,:}} \odot \mathbf{Y}_{I_1,:} - \mathbf{Z}_{I_1,:}$ 
     $\mathbf{W}_{I_1,:} \leftarrow \mathbf{W}_{I_1,:} \oslash \mathbf{X}_{I_1,:}$ 
     $\mathbf{W}_{I_2,:} \leftarrow \max\{\mathbf{Y}_{I_2,:} \oslash \mathbf{Z}_{I_2,:}, 0\}$ 
     $J = \{k | \mathbf{Z}_{1,k} \leq 0\}$ 
    if  $J$  is not empty then
       $\mathbf{W}_{:,J} \leftarrow \mathbf{W}_{:,J} * (\text{diag}(\sum_i \mathbf{W}_{i,J}))^{-1}$ ;
    end
     $n \leftarrow n + 1$ 
  end
   $t \leftarrow t + 1$ 
end

```

Algorithm 4.1: Updating algorithm of L1GNMF

4.3 Experiments and Results on Spoken Pattern Discovery

In this section, we evaluate the algorithms of NMF, GNMF and L1GNMF on the speech database, TIDIGITS. For the implementation of GNMF, the software from [14] is utilized.

A subset of the TIDIGITS database is used to evaluate the proposed algorithm. Without using the utterances with single digits, we selected 6026 training utterances each of which contains at least two digits. The 1001 testing utterances are used for evaluation. The goal is to discover a set of spoken patterns (words) unsupervisedly.

A Gaussian mixture model with M components was first trained unsupervisedly from the MFCC+ Δ + $\Delta\Delta$ vectors of the training data. Subsequently, the utterances are represented as a bag of Gaussians (unigram) and bag-of-Gaussian co-occurrences (bigram) with the same method as presented in section 3.3 of Chapter 3.

4.3.1 Construction of Adjacency Matrices

The representation of the unigram bag-of-features loses sequential information in an utterance here by forming the sum of posterior probabilities. The only sequential information that remains is the (very local) dynamic trend described in the Δ and $\Delta\Delta$ features in the unigram model. In the bigram model, the directed co-occurrences model the sequential property of speech to a larger extent. The patterns (columns of \mathbf{W}) discovered by the NMF also use BoF representations as input data. There is nothing that stops the NMF from generating BoF patterns that contain feature activations that occurred disjoint in time in the training data, which contradicts our expectation that patterns correspond to spoken words. Such solutions could occur due to local extrema, noisy data or non-uniqueness issues. A graph adjacency matrix of Gaussians and Gaussian co-occurrences is thus used to regularize the NMF unigram model and bigram model respectively to yield solutions which group adjacent features into one pattern.

Adjacency matrix of the unigram model

The temporal adjacency of Gaussians is obtained from the adjacency of acoustic frames as illustrated in Figure 4.4(a). The procedure to make an adjacency matrix \mathbf{U} from the training set is as follows.

- (i) Given an utterance and the frame's posterior probabilities of the Gaussians, \mathbf{X}_t , the top N_{best} Gaussians with highest probabilities are selected for each frame. In Figure 4.4(a), the top $N_{best}=3$ Gaussians (black dots) are retained for each frame (rectangular bars representing the list of all Gaussian posteriors).

- (ii) Frames \mathbf{X}_s and \mathbf{X}_t are said to be p -near if $|s - t| = p$. Gaussians on the top $Nbest$ lists of two p -near frames are connected. Some connections (curves and lines) are shown in Figure 4.4(a) with $p=0,1,2$.
- (iii) For a fixed p , its adjacency matrix $\mathbf{U}^{(p)}$ is obtained by counting the Gaussian connections for all the p -near frames in the training utterances ($|s - t| = p$). Then the diagonal elements of $\mathbf{U}^{(p)}$ are set to zero to remove self loops. $\mathbf{U}^{(p)}$ is made symmetric by $\mathbf{U}^{(p)} = \mathbf{U}^{(p)} + (\mathbf{U}^{(p)})^T$ to change directed edges to undirected ones.
- (iv) The adjacency matrix is obtained by summing all the $\mathbf{U}^{(p)}$: $\mathbf{U} = \sum_{p=0}^P \mathbf{U}^{(p)}$ for some chosen P .

Adjacency matrix of the bigram model

Now we present how to build the undirected graph of the Gaussian co-occurrences or Gaussian pairs in Figure 4.4(b). Firstly, we define the *adjacency of frame pairs*: $(\mathbf{X}_t, \mathbf{X}_{t+2})$ and $(\mathbf{X}_{t+1}, \mathbf{X}_{t+3})$. We suppose that the temporal property of a frame-pair is given by its head frame (\mathbf{X}_t and \mathbf{X}_{t+1} respectively), i.e. whether two frame pairs are adjacent or not is determined by their first frame. If the two heads are spaced less than P frames, we say the two frame-pairs are adjacent. *Two Gaussian pairs are adjacent* if they appear in the $Nbest$ ($\leq K^2$) list of Gaussian pairs of two adjacent frame pairs. The total adjacency frequency of two Gaussian pairs is the accumulation over all the adjacent frame-pairs of the utterances of the training set.

The obtained adjacency matrix for the co-occurrences defined by lags τ_i and τ_j is stored in matrix $\mathbf{U}^{(ij)}$. Hence, for three lags, the final adjacency matrix \mathbf{U} would have 9 blocks as is given in Eq. (4.19).

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}^{(11)} & \mathbf{U}^{(12)} & \mathbf{U}^{(13)} \\ \mathbf{U}^{(21)} & \mathbf{U}^{(22)} & \mathbf{U}^{(23)} \\ \mathbf{U}^{(31)} & \mathbf{U}^{(32)} & \mathbf{U}^{(33)} \end{bmatrix} \quad (4.19)$$

For both the unigram and the bigram model, since the adjacency matrix is made from data which contains a lot of uncertainty or variation of speech, not all of the connected features in the obtained adjacency matrix are really neighbors in the expected ideal patterns. In order to remove spurious adjacency relations and to make the model robust, a threshold Γ is applied to make \mathbf{U} sparse and logical:

$$\mathbf{U} \leftarrow \mathbf{U} > \Gamma. \quad (4.20)$$

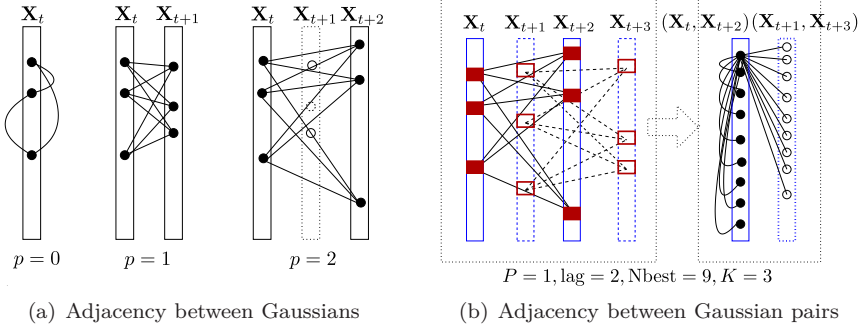


Figure 4.4: Adjacency of features. In both figures, the black dots and circles are the features on which the adjacency is defined. \mathbf{X}_t is the posterior probabilities of frame \mathbf{O}_t on the Gaussians. p and P are parameters to define adjacency. Nbest and K are parameters to control sparsity. lag is the parameter to define Gaussians co-occurrences.

The diagonal elements of the degree matrix \mathbf{D} are the column sums of \mathbf{U} . The details of L1GNMF on bigram models can be found in [103].

4.3.2 Evaluation Method

With the data matrix \mathbf{V} , the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{U}$, \mathbf{W} and \mathbf{H} can be estimated by using the algorithm in Table 4.1. In the stage of evaluation on the testing set, the n -th testing utterance \mathbf{V}'_n is explained as a linear combination of the obtained patterns \mathbf{W} : $\mathbf{V}'_n \approx \mathbf{W}\mathbf{H}'_n$. For all the utterances in the testing set, the coefficient matrix \mathbf{H}' is estimated by solving the convex problem in Eq.(4.21).

$$\min_{\mathbf{H}'} \mathcal{F}_0(\mathbf{V}' || \mathbf{W}\mathbf{H}') \tag{4.21}$$

where \mathbf{V}' is the data matrix of the testing set.

It is not a trivial problem to compute an evaluation metric from \mathbf{H}' because the obtained patterns in \mathbf{W} are not labeled in the unsupervised training process. We therefore estimate an analysis mapping matrix \mathbf{Q} that will give an interpretation to the obtained patterns. Assume that a subset of the training data is labeled in the form of a ground truth matrix \mathbf{G} : if the utterance d_n is known to contain the l -th word k times, $G_{l,n} = k$; otherwise, $G_{l,n} = 0$, where $l = 1, 2, \dots, L$ and L is the number of ground truth words. By solving the convex optimization problem in Eq.(4.22), the mapping matrix \mathbf{Q} between the

obtained patterns and the ground truth is obtained.

$$\min_{\mathbf{Q}} \mathcal{F}_0(\mathbf{G} || \mathbf{QH}) \tag{4.22}$$

where \mathbf{H} are the pattern activations of the subset of the training data and obtained from the training stage. Note that \mathbf{W} is still obtained without supervision and that \mathbf{G} is only used to estimate the analysis mapping \mathbf{Q} such that the results can be interpreted easily.

The estimate of the activation of the words in the testing utterance $\hat{\mathbf{G}}'$ is subsequently computed by Eq.(4.23) using the mapping matrix \mathbf{Q} .

$$\hat{\mathbf{G}}' = \mathbf{QH}' \tag{4.23}$$

The performance of the model is evaluated by comparing $\hat{\mathbf{G}}'$ and \mathbf{G}' to obtain the unordered word error rates (UWER) as explained in section 2.3.2 of Chapter 2.

4.3.3 Parameter Setting and Results

The analysis window for each frame was 25ms and the shift was 10ms. For the unigram model, the number of Gaussians was $M=500$. Only the top $N_{best}=4$ Gaussians were retained when constructing the adjacency matrix. The neighborhood parameter was $P=3$ frames (30ms) which means we only consider adjacency of Gaussians that do not exceed the length of a word. A threshold needs to be estimated for a binary adjacency matrix in Eq.(4.20). Suppose different patterns have different features (Gaussians here, Gaussian co-occurrence in the bigram model). The minimum frequency of adjacency of a feature approximates the average frequency of the patterns in the training dataset. Treating digits from male and female speakers as different patterns, the average frequency of a pattern is $6026 (\# \text{ utterances}) / 11 (\# \text{ digits}) / 2 (\# \text{ genders}) \approx 275$. For this unigram model we took $\Gamma=200$ in our experiments. The number of main iterations was $N_1=500$ in Algorithm 4.1. The results of the unigram model are listed in Table 4.1. Five attempts each with random initializations are tested.

The regularization parameter λ in GNMF has been tuned for $R = 30$ and $\lambda = 1$ seems to be the best choice as is shown in Figure 4.7. The value $\lambda = 1$ is copied to other choices of R . We did not tune λ for every situation because the GNMF algorithm is slow.

In the bigram model, the number of Gaussians was $M=200$. For each frame, at most $K = 3$ Gaussians were retained. The lags of the frame pairs were

Table 4.1: UWER of NMF, GNMF and L1GNMF with the unigram model (500 Gaussians) on TIDIGITS.

	$R = 20$	$R = 25$	$R = 30$	$R = 40$	$R = 50$
NMF	13.69±2.96	7.61±2.27	4.05±1.69	3.13±1.07	2.83±0.30
GNMF $\lambda=1$	13.42±2.51	5.25±2.12	3.28±1.93	3.45±0.98	3.04±0.50
L1GNMF $\lambda=100$	6.62±1.99	2.73±0.51	2.64±0.12	2.63±0.08	2.68±0.15

Table 4.2: UWER of NMF, GNMF and L1GNMF with the bigram model (200 Gaussians) on TIDIGITS.

	$R = 20$	$R = 25$	$R = 30$	$R = 50$
NMF	13.93±0.99	8.71±0.95	5.91±1.98	2.88±0.10
L1GNMF, $\lambda=100$	10.43±1.06	4.12±1.45	2.39±0.30	2.20±0.08

20, 50 and 90 ms, which represent contextual dependence with different time scales. Hence the total number of features is 3×200^2 . For the adjacency model, the top $N_{best}=4$ Gaussian-pairs were retained for each frame-pair, and the neighborhood parameter is $P=3$ frames (30ms). A reasonable value for the threshold parameter Γ to make the adjacency matrix logical is obtained like in the above. Since overestimating the threshold leads to the standard NMF, in this bigram model we conservatively take a slightly larger threshold $\Gamma=300$. The results of the bigram model are listed in Table 4.2 from 5 random initializations. Due to the high dimensionality and the heavy computational budget in GNMF, only the results from NMF and L1GNMF are reported.

4.3.4 Conclusion

The best UWER from the above unsupervised NMF learning is 2.20%. So L1GNMF gets good results even without supervision about the keywords and using unsupervisedly clustered Gaussians. From Table 4.1 and Table 4.2, it is observed that L1GNMF always outperforms NMF and GNMF by finding more accurate representations for the digits in the database. GNMF fails to improve NMF on the unigram models in Table 4.1. Due to its complexity, GNMF cannot solve the optimization problem in the bigram models for TIDIGITS. The performance of GNMF with λ larger than 1 on TIDIGITS with the unigram models can be found in Figure 4.7 which justifies the choice of $\lambda=1$.

However, the performance with a small λ approximates the performance of NMF according to the continuity of the update algorithm of GNMF with respect to λ in [15]. The abnormal behavior of GNMF deserves further experimental analysis which will be given in section 4.3.5.

By reading the changes of UWERs w.r.t. R of all three algorithms, two interesting trends are discovered. One is that the steepest descent of UWER occurs between $R = 20$ and $R = 25$. Only slow improvement is observed when $R > 25$. It means that a model with $R = 25$ is almost sufficient for modeling the underlying data structure. Since there are 11 digits with male and female voices, the dimension of the data could be 22 which is just between 20 and 25. The other one is that with a large dimension $R = 50$ all three algorithm are doing well. This can be explained by the evaluation method of the unsupervised NMF algorithms in section 4.3.2 where any testing utterance is reconstructed by a linear combination of the discovered patterns. More patterns provide more choices of the reconstruction, which would not harm the approximation in Eq.(4.21). Of course, it is not true if R was so large that over-learning happens.

4.3.5 Comparison of GNMF and L1GNMF

To simplify the discovery task, experiments were made on the speech data using the $M=3628$ Gaussians from the supervised training of HMM in section 3.3. Those Gaussians have a particular structure where the corresponding HMM is constructed such that the Gaussians are not shared among HMM states. So each digit is represented by a subset of Gaussians and the columns of \mathbf{W} have strong relations to the Gaussian weight matrices of the GMMs in Eq.(1.14). We therefore call them *oracle Gaussians*. With the property that Gaussians are tied to some hidden state and the state is tied to a digit, the feature clustering that is imposed in the graph regularization becomes easier than the one in subsection 4.3. The performance of the three algorithms is listed in Table 4.3. With these features, GNMF outperforms NMF significantly for some choices of R . The regularization parameter λ for GNMF and L1GNMF is tuned with $R=30$ and the best values are applied for all R 's.

Though having obtained good results, GNMF still has its intrinsic problems which will be discussed by analyzing and comparing the optimization process of GNMF and L1GNMF with the experiments using oracle Gaussians.

Table 4.3: UWER of NMF, GNMF and L1GNMF with oracle Gaussians on TIDIGITS.

# patterns R	20	25	30	40	50
NMF	13.37±2.02	5.05±1.62	2.25±1.66	1.48±0.16	1.50±0.18
GNMF $\lambda=100$	10.03±1.39	2.43±1.23	1.85±1.02	1.48±0.12	1.40±0.09
L1GNMF $\lambda=1000$	10.75±2.06	4.30±2.02	1.49±0.25	1.43±0.08	1.44±0.15

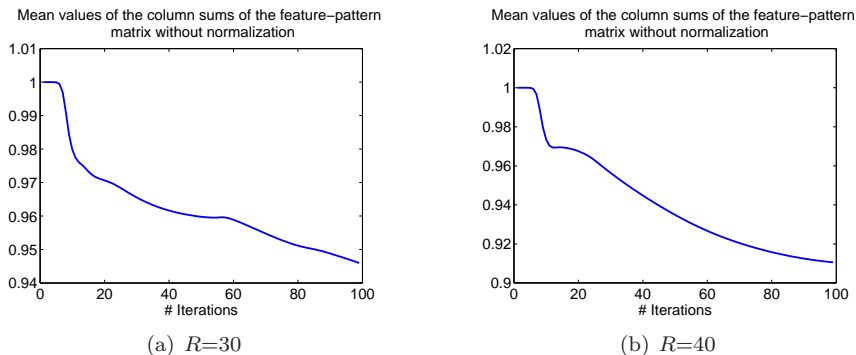


Figure 4.5: The effect of not normalizing the columns of the pattern matrix \mathbf{W} from GNMF with $\lambda=1000$ and oracle Gaussians. The mean values of the column sums of \mathbf{W} at each iteration become smaller and smaller with more iterations, which can produce trivial patterns $\mathbf{W}_k=0$.

ℓ_1 normalization

The consequence of omitting the normalization of the columns of the feature-pattern matrix \mathbf{W} is that the average values of the matrix tend to become smaller and smaller with more iterations as shown by the mean values of the column sums of \mathbf{W} at each iteration in Figure 4.5. This would result in trivial patterns with only tiny or zero activations of features after a number of iterations. So for the GNMF algorithm used to obtain Table 4.1, 4.3, and 4.7, the columns of \mathbf{W} were normalized at every iteration by replacing it with $\mathbf{W}\mathbf{S}$ with $\mathbf{S} = \text{diag}(1/\sum_i W_{i,1}, \dots, 1/\sum_i W_{i,R})$. \mathbf{H} was inversely scaled by $\mathbf{S}^{-1}\mathbf{H}$.

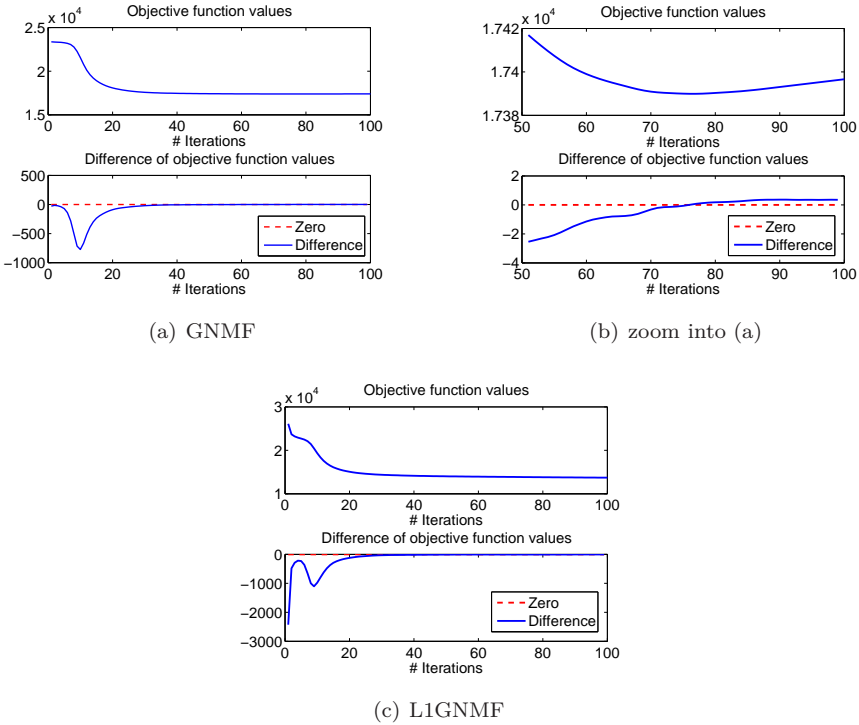


Figure 4.6: The objective function values and the differences of the objective function values of GNMF and L1GNMF with $R=30$, $\lambda=10000$ and oracle Gaussians. With the normalization at every iteration, the monotonous decreasing of the objective function in GNMF is violated. L1GNMF does not have such a problem.

Convergence

Because of the normalization required due to the scaling problem, the monotonous decrease of the objective function is sometimes violated as is shown in Figure 4.6(a) and Figure 4.6(b). The convergence is thus not guaranteed theoretically nor numerically. However, the objective function does decrease monotonously in the L1GNMF as seen in Figure 4.6(c).

Complexity

The number of arithmetic operations required per iteration of NMF, GNMF and L1GNMF are listed in Table 4.4 where the counts of NMF and GNMF are from [15]. The detailed analysis is given in row 2 through row 5. The bottom row summarizes the dominant terms. To make an intuitive comparison, one can take N_2 to be the same as the number of conjugate gradient (CG) iterations q in GNMF [15]. ([15] advises to select $q=20$, while our experience is that N_2 can be even smaller. We use $N_2=5$.) Thus it is intuitive to see that the complexity of L1GNMF is $O((N+p+q)MR)$ which is less than that of GNMF, $O((N+q(p+4))MR)$. The difference lies in the iterations regarding the update of \mathbf{W} . For each column of \mathbf{W} , GNMF has matrix-vector computations with complexity $O(pM)$ at each CG.

Table 4.4: Computation counts for each main iteration of NMF, GNMF and L1GNMF

	NMF	GNMF	L1GNMF
fladd	$4MNR + (M + N)R$	$4MNR + (2N + M)R + q(p + 4)MR$	$4MNR + MRp + MR + N_2(2MR + 2\tilde{p}R + R)$
fmlt	$4MNR + (M + N)R$	$4MNR + (M + N)R + Mp + q(p + 4)MR$	$4MNR + NR + MR + 3MRp + N_2(4MR + 3\tilde{p}R)$
fdiv	$2MN + (M + N)R$	$2MN + NR$	$2MN + N_2MR$
fsqrt	0	0	$N_2\tilde{p}R$
total	$O(MNR)$	$O((N + q(p + 4))MR)$	$O((N + N_2 + p)MR)$

fladd:a floating-point addition

fdiv:a floating-point division

N :the number of utterances

R : the number of patterns

q :the number of iterations in CG of [1]

\tilde{p} :the number of features with positive graph degrees, $\tilde{p} \leq M$

N_2 :the number of iterations to solve the 3rd order equation groups

fmlt:a floating-point multiplication

fsqrt:a floating-point square root

M :the number of features

p :the number of nearest neighbors in [1]

The average computation time required by some of the experiments are also given in Table 4.5.

Parameter sensitivity

There are two key parameters in the model: the regularization parameter λ and the graph adjacency threshold Γ . Figure 4.7 shows the sensitivity of GNMF and L1GNMF w.r.t. these two parameters.

In Figure 4.7(a), Figure 4.7(b) and Figure 4.7(d), it is observed that L1GNMF outperforms NMF for almost all scales of λ . Large improvements always happen at the scale of 10^2 or 10^3 . Very large λ could yield poor performance in

Table 4.5: Comparison of CPU time required for 500 iterations of NMF, GNMF and L1GNMF. The values (in seconds) were averaged over 50 random tests.

Algorithm	NMF	GNMF	L1GNMF
TIDIGIT, $R=20$	404	7578	896
TIDIGIT, $R=25$	580	8465	977
TIDIGIT, $R=30$	615	11040	1075

L1GNMF. For the experiments on TIDIGITS (with unigram) and Caltech256 with blindly clustered features, $\lambda=1$ seems to be a safe choice for GNMF, though it does not improve significantly over NMF. By using the oracle features, GNMF with $\lambda=100$ improves NMF noticeably.

L1GNMF is observed to outperform GNMF for any adjacency threshold Γ in the experiments of spoken and visual pattern discovery. Graph adjacency with small Γ could contain wrong connections between features, which can explain the increase of error rates in Figure 4.7(c) with small Γ 's. Large Γ produces very sparsely connected graph adjacency matrix, thus few features are regularized, so the performance of L1GNMF and GNMF tend to be similar. A rule of thumb for the selection of Γ is the ratio of the number of utterances to the number of patterns as suggested in section 4.3 and section 4.4.1.

4.4 Evaluation of L1GNMF on Other Databases

In this section, two more databases are utilized to evaluate the proposed algorithm by comparing it with NMF and GNMF [15].

4.4.1 Visual Object Discovery on Caltech256

In this section, the performance of the proposed model is tested on an image classification task in terms of the visual objects it discovers. A subset of 20 categories of the Caltech256 database is selected as shown in Table 4.6 [111].

BoF representation of images

Images are represented by the scale invariant feature transform (SIFT), which extracts local structures of images to overcome the variation due to scale, illumination and viewpoints [68]. The method extracts interest points at

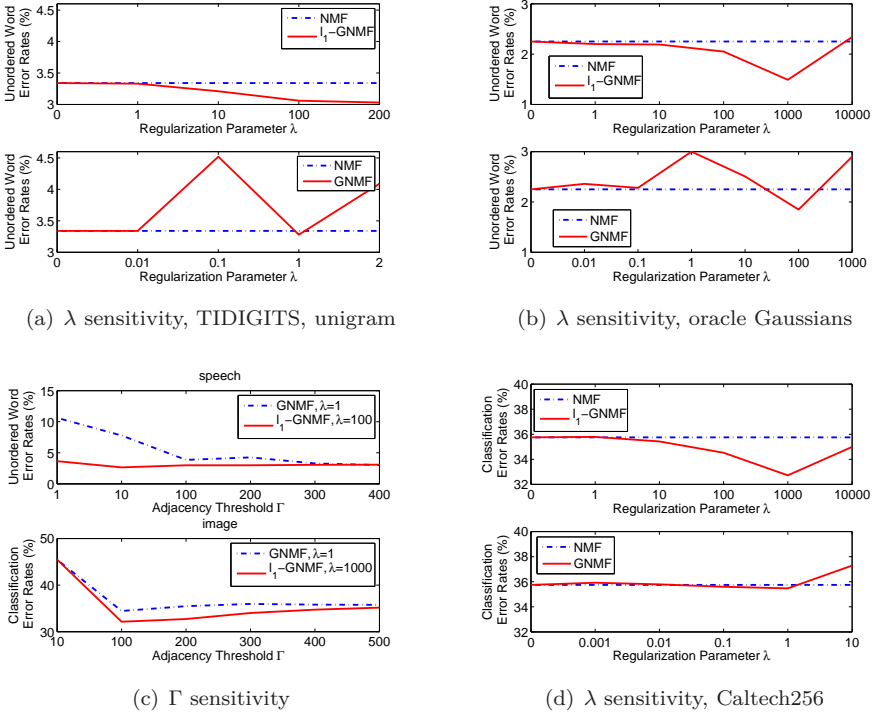


Figure 4.7: Parameter sensitivity analysis of GNMf and L1GNMF. (a) is with $R=30, \Gamma=300$ for speech. (b) is with $R=30, \Gamma=200$ for the oracle Gaussians. (c) is with $R=30$ for speech (top) and for image (bottom). (d) is with $R=30, \Gamma=200$ for image. This analysis provides evidences for the choices of the regularization parameters in GNMf and L1GNMF.

different scales like the circles drawn in Figure 4.8(a) and describes them by two kinds of information. The first one is the location (x_t, y_t, r_t, o_t) which indicates where the interest point is located by the coordinate (x_t, y_t) , the scale r_t at which the interest point is extracted, and the orientation o_t . The orientation o_t is discarded here. The second one is a histogram e_t which is a 128 dimensional vector for the description of the variation of gray scales around the interest point. By k -means clustering of the SIFT descriptors $\{e_t\}$ of the training images (color images are converted to be gray scale), M clusters are obtained called *visual words*. The occurrence statistics of the visual words serve as features used in the BoF representation of images, as explained below.

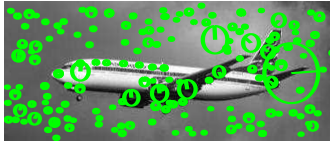
Table 4.6: 20 object categories selected from Caltech256

category ID	category	# training/testing images
1	American flag	73/24
2	fern	80/26
3	French horn	68/23
4	leopards 101	143/47
5	pci card	79/26
6	tombstone	68/23
7	airplanes 101	600/200
8	diamond ring	88/25
9	fire extinguisher	63/21
10	ketch 101	83/28
11	mandolin	70/23
12	rotary phone	63/21
13	Pisa tower	68/23
14	faces easy 101	326/108
15	dice	72/24
16	fireworks	75/25
17	killer whale	68/22
18	motorbikes 101	600/200
19	roulette wheel	63/21
20	zebra	71/24

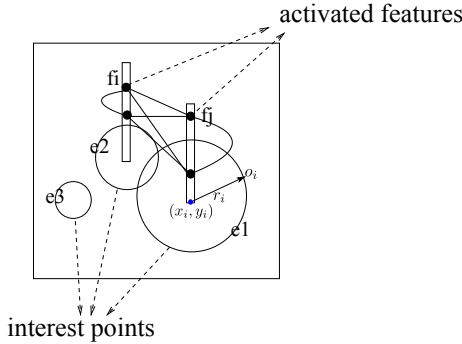
The descriptors of the training and the testing set are subsequently vector quantized by using the visual words. Soft-VQ is used to label the interest points with the visual words. For each interest point e_t , the top K visual words and the associated scores are selected as descriptor. Suppose the first $K + 1$ Euclidian distances (sorted in increasing order) are $z_{i_1}, z_{i_2}, \dots, z_{i_{K+1}}$, then the score on the k -th visual word is defined as $e^{-z_{i_k}/z_{i_{K+1}}}$. All scores of the retained visual words are subsequently transformed to values between 0 and 1 in every retained visual word:

$$p(f_{i_k}; e_t) = \frac{e^{-\frac{z_{i_k}}{z_{i_{K+1}}}} - e^{-1}}{1 - e^{-1}}, 1 \leq k \leq K + 1 \quad (4.24)$$

where f_{i_k} is the i_k -th visual word and e_t is the interest point. The $K + 1$ -th score is always zero, so only the top K visual words are retained for the descriptor e_t . By normalizing the scores to sum to 1 over all visual words, we can interpret them as *posterior probabilities* of the descriptors e_t on the visual words: $\Pr(f_{i_k}|e_t)$. By accumulating the posterior probabilities of all the descriptors on the j -th image \mathcal{I}_j as in Equation (4.25), an image is represented



(a) Illustration of SIFT points



(b) The definition of adjacency

Figure 4.8: Definition of graph adjacency between SIFT features. SIFT points are extracted with different scales in (a), so the adjacency of SIFT points depends on the scales and the distance between the two centers in (b).

by its bag of visual words,

$$V_{ij} = \sum_{e_t \in \mathcal{I}_j} \Pr(f_i | e_t) \quad i = 1, \dots, M \tag{4.25}$$

yielding $\mathbf{V}_{M \times N}$ on the training set.

Spatial adjacency of features

Two descriptors $(x_1, y_1, r_1, \sigma_1)$ and $(x_2, y_2, r_2, \sigma_2)$ are adjacent if $\frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}}{r_1+r_2} < 1.1$, i.e. if they are sufficiently close relative to their scales. A descriptor is also adjacent with itself. For each descriptor, N_{best} visual words with high scores are selected. The process is illustrated in Figure 4.8(b) where $N_{best}=2$, the circles are descriptors, the rectangular bars are the interest points labeled by the visual words, the small black dots form the N_{best} list of visual words, and the lines are connections of the retained visual words. The $N_{best} \times N_{best}$ visual words of two adjacent descriptors are adjacent (connected

in Figure 4.8(b)). By accumulating the adjacency frequencies of the visual words over all the adjacent descriptors of the images of the training data, the adjacency matrix \mathbf{U} is obtained. Next its diagonal elements are set to zero. It is made symmetric by $\mathbf{U} \leftarrow \mathbf{U} + \mathbf{U}^T$ and logical by $\mathbf{U} \leftarrow \mathbf{U} > \Gamma$, where Γ is a threshold. The degree matrix \mathbf{D} is a diagonal matrix with diagonal elements equal to the column sum of \mathbf{U} ; the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{U}$.

The spatial closeness in our model is exploited for any given set of visual words, which is different from the model in [36] and [121] where the spatial closeness of local patches (images in our terminology) is used as a constraint to train a particular set of visual words.

Results of visual object discovery and image classification

For each category, 3/4 of the images were used for training and the remaining 1/4 of the data for testing. The number of training and testing images are listed in Table 4.6.

The extraction of SIFT features and the training of visual words were performed with the software in [116] by using the default parameters. The number of visual words used for labeling a descriptor is $K=3$. The parameter to count adjacency of visual words is $N_{best}=3$. The threshold to make the graph adjacency matrix \mathbf{U} logical is $\Gamma \geq \#training\ images / \#Categories = 2835/20$. The basic assumption for this is again that an adjacency of features of some image category appears at least once in any image of the category. We took a slightly larger $\Gamma=200$ to avoid spurious adjacency from noise. The number of visual words was $M=2000$ and the regularization parameter is $\lambda=1000$. The model's sensitivity w.r.t. λ and Γ is analyzed in Figure 4.7.

We use *classification error rate* to evaluate the performance on visual object discovery. The entry of the estimate of the ground truth matrix of the testing set, $\hat{G}'_{l,n}$, indicates to which degree image n belongs to category l . Since each image only belongs to one category, we assign image n to the category with the largest activation in the corresponding column $\hat{\mathbf{G}}'_n$. The classification error rate is computed as the percentage of testing images classified incorrectly. The error rates of image classification from five random tests are reported in Table 4.7. We have tuned the regularization parameter of GNMF for best accuracy. More details of L1GNMF on Caltech256 can be found in [101].

Table 4.7: Classification error rates of NMF, GNMF and L1GNMF on Caltech256.

# patterns R	25	30	40	50	60
NMF	37.68±0.44	35.75±0.80	34.54±0.61	34.42±0.72	33.75±0.39
GNMF $\lambda=1$	35.55±0.82	35.47±0.87	33.79±0.53	33.08±0.43	33.02±0.49
L1GNMF $\lambda=1000$	34.69±1.37	32.72±1.10	32.24±1.32	30.16±0.77	30.50±0.34

4.4.2 Document Clustering on TDT2

We also conducted experiments on the TDT2 corpus (Topic Detection and Tracking) [15]. The TDT2 corpus consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs (CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In this experiment of document clustering, those documents appearing in two or more categories were removed, and only the largest 10 categories were kept, thus leaving us with $N=7456$ documents in total.

For the comparison between GNMF and L1GNMF, we use the data matrix \mathbf{V} provided by [14]. The data matrix is composed of the term (text words) frequencies weighted by their inverse document frequencies (tf-idf). The number of features, i.e. the number of terms or text words, was $M=36771$. The graph adjacency matrix $\mathbf{U}_{N \times N}$ was computed by taking the $P=7$ nearest neighbors for each document with the code provided in [14]. $R=10$ was the number of patterns (document categories in this task) which was set as the oracle number of document classes.

The following modifications are to apply our algorithm on the TDT2 database and to be comparable with the model in [14].

Tri-factorization with normalized cut weights

We update the original L1GNMF model to fit it into the document clustering problem.

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{S}, \mathbf{H}} \quad & \text{KLD}(\mathbf{V}\mathbf{\Gamma} || \mathbf{W}\mathbf{S}\mathbf{H}\mathbf{\Gamma}) + \lambda \text{Tr}((\mathbf{H}\mathbf{\Gamma})(\mathbf{\Gamma}^{-1}\mathbf{L}\mathbf{\Gamma}^{-1})(\mathbf{H}\mathbf{\Gamma})^T) \\ \text{s.t.} \quad & \sum_j \gamma_j H_{k,j} = 1, \forall k \end{aligned} \quad (4.26)$$

where $\mathbf{\Gamma} = \text{diag}(\gamma_1, \dots, \gamma_R)$ are called normalized cut weights (NCW) of the documents in [14]. The motivations for the modifications are as follows.

- Tri-factorization learning.

The probabilistic model is changed from $\Pr(t_i|d_j) = \sum_k \Pr(t_i|z_k)\Pr(z_k|d_j)$ to $\Pr(t_i, d_j) = \sum_k \Pr(t_i|z_k)\Pr(d_j|z_k)\Pr(z_k)$. This is because there are much more text words or features than documents, i.e. the number of rows of \mathbf{V} is much larger than the number of columns. This is because, to apply the original L1GNMF model on $\mathbf{V}^T \approx \mathbf{H}^T \mathbf{W}^T$ with the column-wise normalization of \mathbf{H}^T , we have to normalize the rows of \mathbf{V} . This would have numerical problems due to the rareness of related documents given a feature. Also its corresponding probabilistic model $\Pr(d_j|t_i) = \sum_k \Pr(d_j|z_k)\Pr(z_k|t_i)$ is not well defined.

- Weighted normalization.

With the NCW $\mathbf{\Gamma}$, the normalization of \mathbf{H} is also weighted like $\sum_j \gamma_j H_{k,j} = 1$ to make the proposed algorithm L1GNMF still workable for the newly defined optimization problem.

With the following transformation, $\tilde{\mathbf{V}} = \mathbf{V}\mathbf{\Gamma}$, $\tilde{\mathbf{H}} = \mathbf{H}\mathbf{\Gamma}$ and $\tilde{\mathbf{L}} = \mathbf{\Gamma}^{-1}\mathbf{L}\mathbf{\Gamma}^{-1}$, the optimization problem in Eq.(4.26) can be transformed to a graph regularized non-negative matrix tri-factorization (GNMTF) problem as follows.

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{S}, \tilde{\mathbf{H}}} \quad & \text{KLD}(\tilde{\mathbf{V}} || \mathbf{W}\mathbf{S}\tilde{\mathbf{H}}) + \lambda \text{Tr}(\tilde{\mathbf{H}}\tilde{\mathbf{L}}\tilde{\mathbf{H}}^T) \\ \text{s.t.} \quad & \sum_j \tilde{\mathbf{H}}_{k,j} = 1, \forall k \end{aligned} \quad (4.27)$$

When updating \mathbf{W} , one can put $\mathbf{S}\tilde{\mathbf{H}}$ together and use the normal NMF algorithm. When updating $\tilde{\mathbf{H}}$, one can put $\mathbf{W}\mathbf{S}$ together and apply L1GNMF with a transposition. The update of \mathbf{S} is $\mathbf{S} \leftarrow \mathbf{S} \odot (\mathbf{W}^T * (\tilde{\mathbf{V}} \oslash (\mathbf{W} * \mathbf{S} * \tilde{\mathbf{H}})) * (\tilde{\mathbf{H}})^T)$. At the end of the update iterations, \mathbf{H} is obtained by re-scaling $\tilde{\mathbf{H}}$ like $\tilde{\mathbf{H}}\mathbf{\Gamma}^{-1}$. Then \mathbf{H} is utilized for evaluation as described later.

Initializations and parameters

***k*-means initialization:** Initialization selection is applied in the GNMF of [14]: *k*-means clustering is performed on the columns of \mathbf{V} to get cluster centers which serve as the initial columns of \mathbf{W} and as the coefficients in \mathbf{H} . In L1GNMF, \mathbf{H} is scaled to $\tilde{\mathbf{H}}$, together with \mathbf{W} , to initialize Eq.(4.27). With β NMF iterations, to update $\{\mathbf{W}, \tilde{\mathbf{H}}\}$, we obtain a cost value. Repeating the

above k -means process α times, and select the \mathbf{W} and $\tilde{\mathbf{H}}$ with the lowest cost in Eq.(4.27) with β iterations and continue the NMF iterations $N_1 - \beta$ times.

larger N_1 : N_1 is the number of main NMF iterations. In the experiments of GNMF where each iteration requires much more computations than L1GNMF, N_1 was selected as 50, but here thanks to the fast computation we can take a large N_1 .

Table 4.8: Performance of NMF,GNMF and L1GNMF on TDT2

	R	α	β	N_1	CPU time (s)	error rates
NMF	8	10	10	500	436	33.79±4.44
GNMF	8	10	10	50	6060	16.97±4.37
L1GNMF	8	10	10	500	605	17.90±4.31
NMF	10	10	10	500	570	29.62±4.79
GNMF	10	10	10	50	7793	13.43±3.77
L1GNMF	10	10	10	500	725	15.30±4.44
NMF	12	10	10	500	1278	35.24±5.60
GNMF	12	10	10	50	8949	13.30±4.10
L1GNMF	12	10	10	500	1524	14.61±5.04
NMF	15	10	10	500	1331	40.50±5.38
GNMF	15	10	10	50	10521	12.13±2.82
L1GNMF	15	10	10	500	1791	13.66±4.39

Evaluation

We use the same method as in [14] to evaluate GNMF and L1GNMF on the TDT2 database. The dimension of the documents is first reduced from M to R , i.e. from \mathbf{V} to \mathbf{H} . k -means clustering is subsequently applied to group the documents (columns of \mathbf{H}) into R_0 classes where R_0 is the number of grounding labels. The Hungarian algorithm is subsequently applied to link each class to a grounding label. Finally the mislabeled documents are counted to compute the error rate. Being different from the examples on TIDIGITS and Caltech 256, there is no training or testing dataset. The learning and evaluation are both conducted on the same dataset. The results are given in Table 4.8. On this database, L1GNMF gets similar performance as GNMF but with much lower CPU cost.

4.5 How the Graph Regularization Works

4.5.1 Interpreting the Discovered Patterns

For better understanding the discovered patterns, we link them to grounding truth and show some examples for each of the discovered patterns.

Links between patterns and grounding labels

A method of interpreting the discovered patterns is by reading the mapping matrix \mathbf{Q} obtained from Eq.(4.21). Some mapping matrices from the unigram model of TIDIGITS are shown in Figure 4.9, where for clarity, the columns are permuted by the order of digits they represent. The top drawings are links between the discovered patterns and the digits and the bottom drawings are the links between the patterns and the genders. From Figure 4.9(b), we can see the patterns obtained by L1GNMF coincide well with the ground truth of both digits and genders. Every pattern represents some version of a digit. Some digits are modeled by more than one pattern. The NMF model can almost do the same job, but patterns 21 through 25 are mixtures of several digits in Figure 4.9(a). That means the NMF model discovers mixtures of several digits from the same gender, which does not make much sense temporally. For instance, from Figure 4.9(a) it can be read that pattern 21 discovered by NMF is related to both “six” and “eight” and that pattern 25 is related with “two” and “five”. However, L1GNMF created separate patterns for every digit as seen in Figure 4.9(b).

For this *pure* dataset (pure in the sense that all data are composed of 11 repeated patterns (“one” to “nine”, “zero” and “oh”) and silence), L1GNMF has a better ability than NMF to figure out that the intrinsic dimension (22 as we know) is between 20 and 25 as the steepest descent of error rates happens in this interval in Table 4.1, Table 4.3 and Table 4.2.

Examples of a discovered pattern

In the experiments of Caltech256, the reduction in classification error rates from NMF to L1GNMF obtained in Table 4.7 may be explained by a similar phenomenon of grouping adjacent features into one pattern, much like in the speech example. However, the benefit is not that significant on the image data. Some categories are mixed into one pattern as seen in Figure 4.10(b). This is because, unlike the pure dataset TIDIGITS, the Caltech256 database

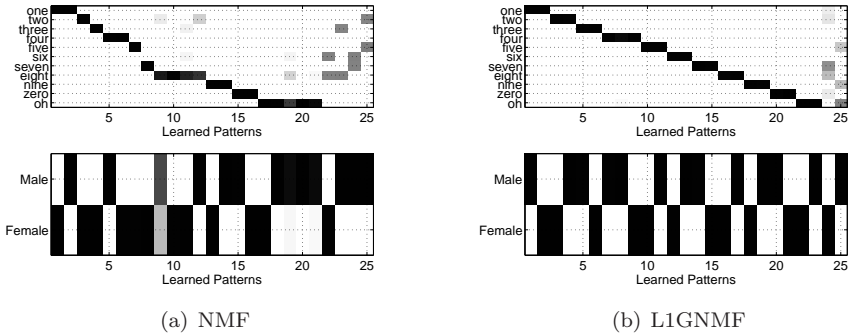


Figure 4.9: The mapping matrices obtained on the speech data by NMF and L1GNMF with the same initial \mathbf{W} and \mathbf{H} . The top panel on each figure links the discovered patterns to the digits, while the bottom panel interprets the patterns by genders. From the top panels, we observe that one digit can be linked to several patterns. The bottom panels show that the patterns linked to the same digit can be separated into male and female versions. L1GNMF gives much more clear links than NMF.

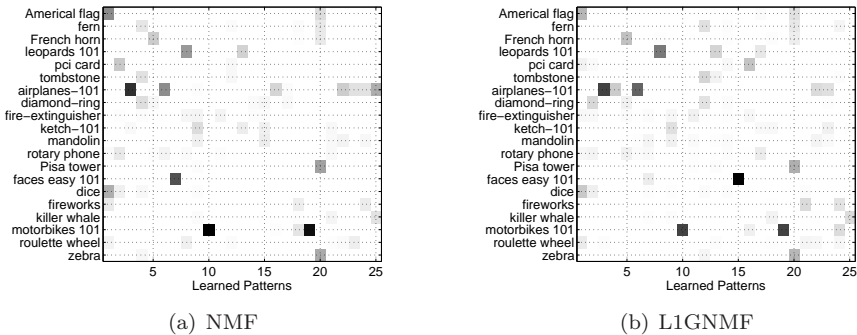


Figure 4.10: The mapping matrices obtained on the image data by NMF and L1GNMF with the same initial \mathbf{W} and \mathbf{H} .

contains a lot of background objects which are not reflected by the categories. However, most of the categories from Catech101 are modeled perfectly like category “leopards 101”, “airplanes 101”, “faces easy 101” and “moterbikes 101”.

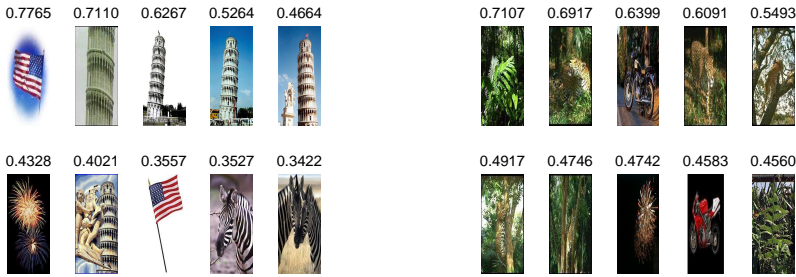
Some interesting interpretations of the obtained patterns are revealed by analyzing the top activated images of each pattern. The top-10 images

with the highest activations in $\mathbf{H}'_{k,:}$. (\mathbf{H}' is the co-efficient matrix containing the weight of the patterns on the testing set) for pattern k are shown in Figure 4.11. From Figure 4.10(b), we observed that pattern 20 is related to three categories: category 1 (American flag), category 13 (pisa tower) and category 20 (zebra). Their common theme is *stripes* as shown in Figure 4.11(a). Pattern 24 models some kind of spread shape similar with “fern” and “fire works” as witnessed by Figure 4.11(b). Hence the patterns that are discovered with unsupervised learning do not necessarily correspond to image categories, but to some intermediate image characteristics. This fact suggests that it is not wise to build one single unsupervised model to relate the low-level visual features (visual words) and the high-level semantic categories directly. Instead some intermediate levels of abstraction are necessary to discover the shapes like stripes and fern as suggested by the deep learning theory [46][81].

By analyzing the number of images correctly classified per category in Figure 4.12, we conclude that L1GNMF improves the classification by performing better than NMF on the categories with a small number of images, e.g. the 17-th category, “killer whale”. In Figure 4.10(a), we see that NMF produces a common pattern (the 25-th column) for “airplane” and “killer whale”, probably because the two objects have a similar shape. L1GNMF models the “killer whale” quite well in the 25-th column of Figure 4.10(b) with little confusion with other categories. The above phenomenon is also observed by comparing Figure 4.11(c) and Figure 4.11(d).

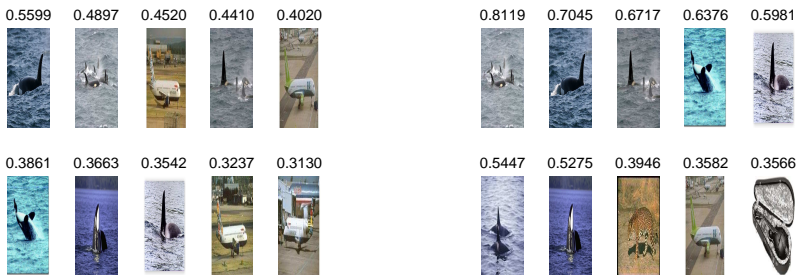
4.5.2 The Role of Graph Regularization

The merit of graph regularization can be depicted by the following explanation of NMF and GNMF. With the graph regularization, the basis vectors \mathbf{W}'_k 's in Figure 4.13 are twisted in order to cover the *adjacent* data points of the training dataset, which would therefore yield better results which has larger probability to cover both training and testing data points than the solutions from NMF. Certainly, the performance of NMF can be improved by increasing the number of basis in NMF (until over-training), as is observed in the extensive experiments on TIDIGITS, Caltech256 and TDT2. With graph regularization, the model's sensitivity with respect to the factorization dimension and the number of features is alleviated.



(a) 10 images with highest activations with L1GNMF for the 20-th column of Figure 4.10(b).

(b) 10 images with highest activations with L1GNMF for the 24-th column of Figure 4.10(b).



(c) 10 images with highest activations with NMF for the 25-th column of Figure 4.10(a)

(d) 10 images with highest activations with L1GNMF for the 25-th column of Figure 4.10(b)

Figure 4.11: 10 prototypical images and their corresponding scores for each of the selected patterns.

4.6 Conclusion

Proximity within a given set of features has been exploited to improve unsupervised pattern discovery in a graph regularized NMF framework. A new algorithm to solve the optimization problem as well as preserving the ℓ_1 normalization of the patterns has been proposed and its convergence was proven. Experiments on spoken and visual pattern discovery showed the efficacy and robustness of the algorithm. Besides the temporal and spatial proximity between features, other kinds of proximity, such as synonymy in text processing, can also be incorporated easily into the proposed model by

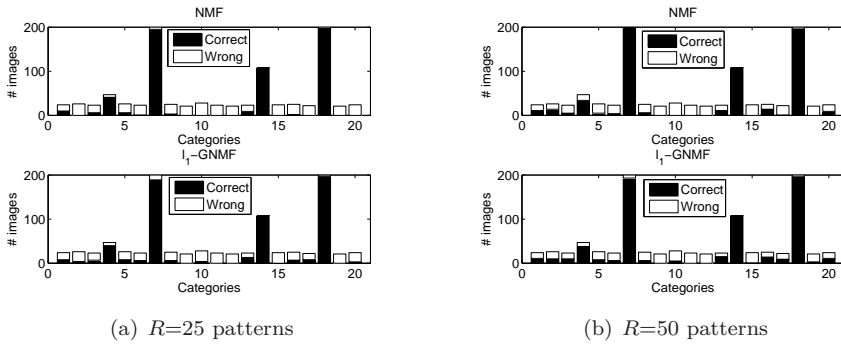


Figure 4.12: The comparison between NMF and L_1 GNMF of the number of images classified correctly per category.

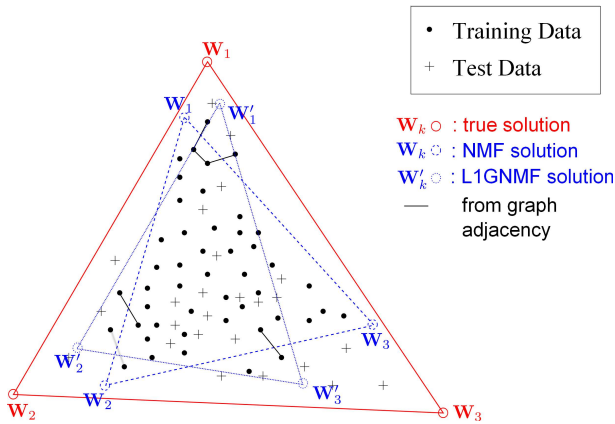


Figure 4.13: The geometric explanation of graph regularized NMF. Compared to Figure 4.1, the graph regularization can adjust the NMF solutions by grouping adjacent points together. The obtained solutions W'_k would perform better than the NMF solution W_k in the sense of covering training and testing data points.

constructing their respective graph adjacency matrices.

However, the examples given in spoken and visual pattern discovery are not easily extended to cases with large pattern dictionaries. The reason is that, besides the non-convexity of the factorization and the large computational budget in the matrix factorization, a large number of feature-pattern associations need to be learned for each pattern. The speech and image task discussed here will

only scale to large pattern inventories if some form of intermediate associations (or sub-patterns) can be learned and reused among patterns, hence reducing the number of parameters to be learned for a pattern. Inspired by human cognition and successes in deep learning [46][81], we suggest a multi-layer approach in the next chapter.

Chapter 5

Learning of Sub-word Units with Non-negative Matrix Tri-Factorization

Word-sized spoken pattern discovery and evaluation, with supervised and unsupervised NMF learning methods, were discussed in the previous chapters. In this chapter, we study machine learning methods for discovering sub-word units with a finer granularity. Non-negative matrix tri-factorization (NMTF) is applied to the word-sized spoken patterns obtained in Chapter 3 (supervised) and Chapter 4 (unsupervised) to learn sub-word units. The units turn out to be HMM states or segments of consecutive states as intermediate representations. Its good performance on vocabulary acquisition in terms of high accuracy and fast learning rate demonstrates the usefulness of this sub-word unit representation of speech.

5.1 Hidden Markov Models of Spoken Words

Hidden Markov models (HMM) have been used successfully in automatic speech recognition (ASR) for several decades. Their success can be attributed to at least two aspects. One is modeling the observations of the hidden states with statistical models, e.g. Gaussian mixture models (GMM). The other is modeling the sequential nature of speech with a left-to-right structure. In this

section, we first explain the configuration of HMMs for word recognition and then study the role of hidden states to improve our NMF learning models.

5.1.1 Configuration and Parameters

A three-state HMM to model a word \mathcal{W}_r is shown in the left part of Figure 5.1. The HMM is characterized by the following elements.

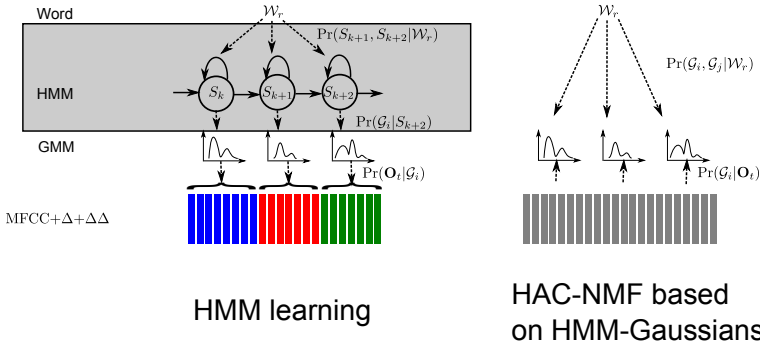


Figure 5.1: Comparison between a HMM and a HAC-NMF model of a spoken word. In a HMM, a *word* is modeled by *hidden states* while a *hidden state* is modeled by a GMM, i.e. *Gaussians and weights* where the Gaussians are the probability distributions on the observation frames. However, in HAC-NMF, though we have introduced *posterior probabilities on Gaussian pairs* as a bigram model in Chapter 3, the layer of hidden states is missing.

- The observation vector per frame \mathbf{O}_t which is usually a MFCC+ Δ + $\Delta\Delta$ vector at frame t .
- The hidden states $\{S_1, \dots, S_k, \dots, S_K\}$.
- The acoustic model $b_k(\mathbf{O}_t) = \Pr(\mathbf{O}_t | S_k)$ which is the likelihood of frame \mathbf{O}_t given state S_k . By modeling a state as a Gaussian mixture model (GMM) with weights $\Pr(\mathcal{G}_i | S_k)$, the acoustic likelihood is $\Pr(\mathbf{O}_t | S_k) = \sum_i \Pr(\mathbf{O}_t | \mathcal{G}_i) \Pr(\mathcal{G}_i | S_k)$. Let \mathbf{A} denote the weight matrix or emission matrix with elements $A_{ik} = \Pr(\mathcal{G}_i | S_k)$.
- The transition matrix $\mathbf{T}_{K \times K}$ whose element $T_{k,k'}$ is the conditional probability of transition from S_k to $S_{k'}$: $\Pr(S_{k'} | S_k), \forall t$.

The basic per-word HMMs can thus be connected with each other according to a language model for the prospective ASR. The details of the training of

HMM and its application to ASR will be given in the next chapter. Here we draw the basic configuration of the per-word HMM to explain our motivation for unsupervised learning of hidden states.

The NMF model of word learning is shown on the right part of Figure 5.1. An observation frame \mathbf{O}_t and the Gaussians are linked through the posterior probabilities of Gaussian co-occurrences $\Pr(\mathcal{G}_i, \mathcal{G}_j | \mathbf{O}_t)$ which is constructed from the Gaussian posterior probabilities $\Pr(\mathcal{G}_i | \mathbf{O}_t)$ in the HMM. If using the Gaussians obtained from GMMs of a HMM (which are called *oracle* Gaussians in Chapter 4), one major difference between NMF and the HMM baseline is removed for comparison to the performance of HMM. A second difference that is addressed in this chapter is the *shallow* structure of the NMF model, i.e. a word is characterized directly from its statistics of Gaussian posteriors. In contrast, an HMM recognizes a state level, where each state is in turn described in terms of a Gaussian mixture. Hence, a layer of hidden states is missing in NMF, while the layer is important to reflect the long-term dependencies in speech while allowing for acoustic variation as will be explained below.

5.1.2 The Role of Hidden States

In hidden states, “hidden” means that the state is not directly visible to the observer but the observable outputs depend on the state. Each state has a probability distribution $\Pr(\mathbf{O}_t | S_k)$ over the possible observations. Therefore the sequence of observations generated by an HMM gives some information about the sequence of states. To evaluate if a sequence is generated by an HMM, Viterbi alignment is conducted to transform the observation sequence into a state sequence with maximal likelihood. The motivation for introducing hidden states is two-fold.

- Tolerance to variation
Speech is subject to multiple sources of variations. Even for the same word, there are no two observation sequences that are the same. Hence the exact frame-by-frame matching of time-frequency features for recognition would be impossible. Instead, hidden states modeled by probabilistic distributions would tolerate some frame-level variation and give similar speech sounds high likelihoods while different speech sounds are given low likelihoods.
- Transitions to reflect long-term dynamics
The local dynamic property of speech is partially modeled in the Δ and $\Delta\Delta$ features. However, the long-term dependencies are not. In ASR systems, long term dependencies are modeled by transitions between

hidden states which is a higher level of abstraction of transitions between observations. We have observed a large drop of unordered word error rate from the unigram model (only considering the occurrence of observations) to the bigram model (considering the co-occurrences of observations) in Chapter 3. Hence the contextual dependencies between hidden states would be further helpful to improve the model's performance.

In this chapter, an intermediate abstraction level, comparable to an HMM state but not limited to first order memory, is introduced. The creation of the intermediate level (referred to as hidden units) does not require supervision and is also obtained by a matrix factorization. Effectively, the co-occurrence modeling in HAC+NMF now happens at the level of the hidden units. The benefit of such an intermediate level is that since there are less hidden units than Gaussians, the co-occurrence statistics require less data to be estimated. This will result in an increased learning rate for new words, since the first layer (the relation between Gaussians and hidden units) is reused.

5.2 Tri-Factorization Learning of Sub-word Units

In the previous chapters, the representation of speech was in the form of a bag-of-features, an example of which is the histogram of the acoustic co-occurrences (HAC). Specially, we have proposed and evaluated the representation by using Gaussian co-occurrences. Given the strong relations between co-occurrence statistics of observations and the hidden states [50, 21, 114, 123] via a matrix tri-factorization, we will study learning methods to exploit sub-word units like hidden states from the co-occurrence representations learned in Chapter 2 and Chapter 3 in this section.

5.2.1 Co-occurrence Statistics and Hidden States

By expressing the idea of [62, 114, 21] in the terminologies of Gaussian co-occurrences, where the observation symbols are Gaussian identities \mathcal{G}_i , the intimate relation between an HMM and non-negative low-rank decompositions of co-occurrences is presented in Eq.(5.1).

$$= \frac{\Pr(\mathcal{G}_i, \mathcal{G}_j)}{\sum_{k,l} \Pr(\mathcal{G}_i | S_k) \Pr(S_k, S_l) \Pr(\mathcal{G}_j | S_l)} \quad (5.1)$$

where $\Pr(\mathcal{G}_i|S_k)$ corresponds to the emission matrix \mathbf{A} in a discrete density HMM or the Gaussian weight matrix in a continuous density HMM and $\Pr(S_k, S_l)$ expresses the transitions between hidden states. The above process is illustrated in Figure 5.2, where a hidden state is described by the Gaussians surrounding it with probabilistic associations $A_{i,k}$. The co-occurrences of Gaussians $\Pr(\mathcal{G}_m, \mathcal{G}_{m'})$ are thus strongly related to the co-occurrences of the underlying hidden states $\Pr(S_k, S_{k'})$ via the emission probabilities.

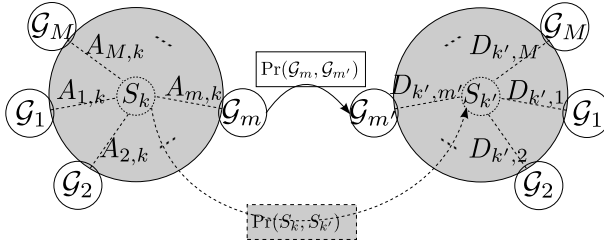


Figure 5.2: The relationship between Gaussian co-occurrences and hidden states. Every hidden state is modeled by a multinomial distribution on the Gaussians. Hence, the Gaussian co-occurrence is modeled by *the co-occurrences of hidden states and the emission of Gaussians from every hidden state.*

The matrix form of Eq.(5.1) is in Eq. (5.2),

$$\mathbf{C} = \mathbf{A}\mathbf{B}\mathbf{D} \tag{5.2}$$

where \mathbf{C} is the co-occurrence matrix of Gaussians $C_{i,j} = \Pr(\mathcal{G}_i, \mathcal{G}_j)$, \mathbf{B} is the co-occurrence matrix of the hidden states $B_{k,l} = \Pr(S_k, S_l)$, \mathbf{A} with $A_{i,k} = \Pr(\mathcal{G}_i|S_k)$ is the emission matrix on the left side and \mathbf{D} is the one on the right side. As the number of Gaussians is usually larger than the number of states, the role of \mathbf{A} or \mathbf{D}^T is to *embed* some hidden space of states to the space of Gaussians. Hence they are called *embedding matrices* and will be obtained by matrix factorization later. If the process is really Markovian, then $\mathbf{A} = \mathbf{D}^T$. However, it is worthwhile to estimate \mathbf{A} and \mathbf{D} separately such that the impact of this assumption on accuracy can be validated.

We call the prospective states *hidden units* because the learning of Eq. 5.2 is unsupervised. Thus the columns of \mathbf{A} and the rows of \mathbf{D} don't have to correspond to the hidden states of a supervisedly trained HMM, but the two are expected to behave similarly. An algorithm to learn \mathbf{A} , \mathbf{B} and \mathbf{D} from Gaussian co-occurrences \mathbf{C} , by solving $\operatorname{argmin}_{\mathbf{A}, \mathbf{B}, \mathbf{D}} \operatorname{KLD}(\mathbf{C} || \mathbf{A}\mathbf{B}\mathbf{D})$ with normalization of the columns of \mathbf{A} and the rows of \mathbf{D} is illustrated in Algorithm 5.1 [124].

Besides the above co-occurrence statistics, high order statistics, such as co-occurrences of Gaussian co-occurrences, were deployed for learning of HMMs

in [21]. The combinatorial configurations of high order co-occurrences are put into a *Hankel* matrix in [33]. Due to the complexity, the method in [33] is only valid for HMM with a few states and observations.

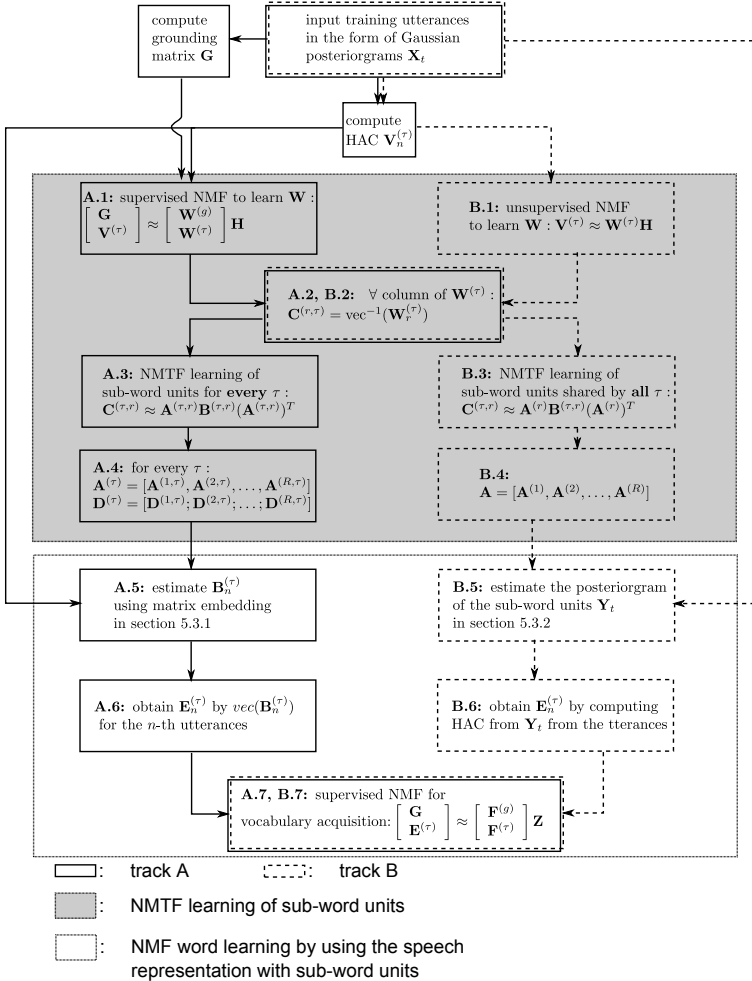


Figure 5.3: The flowchart of the proposed learning methods. Two tracks are presented where *Track A* is for supervised learning of hidden units and *Track B* is for unsupervised learning of hidden units.

5.2.2 NMF Learning of the Gaussian Co-occurrence Matrix

The rank and complexity of the learning model in Eq.(5.2) grows linearly with the number of hidden states of the underlying HMM. Its validity is only guaranteed for the “thin” factorization where $\text{rank}(\mathbf{B}) \ll \text{rank}(\mathbf{C})$, i.e. to learn a couple of hidden states with a high dimension of observations. In [62, 21], only HMMs of far smaller complexity are considered. So a good way to overcome this problem is to learn a HMM for each word separately. In this case, we should construct a co-occurrence matrix $\mathbf{C}^{(r)}$ for word \mathcal{W}_r . However, the available training data is continuous speech where word boundary information is not available. An additional segmentation step should be avoided if possible.

Based on the results of Chapter 2 and Chapter 3, the discovered vocabulary patterns are stored in the columns of the acoustic pattern matrix \mathbf{W} which are the *repeated* parts in the original data matrix \mathbf{V} . For supervised NMF, the model is,

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{V} \end{bmatrix} \approx \begin{bmatrix} \mathbf{W}^{(g)} \\ \mathbf{W} \end{bmatrix} \mathbf{H}. \quad (5.3)$$

For unsupervised NMF in Chapter 4, the model is,

$$\mathbf{V} \approx \mathbf{W}\mathbf{H}. \quad (5.4)$$

The vocabulary representation in \mathbf{W} is just the co-occurrence statistics of a word model. So now we can use the discovered per-word Gaussian co-occurrence model, i.e. the columns \mathbf{W}_r , as co-occurrence matrices $\mathbf{C}^{(r)}$, to generate a set of embedding matrices. Given the two techniques for learning \mathbf{W} , in Figure 5.3, we use Track A.1 to denote the stages related to *supervised* learning of hidden units and use Track B.1 to denote the stages related to *unsupervised* learning of hidden units. The tracks will be denoted in the following contexts as well.

The NMTF learning process is depicted in Figure 5.4. For some lag τ , denote its corresponding block in \mathbf{W} as $\mathbf{W}^{(\tau)}$. The r -th column of $\mathbf{W}^{(\tau)}$ is reshaped to a $M \times M$ matrix of co-occurrences of Gaussians $\mathbf{C}^{(\tau,r)}$ (Track A.2 and Track B.2). Then with the optimization,

$$\text{argmin}_{\mathbf{A}^{(\tau,r)}, \mathbf{B}^{(\tau,r)}, \mathbf{D}^{(\tau,r)}} \text{KLD}(\mathbf{C}^{(\tau,r)} || \mathbf{A}^{(\tau,r)} \mathbf{B}^{(\tau,r)} \mathbf{D}^{(\tau,r)}), \quad (5.5)$$

we obtain $\mathbf{A}^{(\tau,r)}$, $\mathbf{D}^{(\tau,r)}$ as embedding matrices of the hidden states to the Gaussians and $\mathbf{B}^{(\tau,r)}$ as the co-occurrences of the hidden states of this word with contextual dependence τ . The process corresponds to Track A.3 in Figure 5.3. The detailed algorithms for non-negative matrix tri-factorizations

are given in the next subsection. $\mathbf{B}^{(\tau,r)}$ is set to be upper-band-diagonal to enforce the left-to-right structure of the HMM.

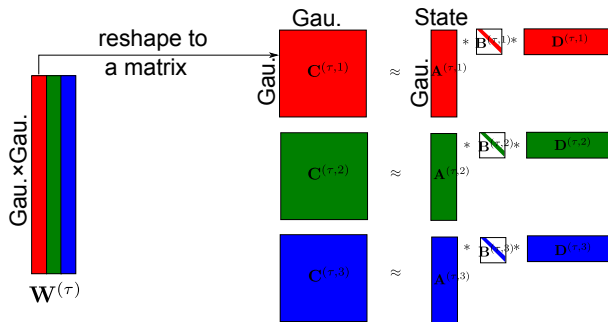


Figure 5.4: Learning of hidden states from the matrix \mathbf{W} learned by NMF. A column of \mathbf{W} is the bag of co-occurrences of features for a discovered pattern. \mathbf{W} can have multiple lags τ which inherited from the data matrix \mathbf{V} as presented in Eq.(2.21). Taking the block of \mathbf{W} corresponding to lag τ , $\mathbf{W}^{(\tau)}$, as an example, a column of $\mathbf{W}^{(\tau)}$, $\mathbf{W}_k^{(\tau)}$, is first reshaped to a co-occurrence matrix $\mathbf{C}^{(\tau,k)}$. NMTF is subsequently performed on the co-occurrence matrix to learn hidden states.

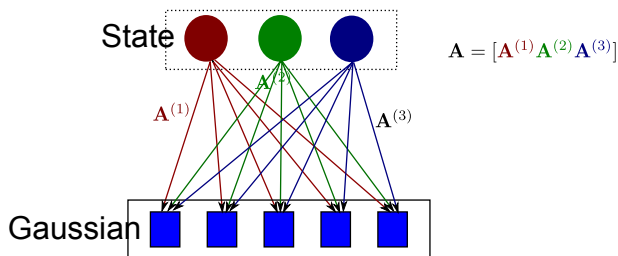


Figure 5.5: The graphical model between Gaussians and hidden states learned by the symmetric NMTF. The weights of the Gaussians for the hidden state S_k are learned and stored in a column vector $\mathbf{A}^{(r)}$. The vectors are stacked to form the full weight matrix.

By applying the same procedure for all the vocabulary patterns $r = 1, \dots, R$ with the fixed τ , we estimate the overall embedding matrices $\mathbf{A}^{(\tau)} = [\mathbf{A}^{(\tau,1)}, \dots, \mathbf{A}^{(\tau,R)}]$ and $\mathbf{D}^{(\tau)} = [(\mathbf{D}^{(\tau,1)})^T, \dots, (\mathbf{D}^{(\tau,R)})^T]^T$ and the overall co-occurrence matrix of hidden states $\mathbf{B}^{(\tau)}$ is the block diagonal matrix with blocks $\mathbf{B}^{(\tau,1)}, \dots, \mathbf{B}^{(\tau,R)}$. Different patterns have different hidden states. For any other τ , the process is the same. The obtained embedding matrices $\mathbf{A}^{(\tau)}$ and $\mathbf{D}^{(\tau)}$ are saved for processing test data in Track A.4. In this process, the

co-occurrence matrix of hidden states, $\mathbf{B}^{(\tau)}$, is estimated as will be explained in section 5.3.

More constraints can be imposed by assuming all the embedding matrices $\mathbf{A}^{(\tau)}$ and $(\mathbf{D}^{(\tau)})^T$ are equal. That is, the associations between Gaussians and hidden states are described solely by one emission matrix \mathbf{A} like in HMM. The transitions with different lags are reflected in the co-occurrence matrix of hidden states $\mathbf{B}^{(\tau)}$. Therefore, the formulation of the new NMTF problem for each $\mathbf{C}^{(\tau,r)}$ is,

$$\operatorname{argmin}_{\mathbf{A}^{(\tau)}, \mathbf{B}^{(\tau,r)}} \sum_{\tau} \operatorname{KLD} \left(\mathbf{C}^{(\tau,r)} \parallel \mathbf{A}^{(\tau)} \mathbf{B}^{(\tau,r)} (\mathbf{A}^{(\tau)})^T \right), \quad (5.6)$$

which is denoted by Track B.3 in Figure 5.3.

The obtained associations between Gaussians and hidden states can be represented by a directed probabilistic graphical model as shown in Figure 5.5. The overall embedding matrix is obtained by stacking the embedding matrices from every column of \mathbf{W} , i.e. $\mathbf{A} = [\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(R)}]$ denoted by Track B.4 in Figure 5.3.

5.2.3 Non-negative Matrix Tri-Factorization (NMTF)

Notice there are two kinds of connections (or weights) between Gaussians and states: \mathbf{A} denotes the matrix of left-Gaussian probabilities $\Pr(\mathcal{G}_i | S_k)$, \mathbf{D} denotes the matrix of right-Gaussian probabilities $\Pr(\mathcal{G}_j | S_l)$ in Figure 5.2. Here we give out the algorithms for both the asymmetric tri-factorization (Track A.3 in Figure 5.3) and the symmetric one where \mathbf{A} is assumed to be equal to \mathbf{D}^T . The second one is discussed together with multiple contextual dependencies (Track B.3 in Figure 5.3).

Asymmetric tri-factorization

With the algorithm in Algorithm 5.1, the embedding matrices \mathbf{A} and \mathbf{D} , the hidden units and their co-occurrence matrix \mathbf{B} are obtained. The asymmetrical characteristic of the co-occurrence matrix \mathbf{C} is not only reflected in the co-occurrence matrix of hidden states, \mathbf{B} , but also in the left and right embedding matrices \mathbf{A} and \mathbf{D} . The derivation of the algorithm is straightforward, one just need to fix two variables to update the remaining one as in NMF. The convergence is thus also ensured by the proof of NMF in [65].

```

input :  $\mathbf{C}$  and initial estimates for  $\mathbf{A}, \mathbf{B}$  and  $\mathbf{D}$ 
output:  $\mathbf{A}, \mathbf{B}, \mathbf{D}$ 

 $n = 0$ ;
while  $n < \# \text{ iterations}$  do
   $P_{k,l} \leftarrow \sum_i (\mathbf{A} * \mathbf{B})_{i,l}, 1 \leq k \leq M$ ;
   $\mathbf{D} \leftarrow \mathbf{D} \odot (\mathbf{B}^T * \mathbf{A}^T * (\mathbf{C} \oslash (\mathbf{A} * \mathbf{B} * \mathbf{D}))) \oslash \mathbf{P}$ ,
   $D_{k,l} \leftarrow D_{k,l} / \sum_j D_{k,j}, B_{l,k} \leftarrow B_{l,k} * \sum_j D_{k,j}$ ;
   $Q_{k,l} \leftarrow \sum_i (\mathbf{B} * \mathbf{D})_{k,i}, 1 \leq l \leq M$ ;
   $\mathbf{A} \leftarrow \mathbf{A} \odot ((\mathbf{C} \oslash (\mathbf{A} * \mathbf{B} * \mathbf{D})) * \mathbf{D}^T * \mathbf{B}^T) \oslash \mathbf{Q}$ ,
   $A_{k,l} \leftarrow A_{k,l} / \sum_i A_{i,l}, B_{l,k} \leftarrow B_{l,k} * \sum_i A_{i,l}$ ;
   $\mathbf{B} \leftarrow \mathbf{B} \odot (\mathbf{A}^T * (\mathbf{C} \oslash (\mathbf{A} * \mathbf{B} * \mathbf{D})) * \mathbf{D}^T)$ ;
   $n \leftarrow n + 1$ ;
end

```

Algorithm 5.1: Pseudo code for asymmetric NMTF $\mathbf{C} \approx \mathbf{ABD}$.

With the zero-locking property of the multiplicative update as is presented in section 2.1.1 of Chapter 2, \mathbf{B} will keep this structure during subsequent iterations. Each column of \mathbf{A} (or \mathbf{D}^T) is normalized to unity column-wise to ensure it is an emission matrix of an HMM. An interesting property is $\sum_{m,m'} B_{m,m'} = \sum_{k,k'} C_{k,k'}$ given $\sum_m A_{m,k} = 1$ and $\sum_m D_{k,m} = 1$, so \mathbf{B} can be interpreted as a joint probability. The tri-factorization is actually “embedding” the joint distribution of hidden states (low dimensional space) in \mathbf{B} into the joint distribution of observations (high dimensional space) in \mathbf{C} while keeping their probabilistic rationale [124].

The learning is separated for each contextual dependence τ . For every choice of τ , one inputs $\mathbf{C}^{(\tau)}$ and gets the embedding matrices $\mathbf{A}^{(\tau)}$ and $\mathbf{D}^{(\tau)}$ and the co-occurrence matrix of hidden states $\mathbf{B}^{(\tau)}$. The learned embedding matrices are also utilized separately when generating new representations of the utterance in section 5.3.

Symmetric tri-factorization

With multiple τ 's, the learning algorithm of \mathbf{A} and $\mathbf{B}^{(\tau)}$ from $\mathbf{C}^{(\tau)}$ is shown in Table 5.2 for solving Eq.(5.6). The derivation and proof of convergence of the algorithm are in Appendix C.

The zero locking property is still valid in this symmetric version of NMTF. With the sum-to-unity constraints of the columns of \mathbf{A} , it is straightforward to see that $\sum_{i,j} C_{i,j}^{(\tau)} = \sum_{k,l} B_{k,l}^{(\tau)}$. That is the total count of co-occurrences of

Gaussians in $\mathbf{C}^{(\tau)}$ with lag τ is the same as the total count of co-occurrences of the hidden units in $\mathbf{B}^{(\tau)}$. This property is useful for deriving the scale-keeping algorithm of HMM training in 6.3.1.

```

input :  $\{\mathbf{C}^{(\tau)}\}$  and initial estimates for  $\mathbf{A}$  and  $\{\mathbf{B}^{(\tau)}\}$ 
output:  $\mathbf{A}, \{\mathbf{B}^{(\tau)}\}$ 

 $n = 0$ ;
while  $n < \# \text{ iterations}$  do
     $\mathbf{P}^{(\tau)} \leftarrow \mathbf{1}_{M \times 1} * \sum_i (\mathbf{A} * (\mathbf{B}^{(\tau)} + (\mathbf{B}^{(\tau)})^T))_{i,:}$ ;
     $\mathbf{Q}^{(\tau)} \leftarrow \mathbf{C}^{(\tau)} \oslash (\mathbf{A} * \mathbf{B}^{(\tau)} * \mathbf{A}^T)$ ;
     $\mathbf{A} \leftarrow \mathbf{A} \odot (\sum_{\tau} \mathbf{Q}^{(\tau)} * \mathbf{A} * (\mathbf{B}^{(\tau)})^T + (\mathbf{Q}^{(\tau)})^T * \mathbf{A} * \mathbf{B}^{(\tau)}) \oslash (\sum_{\tau} \mathbf{P}^{(\tau)})$ ;
     $A_{i,k} \leftarrow A_{i,k} / \sum_{i'} A_{i',k}$ ,  $B_{k,l}^{(\tau)} \leftarrow \sum_{i'} A_{i',k} * B_{k,l}^{(\tau)} * \sum_{i'} A_{i,l}$ ;
     $\mathbf{B}^{(\tau)} \leftarrow \mathbf{B}^{(\tau)} \odot (\mathbf{A}^T * (\mathbf{C}^{(\tau)} \oslash (\mathbf{A} * \mathbf{B}^{(\tau)} * \mathbf{A}^T)) * \mathbf{A})$ ;
     $n \leftarrow n + 1$ ;
end

```

Algorithm 5.2: Pseudo code for symmetric NMTF $\mathbf{C}^{(\tau)} \approx \mathbf{A}\mathbf{B}^{(\tau)}\mathbf{A}^T$ with multiple contextual dependencies τ .

As illustrated in Figure 5.3, the supervised/unsupervised learning of sub-word units is finished till now. Next, we explain methods for speech representation using the obtained sub-word units.

5.3 Speech Representation Using Sub-word Units

In this section, we present how to represent the training and testing utterances in bag of co-occurrences of the learned hidden units. The first one is a matrix factorization approach directly yielding co-occurrences of hidden units. The second one consists of first obtaining a posteriorgram of hidden units by sequential labeling of the Gaussian posteriorgram and then to computing the bag of co-occurrences from the newly yielded posteriorgram.

5.3.1 Matrix Embedding for Dimension Reduction

With a fixed τ , an utterance is firstly represented by its histogram of co-occurrences of Gaussians, $\mathbf{V}_n^{(\tau)}$, which is reshaped to a M by M matrix named $\hat{\mathbf{C}}_n^{(\tau)}$. Then by using the obtained embedding matrices $\mathbf{A}^{(\tau)}$ and $\mathbf{D}^{(\tau)}$ of lag τ ,

the co-occurrences of states $\hat{\mathbf{B}}_n^{(\tau)}$ is estimated by solving,

$$\operatorname{argmin}_{\hat{\mathbf{B}}_n^{(\tau)}} \operatorname{KLD}(\hat{\mathbf{C}}_n^{(\tau)} \| \mathbf{A}^{(\tau)} \hat{\mathbf{B}}_n^{(\tau)} \mathbf{D}^{(\tau)}) \quad (5.7)$$

Here only $\hat{\mathbf{B}}_n^{(\tau)}$ is going to be updated with the algorithm of Table 5.1. Then $\hat{\mathbf{B}}_n^{(\tau)}$ is reshaped to a column vector as the representation of the utterance, which subsequently forms a column of the new data matrix $\mathbf{E}^{(\tau)}$ with lag τ , i.e. $\mathbf{E}_n^{(\tau)}$. The process is illustrated in Figure 5.6, which is denoted by Track A.5 and Track A.6 in Figure 5.3.

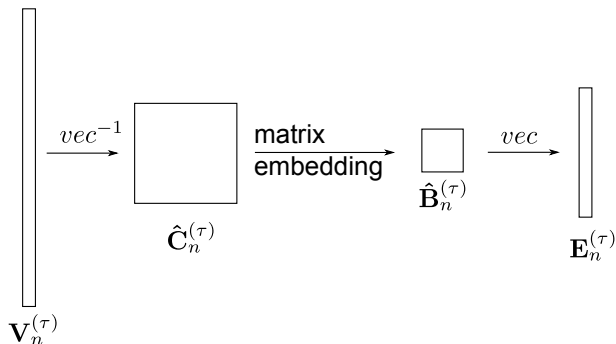


Figure 5.6: Transformation of the co-occurrences of Gaussians in utterance n with lag τ , $\mathbf{V}_n^{(\tau)}$, to co-occurrences of hidden units in $\mathbf{E}_n^{(\tau)}$. vec is the operator to reshape a matrix to a vector by stacking the columns while vec^{-1} is its inverse.

The new data matrix \mathbf{E} is finally the stacking of the matrices $\mathbf{E}^{(\tau)}$ from multiple lags τ . An utterance is always represented as a vector in this method. For a fixed τ , the dimension of the speech representation is reduced from $(\# \text{ Gaussians})^2$ in $\mathbf{V}^{(\tau)}$ to $(\# \text{ hidden units})^2$ in $\mathbf{E}^{(\tau)}$. With the learned embedding matrices, the optimization problem to estimate $\hat{\mathbf{B}}_n^{(\tau)}$ is in fact convex. Hence the complexity is relatively low. However, we should be aware that the mixing of vocabulary patterns could cause confusion especially in long utterances and when some Gaussians are shared by different vocabulary patterns. In our experiments, good results were only obtained on the oracle Gaussians from a supervised HMM where the number of Gaussians is much larger than the number of words.

5.3.2 Sequential Labeling

In this section, we use the symmetric NMTF with constraints $\mathbf{A} = \mathbf{A}^{(\tau)} = (\mathbf{D}^{(\tau)})^T (\forall \tau)$ to model \mathbf{A} as the emission matrix of an HMM. So at the end of the learning procedure, one obtains the directed graphical model between Gaussians and hidden states of Figure 5.5. Sequential labeling of the training and testing utterances are performed by using this graphical model [106]. For the frame at time stamp t , we consider three probability contributions for the hidden units: observation, forward transition and backward transition as illustrated in Figure 5.7 and as denoted by Track B.5 in Figure 5.3.

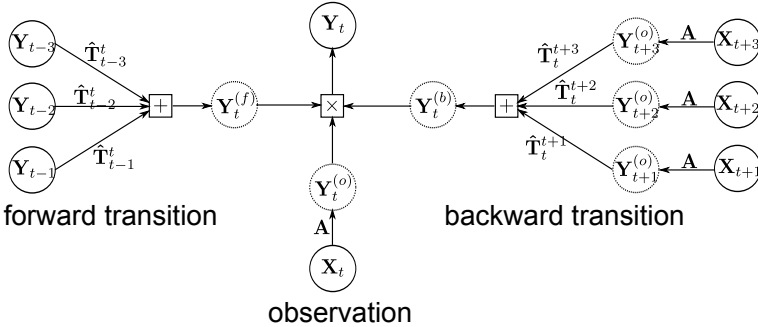


Figure 5.7: Sequential labeling using hidden units learned by NMTF. The frames of an utterance is labeled from left to right. Three types of labeling scores are merged: the forward transition, the observation and the backward transition.

Suppose the Gaussian posteriorgram representation of an utterance is $\{\mathbf{X}_t, t = 1 : T\}$ where $(\mathbf{X}_t)_i = \Pr(x_t = \mathcal{G}_i)$. When zooming into the local frames, the tri-factorization becomes,

$$\begin{aligned}
 & \Pr(x_t = \mathcal{G}_i, x_{t+\tau} = \mathcal{G}_j) \\
 = & \sum_{k,l} \Pr(x_t = \mathcal{G}_i | y_t = S_k) \Pr(y_t = S_k, y_{t+\tau} = S_l) \Pr(x_{t+\tau} = \mathcal{G}_j | y_{t+\tau} = S_l)
 \end{aligned}
 \tag{5.8}$$

where, y_t and $y_{t+\tau}$ are the random variables which assume the state indicator values at frame t and $t + \tau$. Hence with ℓ_1 normalization of the probabilities of every frame, $\{\mathbf{X}_t, t = 1 : T\}$ is the posteriorgram of the utterance labeled by the hidden units. $\Pr(x_t = \mathcal{G}_i, x_{t+\tau} = \mathcal{G}_j)$ and $\Pr(y_t = S_k, y_{t+\tau} = S_l)$ are joint probabilities of Gaussians and the hidden units respectively. So with the

tri-factorization, the Gaussian posteriorgram $\{\mathbf{X}_t, t = 1, \dots, T\}$ is transformed into the posteriorgram of hidden units $\{\mathbf{Y}_t, t = 1, \dots, T\}$ as described below.

The first probability estimate for the hidden units, $\mathbf{Y}_t^{(o)}$, comes from the observation at this time stamp \mathbf{X}_t which is a vector with Gaussian posterior probabilities. $\Pr(x_t|y_t^{(o)} = S_k)$ in Eq.(5.9) is a column of the matrix \mathbf{A} corresponding to state S_k . So $\mathbf{Y}_t^{(o)}$ can be estimated from the NMF in Eq.(5.9).

$$\begin{aligned} & \operatorname{argmin}_{\mathbf{Y}_t^{(o)}} \operatorname{KLD}(\mathbf{X}_t || \hat{\mathbf{X}}_t) \\ &= \operatorname{argmin}_{\mathbf{Y}_t^{(o)}} \operatorname{KLD}(\mathbf{X}_t || \mathbf{A}\mathbf{Y}_t^{(o)}) \end{aligned} \quad (5.9)$$

where $\hat{\mathbf{X}}_t$ is the reconstruction of \mathbf{X}_t with elements $\sum_k \Pr(x_t = \mathbf{G}_m | y_t^{(o)} = S_k) \Pr(y_t^{(o)} = S_k)$, i.e. $\sum_k A_{m,k} Y_{k,t}^{(o)}$.

The second estimate of the probability on the hidden units is from the forward transition as is shown in Eq.(5.10).

$$(\mathbf{Y}_t^{(f,\tau)})^T = (\mathbf{Y}_{t-\tau})^T \hat{\mathbf{T}}_{t-\tau}^t \quad (5.10)$$

where $\hat{\mathbf{T}}_{t-\tau}^t$ is the local transition matrix from frame $t - \tau$ to frame t and $\mathbf{Y}_{t-\tau}$ is the probability vector of hidden units at time stamp $t - \tau$. For each frame t , $\hat{\mathbf{T}}_{t-\tau}^t$ is estimated from the local co-occurrences of Gaussians, $\hat{\mathbf{C}}_{t-\tau}^t$, by the factorization,

$$\operatorname{argmin}_{\hat{\mathbf{T}}_{t-\tau}^t} \operatorname{KLD}(\hat{\mathbf{C}}_{t-\tau}^t || \mathbf{A} \hat{\mathbf{T}}_{t-\tau}^t \mathbf{A}^T) \quad (5.11)$$

$\hat{\mathbf{C}}_{t-\tau}^t$ is constructed from local information: the Gaussian co-occurrence matrix between $\mathbf{X}_{t-\tau}$ and \mathbf{X}_t , i.e. $\mathbf{X}_t \mathbf{X}_{t-\tau}^T$. Subsequently, the estimates from the forward transition with different τ 's are summed and normalized in Eq.(5.12) to obtain its final estimate. The summation here means that the transition between hidden units is not strictly Markovian because the emission process of a state depends on its at most τ -nearest neighbor states.

$$\mathbf{Y}_t^{(f)} = \frac{1}{\tau_0} \sum_{\tau=1}^{\tau_0} \mathbf{Y}_t^{(f,\tau)} \quad (5.12)$$

The third estimate of the probabilities on the hidden units is from the backward transition which is computed from Eq.(5.13). $\mathbf{Y}_{t+\tau}^{(o)}$ and $\hat{\mathbf{T}}_t^{t+\tau}$ are computed

similarly as above by just using the information with the respective time stamps.

$$\operatorname{argmin}_{\mathbf{Y}_t^{(b,\tau)}} \operatorname{KLD}((\mathbf{Y}_{t+\tau}^{(o)})^T \| (\mathbf{Y}_t^{(b,\tau)})^T \hat{\mathbf{T}}_t^{t+\tau}) \tag{5.13}$$

Then the estimates from the backward transition with different τ 's are summed and normalized similarly in Eq.(5.14).

$$\mathbf{Y}_t^{(b)} = \frac{1}{\tau_0} \sum_{\tau=1}^{\tau_0} \mathbf{Y}_t^{(b,\tau)} \tag{5.14}$$

The estimate of \mathbf{Y}_t is the product of the probabilities from observation, forward transition and backward transition in Eq.(5.15). The product implies that the activated hidden unit S_k of frame t should be both observable at this frame and be reachable from its τ -nearest neighbors.

$$\mathbf{Y}_t = \frac{\mathbf{Y}_t^{(o)} \odot \mathbf{Y}_t^{(f)} \odot \mathbf{Y}_t^{(b)}}{\|\mathbf{Y}_t^{(o)} \odot \mathbf{Y}_t^{(f)} \odot \mathbf{Y}_t^{(b)}\|_1} \tag{5.15}$$

To represent long contextual dependencies, a *long-patch* method is utilized to define the co-occurrences of units when constructing the data matrix. Take the hidden-unit-posteriorgram \mathbf{Y}_t as an example. A patch of length $T_0=10$ of the posteriorgram is the sum of the probabilities of units of the frames it contains:

$$\mathbf{Y}_t = \sum_{t'=t}^{t+T_0} \mathbf{Y}_{t'}. \tag{5.16}$$

Finally, a representation of each utterance in terms of a bag of co-occurrences of hidden units is calculated in the same way as the bag of Gaussian co-occurrences in section 3.3.2. Hence the training and testing utterances are represented in their corresponding data matrix \mathbf{E} and \mathbf{E}' respectively, which is denoted by Track B.6 in Figure 5.3.

The method computes a posteriorgram representation where the sequential information is retained. The labeling of each frame merges the observation of the current frame, the forward and backward transitions, which is similar to the rule of forward-backward method of HMM training. Note that we did not use the transition matrix obtained in NMTF, but instead estimated it locally. With

the frame-by-frame labeling, much more computations are required than the matrix embedding proposed in the last subsection. If only taking lag $\tau = 1$, the above learning method can be applied to the HMM training. We will present how to use it to improve the unsupervised HMM training in the next chapter.

In this section, we proposed two methods to yield the new representations of speech using co-occurrences of the hidden units discovered in section 5.2. The corresponding track notes are [A.5] and [A.6] for the supervised track and [B.5] and [B.6] for the unsupervised track. At the end, both tracks yield a new data matrix $\mathbf{E}^{(\tau)}$ where τ is the lag parameter in a co-occurrence of hidden units. Therefore, like in Chapter 2, together with the grounding matrix \mathbf{G} , the training model is given in Eq.(5.17).

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{E} \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{(g)} \\ \mathbf{F} \end{bmatrix} \mathbf{Z} \quad (5.17)$$

where \mathbf{E} is a matrix consists of multiple $E^{(\tau)}$. The process is denoted by [Track A.7] and [Track B.7] in Figure 5.3.

Let \mathbf{E}' denote the new data matrix for the testing data. By solving $\operatorname{argmin}_{\mathbf{Z}'} \operatorname{KLD}(\mathbf{E}' || \mathbf{F}\mathbf{Z}')$ and computing the activations of the digits by $\hat{\mathbf{G}}' = \mathbf{F}^{(g)}\mathbf{Z}'$, the unordered word error rates can be computed as in Chapter 2.

5.4 Experiments on Vocabulary Acquisition

Two groups of experiments have been conducted on TIDIGITS to evaluate the obtained hidden units. The first group follows [Track A] in Figure 5.3 where we learn hidden units from the NMF bigram model of Gaussians developed in Chapter 2 and compare it with the NMF baseline and the HMM baseline and show that given a good Gaussian set the NMF model approaches state-of-the-art HMM results. The second group follows [Track B] in Figure 5.3 where the learning of hidden units works in a completely unsupervised way by using blindly clustered Gaussians. Additionally, we show the advantage of sub-word units in the sense that it can learn from fewer labeled examples.

5.4.1 Experiments with Matrix Embedding

In this section, we compare the performance of matrix embedding with a HMM that is trained with supervision. We want to focus on learning the hidden state

structure and therefore assume the Gaussian set is given. To achieve state-of-the-art results, we copy the Gaussian set from a HMM that is learned with supervision. Experiments with blindly clustered Gaussians will follow in the next section.

For the experiments on the oracle Gaussians, we use the NMTF of section 5.2.3 to learn hidden units and the matrix embedding of section 5.3.1 to obtain a representation of speech in terms of a bag of co-occurrences of hidden units.

Parameters and results

For the training and testing utterances of TIDIGITS, the window length for spectral analysis was 25ms and the frame shift was 10ms. The MFCC extraction used 30 Mel-filter banks from which 12 MFCC coefficients are computed plus the frame’s log-energy. The three vectors of MFCC, Δ and $\Delta\Delta$ were concatenated to a 39-dimensional feature vector. A HMM was trained supervisedly using HTK [125], where 16 states with 20 Gaussians each are used to model every digit, while silence is modeled with 3 states with 36 Gaussians each. There is no sharing of Gaussians across HMM states, yielding $M = 3628$ Gaussians. The same set of Gaussians was used to create the Gaussian posteriorgrams \mathbf{X}_t for use with the NMTF learning framework.

For each frame, the top K_1 and K_2 Gaussians with the highest posterior probability were retained when constructing $\mathbf{V}_n^{(\tau)}$ in Eq.(5.3) (Track A.1 in Figure 5.3) and when computing the matrix embedding estimate $\hat{\mathbf{C}}_n^{(\tau)}$ in Eq.(5.7) (Track A.5 in Figure 5.3) respectively. The lags were chosen as $\tau = 2$, $\tau = 5$ and $\tau = 9$, so the time spacing of the frame-pairs were 20, 50 and 90 ms respectively, which represent contextual dependence at different time scales. L is the number of hidden units per digit (i.e. per column of \mathbf{W}) to be discovered by the tri-factorization (Track A.3 in Figure 5.3). $R_1=12$ and $R_2=12$ were the factorization dimension when in Eq.(5.3) (Track A.1 in Figure 5.3) and Eq.(5.17) (Track A.7 in Figure 5.3) respectively. It is required that $\min(R_1, R_2) \geq 11$ to ensure enough model complexity for the 11 digits.

The unordered word error rates (UWER) are shown in Table 5.1. The first and third row show the baseline performance using direct modeling of Gaussian co-occurrence statistics (i.e. without hidden units) as outlined in Chapter 3. The other rows use the matrix embedding method of section 5.3.1 with different design choices. The 7th column specifies the embedding model, i.e. if the symmetric or the asymmetric NMTF of section 5.2.3 is used in (Track A.3 in Figure 5.3). The 6th column specifies the constraints on the co-occurrence matrix $\mathbf{B}^{(\tau)}$ of hidden units. In “rand”, there are no constraints (with random

initialization); in “diag”, $\mathbf{B}^{(\tau)}$ is upper diagonal with band width τ . The unordered word error rate of the HMM system is 0.15%. For reference, the conventional word error rate of the HMM is 0.25%. Note that the HMM result is obtained with a different recognition paradigm: frame level Viterbi decoding instead of the utterance level co-occurrence count based detection approach.

Table 5.1: Comparison of the oracle Gaussians and the learned hidden units by NMTF on word learning with the grounded NMF learning.

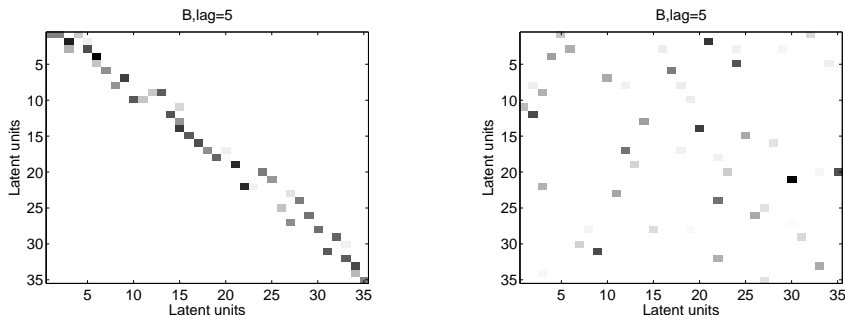
Model (co-occ. of)	K_1	K_2	τ	L	init. B	NMTF	UWER (%)
Gaussians	3	-	[2,5,9]	-	-	-	0.73
hidden units	3	3	[2,5,9]	35	rand	asymm.	1.82
Gaussians	5	-	[2,5,9]	-	-	-	0.58 ⁴
hidden units	5	3	[2,5,9]	35	rand	asymm.	0.44
hidden units	5	5	[2,5,9]	35	rand	asymm.	0.44
hidden units	5	3	[2,5,9]	50	rand	asymm.	0.47
hidden units	5	3	[2,5,9]	35	diag.	asymm.	0.47
hidden units	5	5	[2,5,9]	35	diag.	symm.	0.44
hidden units	5	5	[2,5,9]	20	diag.	asymm.	0.55

The representation of hidden units always performs better than the representation using Gaussians model for $K_1=5$. But embedding matrices with $K_1=3$ fail to improve the corresponding baseline. A sufficient number of Gaussians need to be retained per frame when learning the co-occurrence statistics in \mathbf{V} and \mathbf{W} to obtain the correlation relations between Gaussians and hidden units (or potential HMM states). The performance of the proposed model is robust to the number of hidden units per digit, L , and the number of Gaussians retained per frame in the stage of matrix embedding in Eq.(5.7), K_2 . But note that too few hidden units per column of \mathbf{W} would degrade the model’s performance as shown in the case when $L = 20$ in Table 5.1.

Visualization

Experiments to initialize $\mathbf{B}^{(\tau)}$ with upper-band-diagonal structures were also conducted. In the experiments, $\mathbf{B}^{(\tau)}$ was arranged as (from,to), which implies the lower triangle to be zero. The upper triangle should only have co-occurrences of nearby states. Thus the band width of the upper diagonal

⁴This result was obtained with selection of Gaussian co-occurrences in Chapter 2. It is the best result from the bigram model of NMF.



(a) Upper-band-diagonal initialization of $\mathbf{B}^{(\tau)}$ with $\tau = 5$ (b) Random initialization of $\mathbf{B}^{(\tau)}$ with $\tau = 5$

Figure 5.8: The co-occurrences of the hidden units of digit “one”, $R_1=35$ units per digit, from \mathbf{W} with $K_1=5$.

was restricted to τ diagonals. All other elements of $\mathbf{B}^{(\tau)}$ were zero. With the multiplicative updates used to solve NMTF, they remain zero.

Taking digit “one” with $\tau=5$ as an example, the obtained co-occurrence matrices $\mathbf{B}^{(\tau)}$ are shown in Figure 5.8, where Figure 5.9(a) is with upper-band-diagonal initialization and Figure 5.9(b) is with random initialization. In both cases, $\mathbf{B}^{(\tau)}$ is very sparse, showing that the hidden units co-occur sparsely as the HMM model suggests. The upper-band-diagonal and random initialization perform equally (within the experimental accuracy) in Table 5.1.

An example of the matrix tri-factorization is illustrated in Figure 5.9. The Gaussians associated with the digit and with the silence states are highly activated in the co-occurrence matrix. With the factorization, hidden units performing like hidden states and their transitions are discovered.

The upper-band-diagonal initialization selects a permutation of the hidden units by ordering them with “from”-“to” pairs. By analyzing the embedding matrices, we find that each hidden unit usually activates Gaussians of several successive HMM states in Figure 5.9. So the units are indeed related to the HMM states, but there is (not surprisingly) not a one-to-one relation. Since the model can really discover an HMM-like structure, we may use its outputs as initializations to train an HMM or for sequential decoding with its own sequential structure as in Figure 5.7. The details of NMTF learning of hidden units for an intermediate layer can be found in [102].

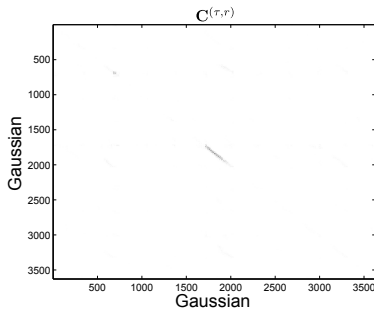
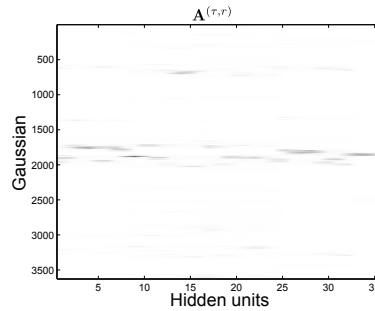
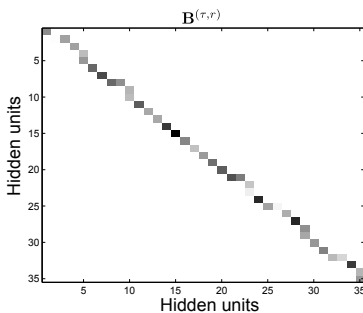
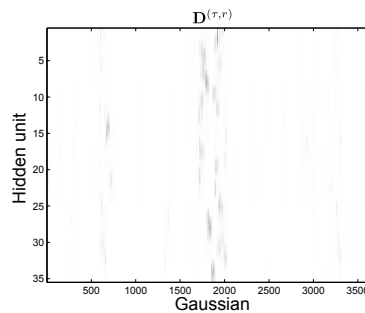
(a) The Gaussian co-occurrences $\mathbf{C}^{(\tau, r)}$ of digit "one"(b) The Gaussian weight matrix $\mathbf{A}^{(\tau, r)}$ of digit "one"(c) The state co-occurrences $\mathbf{B}^{(\tau, r)}$ of digit "one"(d) The Gaussian weight matrix $\mathbf{D}^{(\tau, r)}$ of digit "one"

Figure 5.9: The NMTF learning of HMM states and Gaussian weight matrices with $\tau=2$. $\mathbf{A}^{(\tau, r)}$, $\mathbf{B}^{(\tau, r)}$ and $\mathbf{D}^{(\tau, r)}$ are learned by NMTF: $\mathbf{C}^{(\tau, r)} \approx \mathbf{A}^{(\tau, r)}\mathbf{B}^{(\tau, r)}\mathbf{D}^{(\tau, r)}$. The example corresponds to digit "one".

5.4.2 Experiments with Sequential Labeling and Blindly Clustered Gaussians

In the above section, co-occurrence modeling of learned hidden units from tri-factorization showed better performance than direct single-layer co-occurrence modeling of Gaussians. Those Gaussians were obtained from an HMM learned with supervision. In this section, we show that effective hidden units can be learned without supervision. Therefore, we also switch to a Gaussian set that is also learned without supervision. We will show that the automatic reuse of hidden units allows to learn from fewer labeled data. The process is depicted in Track B of Figure 5.3.

Parameters for learning hidden units and sequential labeling of posteriorgrams

The MFCC feature extraction and database are the same as in section 5.4.1, but now Gaussian mixture of $M=1000$ components was obtained by *unsupervised* training of a Gaussian Mixture Model (GMM) with the EM algorithm from the training data.

The number of recurrent patterns represented in Gaussian co-occurrences, i.e. the number of columns of \mathbf{W} in Eq.(5.4) (Track B.1 in Figure 5.3), is $R_1=25$. The selection of R_1 is inspired from the results of unsupervised NMF learning in Table 4.2 of Chapter 4 where 25 seems to be a good choice at balancing the model complexity and the learning accuracy. The contextual dependency parameter $\tau=1,2,3$ for computing co-occurrences from the posteriorgram \mathbf{X}_t . For each $\mathbf{C}^{(r,\tau)}$ (see Figure 5.4), $L=10$ hidden units are extracted for each column of \mathbf{W} in Track B.3 of Figure 5.3. So there are $L * R_1$ hidden units in total in \mathbf{A} of Track B.4 of Figure 5.3.

With the sequential labeling presented in section 5.3.2, the Gaussian posteriorgram \mathbf{X}_t is transformed into the posteriorgram of hidden units \mathbf{Y}_t in Track B.5 of Figure 5.3. With a new group of lag parameters $\tau = [1, 2, 3]$ for the new posteriorgram \mathbf{Y}_t , the histogram of co-occurrences of hidden units can be calculated as presented in section 3.3.2 and are stored in a new data matrix \mathbf{E} in Eq.(5.18), which is denoted by Track B.6 of Figure 5.3.

Learning words from a few of labeled data

We now compare the two kinds of representations (Gaussian posteriorgram and hidden-unit posteriorgram) by performing weakly supervised spoken pattern discovery. The task is to discover vocabulary patterns when only a fraction of the utterances come with supervisory information. In word learning, a lot of data may have no supervision. For example, in infant vocabulary acquisition, there are many utterances that do not contain a visual counterpart. Also in robot vocabulary acquisition, not all utterances contain examples of what they mean. The learning process can be approximately modeled in the NMF framework through Eq.(5.18).

$$\begin{bmatrix} \mathbf{G}_{:,1:N_1} & 0 \\ \mathbf{E}_{:,1:N_1} & \mathbf{E}_{:,N_1+1:N} \end{bmatrix} \approx \begin{bmatrix} \mathbf{F}^{(g)} \\ \mathbf{F} \end{bmatrix} [\mathbf{Z}_{:,1:N_1} \quad \mathbf{Z}_{:,N_1+1:N}] \quad (5.18)$$

\mathbf{G} is the ground truth matrix as supervision where its entry $G_{s,n}$ shows the frequency of appearance of the ground word s in the utterance n . N_1 is the number of labeled training utterances performing as supervision. During training, we always use all N training utterances, but have an increasing number N_1 of labeled utterances. Cognitively, this process corresponds to an agent or infant hearing a lot of utterances, but only a part of them are interpreted by a teacher, parent or the sensory scene. \mathbf{F} is the pattern matrix, each column of which is a learned vocabulary pattern. $\mathbf{F}^{(g)}$ reflects the associations between the patterns and the ground words. $\mathbf{Z} = [\mathbf{Z}_{:,1:N_1} \ \mathbf{Z}_{:,N_1+1:N}]$ is the coefficient matrix whose columns are the weights of the discovered patterns in the corresponding utterances. The columns of \mathbf{G} , \mathbf{F} and \mathbf{E} are ℓ_1 normalized to balance grounding and acoustics. The evaluation metric is again unordered word error rate calculated in the same way as in Chapter 2.

The representations in bag of co-occurrences of Gaussians are taken as a baseline. One just needs to replace \mathbf{E} in Eq.(5.18) by the bag of co-occurrences of Gaussians as in Chapter 3.

The cost function for computing the approximation in Eq.(5.18) is KLD. The KLD metric is not sensitive to zeros inputs, that is the zero inputs contribute nothing to the total cost. Hence the block 0 in Eq.(5.18) which stands for the unlabeled data has no effect on the optimization: $\text{KLD}(0 \parallel \mathbf{E}^{(g)} \mathbf{Z}_{:,N_1+1:N_2}) = 0$.

Spoken pattern discovery

The common dimension between \mathbf{F} and \mathbf{Z} , i.e. the number of vocabulary patterns, is $R_2=12$ in Eq.(5.18) and in [Track B.7](#) of Figure 5.3. The evaluation metric is unordered word error rate (UWER). This metric can focus the evaluation on the representation of the vocabularies.

As is shown in Figure 5.10, the hidden units perform much better than the Gaussians with only a few labeled utterances ($N_1 \leq 1000$), but not with a sufficiently large number of labeled utterances. This is expected as the parameter sharing in the hidden units lead to smaller data requirement. Apparently, there is enough data eventually to estimate a reliable single layer co-occurrence model. There could also be an effect of lack of fine tuning of the hidden units. As pointed out in [47], a top-down fine tuning with supervision is important to help a multi-layer model beat its single-layer counterparts. Thus in the fine tuning stage, we use the ground truth information to classify the structures extracted in section 2.2 into *digits* and *silence* and model silence with only 3 hidden units. Consistent improvement is observed in Figure 5.11 and

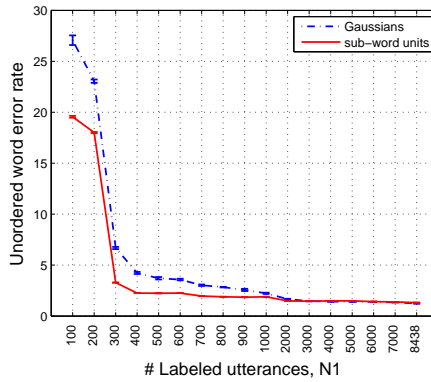


Figure 5.10: The comparison of performance on vocabulary discovery between Gaussians and sub-word units.

Table 5.2. The unordered word error rate 0.8% is the lowest one we obtained using blindly clustered Gaussians till now.

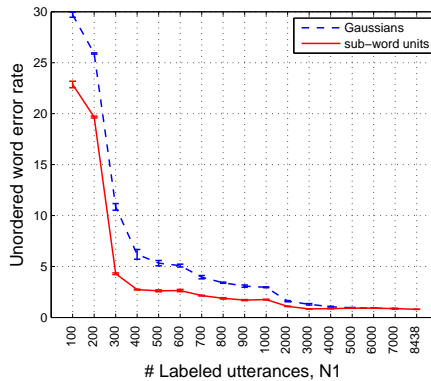


Figure 5.11: The comparison of performance on vocabulary discovery from TIDIGITS between Gaussians and sub-word units. Supervisory information is utilized to identify vocabulary patterns and garbage patterns, based on which 10 sub-word units are learned for the vocabulary patterns and 3 units are learned for the ones related to garbage or silence.

Table 5.2: UWER (%) obtained by using the refined hidden units as features. The baselines from Gaussians are also shown. The overview on all N_1 's can be found in Figure 5.11. This table only shows the results when zooming into the tails of the two plots in Figure 5.11.

N_1	6000	7000	8438
Gaussians	0.95 ± 0.04	0.91 ± 0.00	0.87 ± 0.00
hidden units	0.91 ± 0.00	0.87 ± 0.00	0.80 ± 0.00

Visualization

Figure 5.12 shows the posteriorigram of the utterance “4625” with the extracted hidden units. Piece-wise constant traces or segments are observed from the figure. Some of the hidden units have long durations like 5 to 8 frames which correspond to vowel parts, e.g. the segments of frames between 80 and 95, and the ones between 105 and 120. So it is appropriate to name the hidden units “sub-word units”.

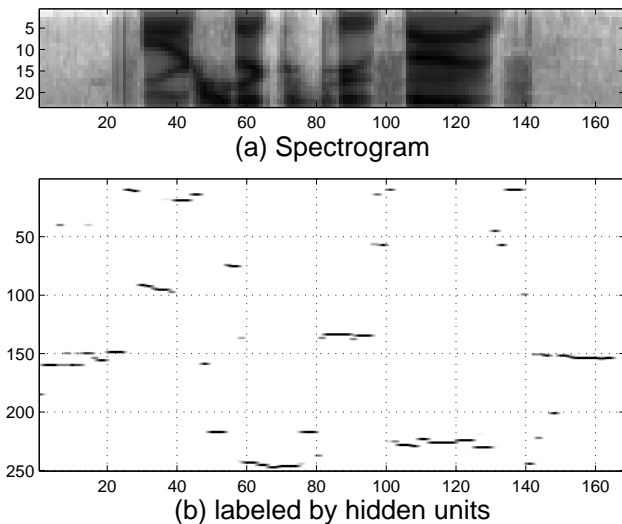
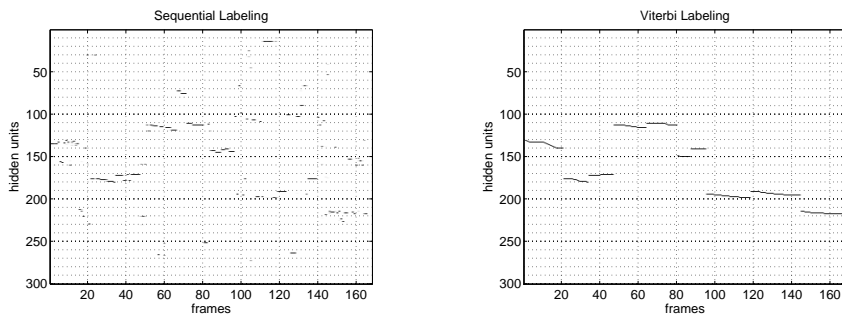


Figure 5.12: Representations of utterance “4625” in posterior probabilities on the learned hidden units. The corresponding spectrogram is also shown for comparison.



(a) Sequential labeling of utterance “4625” with 300 hidden units

(b) Viterbi labeling of utterance “4625” with 300 hidden units

Figure 5.13: Posteriorgrams on learned hidden units for comparing the proposed sequential labeling and the Viterbi labeling.

With $\tau = 1$, the NMTF actually learns a discrete density HMM with emission matrix \mathbf{A} and transition matrix $\mathbf{B}^{(\tau=1)}$. With the emission matrix and Eq.(5.9), the posterior probabilities of the frame \mathbf{X}_t on the hidden states are stored in a vector $\mathbf{Y}_t^{(o)}$. With the new posteriorgram $\{\mathbf{Y}_t^{(o)}, t = 1, \dots, T\}$ and the transition matrix $\mathbf{B}^{(\tau=1)}$, the Viterbi algorithm [118] can find an optimal path to align the observation frames and the hidden states. The optimal path is a state sequence or a winner-take-all posteriorgram. With the Viterbi alignment, further smooth segments are observed as is shown in Figure 5.13. However, due to the 1-best path generated from the Viterbi alignment, the performance in terms of unordered word error rate is not as good as what is obtained with the proposed sequential labeling approach, and even worse than the one using the original Gaussian co-occurrences, as is shown in Figure 5.14.

It is worth to note that the proposed co-occurrence model and labeling approach only consider the nearest neighbors of a data frame, making it a generic tool to extract patterns or topics from images, i.e. it is not limited to one dimensional data.

5.5 Conclusion

We have successfully extended the NMF-based vocabulary acquisition from co-occurrence statistics to now include a layer of sub-word units that are learned using a matrix tri-factorization. The vocabulary patterns discovered by NMF are in a form of *bag of sub-word units* where every sub-word unit itself is in a BoF format. Hence, a word is actually represented in *bag of BoFs*. The

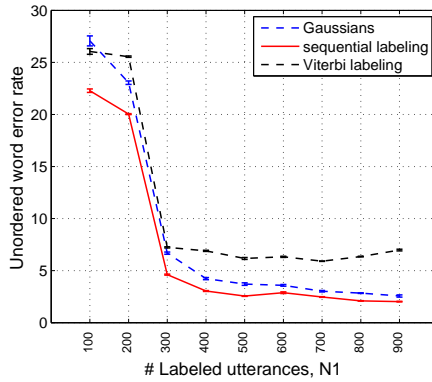


Figure 5.14: Comparison of performance on vocabulary discovery between Gaussians, sequential labeling and Viterbi labeling of hidden units.

method performs better at the acquisition of small vocabularies. With a group of Gaussians from a HMM trained with transcription, the hidden units achieved the best NMF-based result ever. On the experiments with blindly clustered Gaussians, significant improvements were observed on the representation using hidden units over those using Gaussians directly.

The next steps are now to apply the proposed learning scheme to improve the unsupervised training of HMM and to give the NMF-learned vocabulary patterns sequential properties such that they can subsequently be applied as a word recognizer.

Chapter 6

Unsupervised Training of HMMs Using Non-negative Tucker Decomposition

In Chapter 5, we proposed methods to learn sub-word units by using non-negative matrix tri-factorization (NMTF). Lower unordered word error rates were obtained from the speech representation of co-occurrences of sub-word units than from the representation using Gaussian co-occurrences. The discovered sub-word units have shown strong connections to HMM states, while the links between sub-word units and Gaussians are similar to the emission matrix of the HMM. These findings inspire us to train HMMs by using matrix tri-factorization.

In this chapter, three kinds of HMMs: discrete density HMM (DDHMM), semi-continuous density HMM (SCDHMM) and Kullback-Leibler divergence based HMM (KLDHMM), are firstly constructed for sequential pattern discovery, such as vocabulary discovery from speech in this thesis. The representations of co-occurrence statistics are extended to a non-negative tensor which is decomposed to estimate the HMM parameters. This operation is called non-negative Tucker decomposition (NTD) and is computed by combining the NMF model in Chapter 4 and the NMTF model in Chapter 5. Joint training methods between NTD and HMMs are presented subsequently. Experiments of vocabulary discovery on TIDIGITS are finally reported.

6.1 Unsupervised Training of HMMs for Sequential Pattern Discovery

HMMs are good at modeling sequential data and have thus been applied to a lot of tasks of sequential data processing, such as automatic speech recognition (ASR) [87], topic detection and segmentation [7], handwriting recognition [51] and gene analysis [85]. In ASR, the data to be analyzed by HMM is in form of observation sequences,

$$\mathbf{O}_1, \dots, \mathbf{O}_{T_n} \quad (6.1)$$

where \mathbf{O}_t is a frame characterized by MFCC+ Δ + $\Delta\Delta$, T_n is the number of frames in the utterance n . In Chapter 5, we discussed the Gaussian posteriorgram representation of an utterance,

$$\mathbf{X}_1, \dots, \mathbf{X}_{T_n} \quad (6.2)$$

where \mathbf{X}_t are the posterior probabilities of frame \mathbf{O}_t on the Gaussians. Since \mathbf{X}_t is a function of \mathbf{O}_t , we describe the following general configuration and training framework of HMMs by using the symbols of original observations \mathbf{O}_t .

6.1.1 HMM Topology and Acoustic Models

An HMM for sequential pattern discovery and recognition on the sequences $\{\mathbf{O}_1, \dots, \mathbf{O}_{T_n}\}$ where $1 \leq n \leq N$ and N is the total number of sequences, is configured by connecting a number of left-to-right sub-HMMs with non-emitting beginning and end states as is shown in Figure 6.1. Initially, we will consider that each pattern is modeled by one sub-HMM. However, in view of pronunciation variation modeling in speech, it is conceivable that multiple sub-HMMs are used to model a single word.

The HMM is characterized by the following elements [87].

- The hidden states of the r -th sub-HMM $\{S_{r,1}, \dots, S_{r,L_r}\}$ where L_r is the number of states in the sub-HMM of some sequential pattern. So $K = \sum_{r=1}^R L_r$ is the total number of hidden states where R is the number of sub-HMMs. Without any prior knowledge about the sequential patterns, L_r is selected as a constant L in this section.
- The sequence of hidden states $\{Q_1, Q_2, \dots, Q_{T_n}\}$ which are aligned to $\{\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_{T_n}\}$, the n -th training sequence with length T_n .

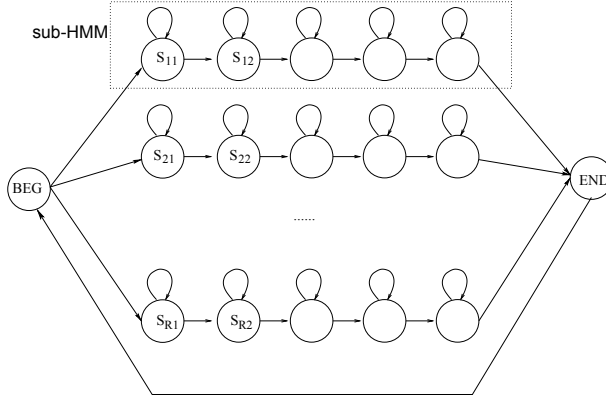


Figure 6.1: The HMM topology used for sequential pattern discovery. Each parallel branch is called a sub-HMM.

- The transition matrix $\mathbf{T}_{K \times K}$ whose element $T_{k,k'}$ is the conditional probability of transition from S_k to $S_{k'}$: $\Pr(Q_{t+1} = S_{k'} | Q_t = S_k), \forall t$. For the HMM in Figure 6.1, \mathbf{T} has a special structure with only non-zero probabilities on the diagonal and sub-diagonal locations $\{T_{(r-1)L+l', (r-1)L+l}, T_{(r-1)L+l, (r-1)L+l+1}\}$ and cross-pattern locations $\{T_{rL, (r'-1)L+1}\}$, where $r, r' = 1, \dots, R$ and $l = 1, \dots, L - 1$ and $l = 1, \dots, L$.
- The initial state distribution $\boldsymbol{\pi}$ where $\pi_k = \Pr(Q_1 = S_k)$. Only the initial states of the patterns, $\{\pi_{(r-1)L+1}, r = 1, \dots, R\}$, can have non-zero probabilities in $\boldsymbol{\pi}$.
- An emission model $a_k(\mathbf{O}_t)$, also called acoustic model, to measure the similarity between the hidden state S_k and the observation \mathbf{O}_t .

To summarize the above description, the HMM is described by a group of parameters $\boldsymbol{\Lambda} = \{a_k(\cdot), \mathbf{T}, \boldsymbol{\pi}\}$. The parameters are non-negative and satisfy $\sum_{k'} T_{k,k'} = 1$ and $\sum_k \pi_k = 1$. According to the configuration of the acoustic model $a_k(\mathbf{O}_t)$, we discuss three types of HMMs as follows.

Discrete density

In discrete density HMM (DDHMM), the frame \mathbf{X}_t of the posteriorgram is quantized into a discrete symbol denoted by X_t . In the case of Gaussian posteriorgrams, the observation alphabet set is just the collection of Gaussians

$\{\mathcal{G}_1, \dots, \mathcal{G}_M\}$ where M is the number of Gaussians, the acoustic model is therefore parameterized by the emission matrix $\mathbf{A}_{M \times K}$ whose element $A_{m,k}$ is the conditional probability of observation \mathcal{G}_m given state S_k : $A_{m,k} = \Pr(X_t = \mathcal{G}_m | Q_t = S_k)$, $\forall t$. At each time t , a Gaussian with index X_t is selected with probability $A_{X_t,k}$, hence

$$a_k(X_t) = A_{X_t,k} \quad (6.3)$$

So the parameters of DDHMM are $\mathbf{\Lambda} = \{\mathbf{A}, \mathbf{T}, \boldsymbol{\pi}\}$

Semi-continuous density

In semi-continuous density HMM (SCDHMM), a Gaussian mixture model (GMM) is utilized to compute the likelihood of observation \mathbf{O}_t given state S_k ,

$$a_k(\mathbf{O}_t) = \sum_m A_{m,k} \mathcal{G}(\mathbf{O}_t; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \quad (6.4)$$

where the matrix \mathbf{A} is the Gaussian weight matrix of the states, $\boldsymbol{\mu}_m$ is the mean of the m -th Gaussian and $\boldsymbol{\Sigma}_m$ is the covariance matrix of the m -th Gaussian which is diagonal here. The Gaussians are shared across the hidden states.

Kullback-Leibler divergence

The Kullback-Leiber divergence HMM (KLDHMM) is based on the posteriorgram representations [2]. A state S_k is characterized by a multinomial distribution on the Gaussians: $\mathbf{A}_k = [A_{1,k}, \dots, A_{M,k}]^T$. The negative KLD between a vector of posteriors \mathbf{X}_t (a distribution to be tested) and a hidden state (the reference distribution) is taken as the *log likelihood* as is shown in Eq.(6.5).

$$\log(a_k(\mathbf{X}_t)) = - \sum_m X_{m,t} \log \frac{X_{m,t}}{A_{m,k}} \quad (6.5)$$

The KLD in Eq.(6.5) is a non-symmetric measure of the difference between the two multinomial distributions, where it measures the expected number of extra bits required to code samples from \mathbf{X}_t when using a code based on \mathbf{A}_k , rather

than using a code based on \mathbf{X}_t . The observation likelihood is thus,

$$\begin{aligned} a_k(\mathbf{X}_t) &= \exp\left(-\sum_m X_{m,t} \log \frac{X_{m,t}}{A_{m,k}}\right) \\ &= \prod_m \exp\left(-X_{m,t} \log \frac{X_{m,t}}{A_{m,k}}\right) \\ &= \prod_m \left(\frac{A_{m,k}}{X_{m,t}}\right)^{X_{m,t}} \end{aligned} \quad (6.6)$$

6.1.2 Training Algorithms

The expectation maximization (EM) method is usually applied to estimate the unknown parameters of an HMM which maximizes the likelihood of the data. Often, the sequential data $\mathbf{O}^{(n)}$ comes with its sequential labels $\mathbf{G}^{(n)}$, resulting in a *supervised* training problem. For instance, in ASR nowadays, the training data is labeled in terms of the words from which an HMM representation for each word label is learned from the data.

$$\sum_{n=1}^N \log \Pr(\mathbf{O}^{(n)}, \mathbf{G}^{(n)} | \Lambda) \quad (6.7)$$

However, in the task of vocabulary acquisition the labels may not be provided or only a few labels are provided. Without loss of generality, we assume the first N_1 sequences are labeled where $0 \leq N_1 \leq N$. The HMM training is therefore *unsupervised* in Eq.(6.8),

$$\sum_{n=1}^N \log \Pr(\mathbf{O}^{(n)} | \Lambda) \quad (6.8)$$

or *semi-supervised* in Eq.(6.9).

$$\sum_{n=1}^{N_1} \log \Pr(\mathbf{O}^{(n)}, \mathbf{G}^{(n)} | \Lambda) + \sum_{n=N_1+1}^N \log \Pr(\mathbf{O}^{(n)} | \Lambda) \quad (6.9)$$

With the assumption that only a few utterances are labeled, i.e. $N_1 \ll N$, the supervised term in Eq.(6.9) contributes little to the total objective function. So we can first apply the unsupervised training to discover sequential patterns and then use (a few) labeled data to explain the patterns. This is also the scenario of unsupervised learning of the bottom layers in deep learning [46][6].

EM training

The EM training of HMMs is based on an auxiliary function of the log likelihood function of Eq.(6.8),

$$\mathcal{A}(\mathbf{\Lambda}, \bar{\mathbf{\Lambda}}) := \sum_n \sum_{\mathbf{Q}^{(n)}} \Pr(\mathbf{Q}^{(n)} | \mathbf{O}^{(n)}, \bar{\mathbf{\Lambda}}) \log \frac{\Pr(\mathbf{Q}^{(n)}, \mathbf{O}^{(n)} | \mathbf{\Lambda})}{\Pr(\mathbf{Q}^{(n)} | \mathbf{O}^{(n)}, \bar{\mathbf{\Lambda}})}, \quad (6.10)$$

where $\bar{\mathbf{\Lambda}}$ is the current guess of HMM parameters and $\mathbf{Q}^{(n)}$ is the state sequence aligned to the observation sequence $\mathbf{O}^{(n)}$. With the concavity of $\log(\cdot)$, it is easy to see that,

$$\begin{aligned} \mathcal{A}(\mathbf{\Lambda}, \bar{\mathbf{\Lambda}}) &\leq \sum_n \log \Pr(\mathbf{O}^{(n)} | \mathbf{\Lambda}), \\ \mathcal{A}(\bar{\mathbf{\Lambda}}, \bar{\mathbf{\Lambda}}) &= \sum_n \log \Pr(\mathbf{O}^{(n)} | \bar{\mathbf{\Lambda}}). \end{aligned} \quad (6.11)$$

The unknown parameters only appear in the term $\log \Pr(\mathbf{Q}^{(n)}, \mathbf{O}^{(n)} | \mathbf{\Lambda})$, so we only need to maximize Eq.(6.12) (Q -function) to update $\mathbf{\Lambda}$ from $\bar{\mathbf{\Lambda}}$.

$$\mathcal{Q}(\mathbf{\Lambda}, \bar{\mathbf{\Lambda}}) := \sum_n \sum_{\mathbf{Q}^{(n)}} \Pr(\mathbf{Q}^{(n)} | \mathbf{O}^{(n)}, \bar{\mathbf{\Lambda}}) \log \Pr(\mathbf{Q}^{(n)}, \mathbf{O}^{(n)} | \mathbf{\Lambda}) \quad (6.12)$$

Using the HMM assumptions, we expand:

$$\log \Pr(\mathbf{Q}^{(n)}, \mathbf{O}^{(n)} | \mathbf{\Lambda}) = \log(\pi_{Q_1^{(n)}}) + \sum_{t=1}^{T_n-1} \log(T_{Q_t^{(n)}, Q_{t+1}^{(n)}}) + \sum_{t=1}^{T_n} \log(a_{Q_t^{(n)}}(\mathbf{O}_t)) \quad (6.13)$$

The Q -function is therefore decomposed into three terms. By removing any terms that are irrelevant in the optimization, they are expressed as follows,

$$\mathcal{Q}_\pi(\boldsymbol{\pi}, \bar{\mathbf{\Lambda}}) := \sum_{n=1}^N \sum_{k=1}^K \gamma_1^{(n)}(k) \log(\pi_k) \quad (6.14)$$

$$\mathcal{Q}_T(\mathbf{T}, \bar{\mathbf{\Lambda}}) := \sum_{n=1}^N \sum_{k, k'=1}^K \sum_{t=1}^{T_n-1} \xi_t^{(n)}(k, k') \log(T_{k, k'}) \quad (6.15)$$

$$\mathcal{Q}_A(\mathbf{A}, \bar{\mathbf{\Lambda}}) := \sum_{n=1}^N \sum_{k=1}^K \sum_{t=1}^{T_n} \gamma_t^{(n)}(k) \log(a_k(\mathbf{O}_t)) \quad (6.16)$$

where $\gamma_t^{(n)}(k) = \Pr(Q_t^{(n)} = S_k | \mathbf{O}^{(n)}, \bar{\mathbf{A}})$ is the posterior probability of state S_k for frame $\mathbf{O}_t^{(n)}$ of utterance n and $\xi_t^{(n)}(k, k') = \Pr(Q_t^{(n)} = S_k, Q_{t+1}^{(n)} = S_{k'} | \mathbf{O}^{(n)}, \bar{\mathbf{A}})$ is the co-occurrence probability of state S_k and state $S_{k'}$ from frame $\mathbf{O}_t^{(n)}$ to frame $\mathbf{O}_{t+1}^{(n)}$ in utterance n . Both of them are estimated by using the current estimate of HMM parameters $\bar{\mathbf{A}}$.

The update of the parameters in the acoustic model should be given special attention regarding the three types of HMMS. In DDHMM, Eq.(6.16) becomes,

$$\mathcal{Q}_A(\mathbf{A}, \bar{\mathbf{A}}) := \sum_{n=1}^N \sum_{k=1}^K \sum_{t=1}^{T_n} \gamma_t^{(n)}(k) \delta_{X_t^{(n)}, i} \log(A_{X_t^{(n)}, k}), \quad (6.17)$$

where i is the Gaussian index and δ is the Dirichlet delta.

In SCDHMM, besides the weight matrix \mathbf{A} , the mean $\boldsymbol{\mu}_m$ and covariance matrix $\boldsymbol{\Sigma}_m$ of Gaussians could also be estimated. One may refer to [125] for the details. By using the auxiliary function of the GMM model, the update of \mathbf{A} is converted to optimizing the following Q -function.

$$\mathcal{Q}_A(\mathbf{A}, \bar{\mathbf{A}}) := \sum_{n=1}^N \sum_{k=1}^K \sum_{t=1}^{T_n} \gamma_t^{(n)}(k) \sum_{m=1}^M \frac{\bar{A}_{m,k} \mathcal{G}(\mathbf{O}_t; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{m'} \bar{A}_{m',k} \mathcal{G}(\mathbf{O}_t; \boldsymbol{\mu}_{m'}, \boldsymbol{\Sigma}_{m'})} \log(A_{m,k}), \quad (6.18)$$

where $\mathcal{G}(\mathbf{O}_t; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ is the likelihood of frame \mathbf{O}_t on Gaussian m and $\bar{\mathbf{A}}$ is the previous estimate of \mathbf{A} in $\bar{\mathbf{A}}$.

For KLDHMM with the acoustic model in Eq.(6.5), the objective function to be maximized is,

$$\mathcal{Q}_A(\mathbf{A}, \bar{\mathbf{A}}) := - \sum_{n=1}^N \sum_{k=1}^K \sum_{t=1}^{T_n} \gamma_t^{(n)}(k) \sum_m X_{m,t}^{(n)} \log \frac{X_{m,t}^{(n)}}{A_{m,k}}, \quad (6.19)$$

which is also log-linear by removing the constant term:

$$\mathcal{Q}_A(\mathbf{A}, \bar{\mathbf{A}}) := \sum_{n=1}^N \sum_{k=1}^K \sum_{t=1}^{T_n} \gamma_t^{(n)}(k) \sum_m X_{m,t}^{(n)} \log(A_{m,k}). \quad (6.20)$$

It is interesting to observe that this is an extended form of DDHMM which takes all the posterior probabilities into account rather than making a hard quantization. With the log-linear property, they are easily solved by using the Lagrange multiplier as shown below.

Finally, with the probabilistic constraints, solving $\boldsymbol{\pi}$, a row of \mathbf{T} and a column of \mathbf{A} boils down to the following constrained optimization problem.

$$\begin{aligned} \max_{y_1, \dots, y_K} \quad & \mathcal{L}(y_1, \dots, y_L) = \sum_{k=1}^K \omega_k \log(y_k) \\ \text{s.t.} \quad & \sum_{k=1}^K y_k = 1, y_k \geq 0. \end{aligned} \quad (6.21)$$

where $\mathbf{y} = (y_1, \dots, y_K)$ denotes $\boldsymbol{\pi}$ or a row of \mathbf{T} or a column of \mathbf{A} , ω_k is the term of constants (w.r.t. \mathbf{y}) derived from Eq.(6.14), Eq.(6.15), Eq.(6.17), Eq.(6.18) or Eq.(6.20). The update algorithm is therefore given in Eq.(6.22) by applying the Lagrange multiplier.

$$y_k = \frac{\omega_k}{\sum_{k'} \omega_{k'}} \quad (6.22)$$

Note that one should replace the number of variables K by the number of Gaussians M when updating a column of \mathbf{A} .

When computing $\gamma_t^{(n)}(k)$, if all the state possibilities of a frame are retained, the algorithm is called *Baum-Welch training* (BW); on the other hand, if only one state identity is assigned to each frame, the process is called *Viterbi alignment* and the algorithm regarding the DDHMM is called the *Segmental k-means Algorithm*.

Simulated annealing to jump out of local extrema

The above optimization process iteratively increases the data likelihood. However, the optimization could be trapped at poor local extremes. Local optima in EM training can be addressed with simulated annealing, i.e. perturbing the parameter estimation randomly with a magnitude that decreases with the iteration number [84]. Simulated annealing is claimed to be able to yield the global optimum with probability one given *sufficiently slow* annealing and *enough* iterations. In practice, those conditions are hard to satisfy. In the example given in [84], a small HMM system with 8 states and 9 observation symbols required 60,000 to 400,000 iterations to attain high data likelihoods.

On the other hand, we should notice the target of vocabulary acquisition is not only to find a group of solutions with high data likelihood, but to discover meaningful sequential patterns in the data. So besides the data likelihood other optimization criteria should also be considered as a target.

Joint labeling and HMM parameters

A model to jointly learn grounding labels and HMM parameters (named JLH here) is proposed in [95] where the labels $\mathbf{G}^{(n)}$ of the n -th utterance are also treated as unknown variables in Eq.(6.23).

$$\max_{\mathbf{G}^{(n)}, \Lambda} \sum_{n=1}^N \log \Pr(\mathbf{O}^{(n)}, \mathbf{G}^{(n)} | \Lambda) \quad (6.23)$$

In summary, this method alternates between estimating the HMM Λ given the labels (supervised training in Eq.(6.24)) and estimating the pattern labels $\mathbf{G}^{(n)}$ given the HMM $\bar{\Lambda}$ (recognition in Eq.(6.25)).

$$\bar{\Lambda} = \operatorname{argmax}_{\Lambda} \sum_{n=1}^N \log \Pr(\mathbf{O}^{(n)} | \Lambda, \bar{\mathbf{G}}^{(n)}), \quad (6.24)$$

$$\bar{\mathbf{G}}^{(n)} = \operatorname{argmax}_{\mathbf{G}^{(n)}} \log \Pr(\mathbf{G}^{(n)} | \mathbf{O}^{(n)}, \bar{\Lambda}). \quad (6.25)$$

The training procedures guarantee the non-decrease of the joint objective function. While it is relevant to apply this optimization method to word discovery, the initialization used in [95] is not: it starts from a spectral discontinuity metric to segment the audio into phone-like segments, which would make it hard for JLH to converge to word-sized units. Therefore, careful initialization of this method should be considered before the above optimization procedures.

Using SVD for training DDHMMs

In [50, 97], methods are proposed for learning HMMS by spectral embedding via singular value decomposition (SVD). It is shown that symbol co-occurrence probabilities of discrete-density HMMS show structure that can be recovered using an SVD. This model benefits from the property that SVD is globally convergent, and it is shown that it will perform well asymptotically in prediction tasks. However, on finite sample sizes, it can yield negative HMM probability estimates. While there may be ad hoc solutions to this concern, imposing a structure constraint on the HMM's transition matrix is not trivial to achieve. Such structure arises from modeling recurring sequential patterns by sub-HMMs, as depicted in Figure 6.1.

More recently, other researchers have also constructed latent variable non-HMM acoustic models for speech. Latent Perceptual Mapping (LPM) [107] also uses co-occurrence frequencies of discrete symbols, but unlike [113], the

decomposition is based on SVD. LPM is an attempt to *sidestep* HMMs, which is a method for estimating HMM parameters (without supervision) and in no sense tries to do away with HMMs. Also the assumptions are very different, as e.g. LPM assumes phone segments are given. A similarity is indeed that VQ labels and some form of latent analysis is used. In that sense, LPM shows more similarity to our earlier work ([75] and [113]) where a latent variable method is applied on VQ data to construct a non-HMM acoustic model. A first important difference is that our earlier work considers VQ-label co-occurrence statistics while LPM uses label-unit co-occurrences. A second one is that LPM uses SVD, while our earlier work uses NMF to impose the constraint that co-occurrence statistics are non-negative.

Well-engineered methods for initializing HMMs

In [55], HMMs are built unsupervisedly by first discovering repeated acoustic patterns, called *word clusters* by segmental dynamic time warping (SDTW) [83, 129, 56], then training HMMs for each word cluster and finally clustering HMM states across words. An important difference with the current work is that a tensor factorization method takes over the role of the SDTW and is incorporated in the iterative process of HMM parameter estimation.

Other relevant work relates to the unsupervised training of HMMs for sub-word units [115, 29]. Their approach is to gradually increase model complexity using well-engineered methods for clustering and splitting states, with the goal of discovering *meaningful* sub-word units. By contrast, we start from an HMM topology for which the parameters are estimated from the data by exploiting a tensor factorization to force the HMM to meaningful solutions, an approach which - in principle - is not tailored towards the units (phones, syllables, words) it is modeling.

6.2 Non-negative Tucker Decomposition for HMM Training

In Chapter 5, co-occurrence based HMM learning was studied where non-negative matrix tri-factorization (NMTF) was successfully applied to learning hidden states and their transitions from co-occurrence statistics obtained from NMF. A low-rank tensor formulation that combines NMF and NMTF is now explored for learning HMMs, which will lead to a structured tensor decomposition. The reader is referred to Appendix B for notations and elementary tensor operations.

6.2.1 Non-negative Tensor Representation of Speech

Co-occurrence statistics for all training utterances are represented in a M -by- M -by- N tensor $\underline{\mathbf{X}}$. Its mode-3 slice, $\underline{\mathbf{X}}_{(3)}^{(n)}$, contains the co-occurrence matrix of utterance n with elements:

$$\underline{X}_{m,m',n} = \sum_{t=1}^{T_n-1} \Pr(X_t^{(n)} = \mathcal{G}_m, X_{t+1}^{(n)} = \mathcal{G}_{m'}), \quad (6.26)$$

Consider a low-rank decomposition of $\underline{\mathbf{X}}$ into a core tensor $\underline{\mathbf{B}}$ and the coefficient matrix \mathbf{A} and \mathbf{H} in Eq.(6.27) [79, 19].

$$\underline{\mathbf{X}} \approx \underline{\mathbf{B}} \times_1 \mathbf{A} \times_2 \mathbf{A} \times_3 \mathbf{H}^T. \quad (6.27)$$

which corresponds to the following element-wise operation,

$$\underline{X}_{m,m',n} \approx \sum_{k,k',r} \underline{B}_{k,k',r} A_{m,k} A_{m',k'} H_{r,n}. \quad (6.28)$$

KLD is the chosen metric for Eq.(6.27) as $\underline{X}_{m,m',n}$ is count data. The underlying probabilistic model is thus,

$$\Pr(\mathcal{G}_m, \mathcal{G}_{m'} | d_n) \approx \sum_{k,k',r} \Pr(S_k, S_{k'} | z_r) \Pr(\mathcal{G}_m | S_k) \Pr(\mathcal{G}_{m'} | S_{k'}) \Pr(z_r | d_n), \quad (6.29)$$

where z_r refers to the r -th sub-HMM and d_n refers the n -th utterance. As is illustrated in Eq.(6.29) and Figure 6.2, with the symmetric constraint of the mode-1 and mode-2 decompositions, matrix \mathbf{A} with entries $\Pr(\mathcal{G}_m | S_k)$ corresponds to the emission matrix of the Gaussian weight matrix of an HMM. The r -th mode-3 slice of the core tensor $\underline{\mathbf{B}}$, $\underline{\mathbf{B}}_{(3)}^{(r)}$ with entries $\Pr(S_k, S_{k'} | z_r)$, is the joint distribution matrix of hidden states of the r -th sub-HMM. $\underline{\mathbf{B}}_{(3)}^{(r)}$ has a sub-diagonal structure to reflect the Markovian property. \mathbf{H} with entries $\Pr(z_r | d_n)$ is the weight matrix of the sub-HMMs in the utterances, i.e. $H_{r,n}$ is the weight of sub-HMM r in utterance n . In the case where a single sub-HMM refers to a single whole-word, the activation $H_{r,n}$ is the probability of occurrence of word r in the utterance n , similar to the occurrence probability of a topic in a document in the area of document retrieval.

6.2.2 Structured Decomposition for HMM Learning

In the above model, all sub-HMMs share the same hidden states, which would cause confusion between different sub-HMMs. Orthogonality between the

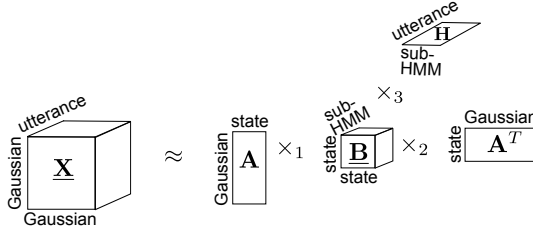


Figure 6.2: Non-negative Tucker decomposition of the data tensor $\underline{\mathbf{X}}$ to yield hidden states and sequential patterns modeled by sub-HMMs simultaneously.

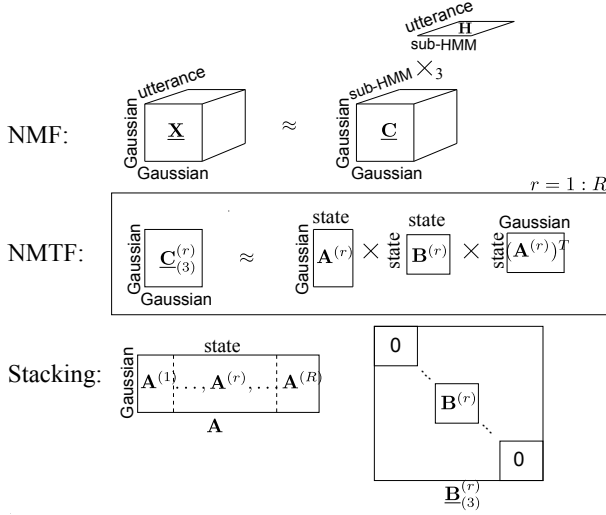


Figure 6.3: Structured non-negative Tucker decomposition $\underline{\mathbf{X}} \approx \underline{\mathbf{B}} \times_1 \mathbf{A} \times_2 \mathbf{A} \times_3 \mathbf{H}^T$ in three stages: (top) NMF to extract recurrent sequential patterns $\{\underline{\mathbf{C}}_{(3)}^{(r)}, r = 1, \dots, R\}$, (middle) NMTF is applied subsequently to learn a sub-HMM for each pattern r , (bottom) stacking the R emission matrices in one emission matrix (left) and expanding the R transition matrices with zeros and stacking these in the core tensor as mode-3 slices (right).

mode-3 slices of the core tensor $\underline{\mathbf{B}}$ is imposed to force different sub-HMMs to occupy different states. This way, the NTD learning generates an HMM as is shown in Figure 6.1.

This leads to the following constraints in the NTD:

- The columns of \mathbf{A} from $(r-1)L+1$ to rL correspond to the HMM states of the r -th sub-HMM.
- Orthogonality among the mode-3 slices of $\underline{\mathbf{B}}$ is imposed. The r -th mode-3 slice of $\underline{\mathbf{B}}$, $\underline{\mathbf{B}}_{(3)}^{(r)}$, is the *extended* joint distribution matrix of hidden states of the r -th sub-HMM, $\mathbf{B}^{(r)}$. As shown at the bottom of Figure 6.3, $\underline{\mathbf{B}}_{(3)}^{(r)}$ is constructed from $\mathbf{B}^{(r)}$ as $\underline{\mathbf{B}}_{(3)}^{(r)} = \text{blkdiag}(0, \dots, 0, \mathbf{B}^{(r)}, 0, \dots, 0)$ such that its size is equal to that of the transition matrix \mathbf{T} of the HMM. Moreover, $\mathbf{B}^{(r)}$ has a sub-diagonal structure for the HMM configuration in Figure 6.1. The transitions between the sub-HMMs will be explained in Eq.(6.31).

With the orthogonal constraints, the NTD can be solved by a two-stage NMF-NMTF algorithm as is shown in Eq.(6.30) and is illustrated in Figure 6.3.

$$\begin{aligned}
 \text{NMF} : \quad & \min_{\underline{\mathbf{C}}, \mathbf{H}} \text{KLD}(\underline{\mathbf{X}} \| \underline{\mathbf{C}} \times_3 \mathbf{H}), \\
 \text{NMTF} : \quad & \min_{\mathbf{A}^{(r)}, \mathbf{B}^{(r)}} \text{KLD}(\underline{\mathbf{C}}_{(3)}^{(r)} \| \mathbf{A}^{(r)} \mathbf{B}^{(r)} (\mathbf{A}^{(r)})^T), \\
 \text{Stacking} : \quad & \mathbf{A} = [\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(R)}], \quad \mathbf{B}_{(3)}^{(r)} = \text{blkdiag}(0, \dots, \mathbf{B}^{(r)}, \dots, 0),
 \end{aligned} \tag{6.30}$$

The size of the tensor $\underline{\mathbf{X}}$ is $M \times M \times N$. By reshaping the mode-3 slices of $\underline{\mathbf{X}}$ into vectors, we obtain the data matrix \mathbf{V} in Eq.(4.1) of Chapter 4 where NMF is subsequently utilized to extract R recurring sequential patterns represented by co-occurrences of Gaussians in a low-rank matrix \mathbf{W} . Every column of \mathbf{W} has size $M^2 \times 1$ and can be expanded to a Gaussian co-occurrence matrix which is stored as a mode-3 slice of the tensor $\underline{\mathbf{C}}$ with size $M \times M \times R$. The weight matrix \mathbf{H} thus has size $R \times N$. For each mode-3 slice $\underline{\mathbf{C}}_{(3)}^{(r)}$ which is a $M \times M$ matrix, NMTF is subsequently applied to find the $M \times L$ emission matrix $\mathbf{A}^{(r)}$ and the $L \times L$ transition matrix of the corresponding sub-HMM. Finally the emission matrix \mathbf{A} of the HMM is obtained by stacking the emission matrices of the sub-HMMs. The core tensor $\underline{\mathbf{B}}$ is constructed by extending and stacking the $\mathbf{B}^{(r)}$'s from the sub-HMMs.

A transformation φ and its inverse between the core tensor $\underline{\mathbf{B}}$ and the transition matrix \mathbf{T} of the HMM are first defined as follows.

$$\begin{aligned}
 \varphi : \quad & \mathbf{T} = \sum_r \underline{\mathbf{B}}_{(3)}^{(r)} + \boldsymbol{\varepsilon}, \\
 & T_{k,k'} \leftarrow T_{k,k'} / \sum_{k'} T_{k,k'}, \\
 \varphi^{-1} : \quad & \mathbf{B}^{(r)} = \mathbf{T}_{(r-1)*L+1:r*L, (r-1)*L+1:r*L}, \\
 & \underline{\mathbf{B}}_{(3)}^{(r)} = \text{blkdiag}(0, \dots, 0, \mathbf{B}^{(r)}, 0, \dots, 0).
 \end{aligned} \tag{6.31}$$

\mathcal{E} is a matrix connecting the end state of one sub-HMM and the beginning state of any other sub-HMM with a small positive constant ϵ (e.g. in the order of $1/K$), that is $\mathcal{E}_{rL,(r'-1)L+1} = \epsilon$ with $r = 1, \dots, R$ and zero elsewhere. The \mathcal{E} matrix may only be utilized when initializing \mathbf{T} , since it can be updated during the EM training.

6.3 Methods for Joint Training of NTD and HMMS

The optimization problems in NTD are not convex, so they could also be trapped in poor local optima. The *bag of co-occurrence* representation of a sequence in NMF of Eq.(6.30) also loses the sequential information which could result in models which contradict the sequential order. In other words, while the function of the NMF is to unmix the co-occurrence statistics originating from the different sequential patterns, there is no constraint on \mathbf{C} to express that the bag-of-co-occurrence representation of the patterns contained in its mode-3 slices can actually be generated from an observation sequence. Below, this constraint is achieved by linking the NTD and the HMM where the BW algorithm is chosen for the EM training of the HMM.

6.3.1 NTD Regularized BW (NTD.Reg.BW)

The first way to combine NTD and HMM is to construct a joint objective function by treating NTD as a regularizer:

$$\max_{\mathbf{A}, \mathbf{T}, \boldsymbol{\pi}, \mathbf{H}} \sum_n \log \Pr(\mathbf{O}^{(n)}; \mathbf{A}, \mathbf{T}, \boldsymbol{\pi}) - \lambda * \text{KLD}(\mathbf{X} || \varphi^{-1}(\mathbf{T}) \times_1 \mathbf{A} \times_2 \mathbf{A} \times_3 \mathbf{H}^T). \quad (6.32)$$

The first term is the log-likelihood of the observations $\{\mathbf{O}^{(n)}\}_{n=1}^N$ on its HMM, and the second term is the Kullback-Leibler divergence of the data tensor \mathbf{X} and its reconstruction. The HMM parameter estimates will fit both data representations: the first term measures how well the HMM is able to generate the actual sequences, while the regularization term expresses that the global co-occurrence view can be decomposed into additive parts along the lines of section 6.2.

\mathbf{H} only occurs in the second term, so its update algorithm remains unchanged from the algorithm of NMF [65]. Similarly, $\boldsymbol{\pi}$ only occurs in the HMM so EM

updates can be used. The optimization of \mathbf{A} and \mathbf{T} in the HMM and the NTD is based on auxiliary functions which boil down to the following optimization problem:

$$\begin{aligned} \max_{y_1, \dots, y_M} \quad & \mathcal{L}(y_1, \dots, y_M) = \sum_{m=1}^M \omega_m \log(y_m) + \lambda \mu_m \log(y_m), \\ \text{s.t.} \quad & \sum_{m=1}^M y_m = 1, y_m \geq 0. \end{aligned} \quad (6.33)$$

where $\mathbf{y} = (y_1, \dots, y_M)$ denotes a column of \mathbf{A} or a row of \mathbf{T} , ω_m is the term of constants (w.r.t. \mathbf{y}) derived from the Q-functions of the EM algorithm $Q_A(\mathbf{A}, \bar{\lambda})$ in Eq.(6.17), Eq.(6.18) or Eq.(6.20) for DDHMM, CDHMM and KLDHMM respectively. μ_m is the term derived from the auxiliary function of the NTD (Eq.(27) in [65] and Eq.(C.3) in the Appendix C). Hence, the joint update algorithm obtained by applying Lagrange multipliers to Eq.(6.33) is as follows.

$$y_m = \frac{\omega_m + \lambda \mu_m}{\sum_{m'=1}^M \omega_{m'} + \lambda \mu_{m'}} \quad (6.34)$$

In Eq.(6.34), the ω_m sum to approximately the total sequence length or the number of frames. Hence, the scale of μ_m should be matched to the amount of data such that one does not have to tune the regularization constant λ to balance the HMM term and the NTD term. A *scale-keeping algorithm* is proposed to ensure the relative scale between the NTD term and the HMM term in Eq.(6.32) is unchanged with respect to the amount of data (i.e. the number of frames). The update of \mathbf{H} only involves NMF, $\mathbf{V} \approx \mathbf{W}\mathbf{H}$, where a column of \mathbf{V} or \mathbf{W} is a flattened mode-3 slice of $\underline{\mathbf{X}}$ or $\underline{\mathbf{C}}$. With the updated \mathbf{H} , \mathbf{W} is computed by,

$$W_{i,r} \leftarrow W_{i,r} \sum_n \frac{V_{i,n}}{\sum_t W_{i,t} H_{t,n}} H_{r,n}. \quad (6.35)$$

It is straightforward to check that,

$$\sum_r W_{i,r} = \sum_r W_{i,r} \sum_n \frac{V_{i,n}}{\sum_t W_{i,t} H_{t,n}} H_{r,n} = \sum_n V_{i,n}, \quad (6.36)$$

which states that the row sums in \mathbf{V} are retained in the row sums of the obtained low rank matrix \mathbf{W} . Hence, the total count of Gaussian co-occurrences in $\underline{\mathbf{X}}$ is the same as the total count of Gaussian co-occurrences in $\underline{\mathbf{C}}$ that is,

$$\sum_{m,m',n} \underline{X}_{m,m',n} = \sum_{m,m',r} \underline{C}_{m,m',r}. \quad (6.37)$$

In the following NMTF algorithm $\mathbf{Z} \approx \mathbf{X}\mathbf{Y}\mathbf{X}^T$, the input is $\mathbf{Z} = \underline{\mathbf{C}}_{(3)}^{(r)}$ and the outputs are $\mathbf{X} = \mathbf{A}^{(r)}$ and $\mathbf{Y} = \mathbf{B}^{(r)}$. In section 5.2.3 of Chapter 5, we showed that $\sum_{m,m'} Z_{m,m'} = \sum_{k,k'} Y_{k,k'}$ with the column-wise normalization of \mathbf{X} . Therefore the count of Gaussian co-occurrences in the slice $\underline{\mathbf{C}}_{(3)}^{(r)}$ is equal to the count of state co-occurrences in $\mathbf{B}^{(r)}$. The count is also equal to the count of state co-occurrences in the slice $\underline{\mathbf{B}}_{(3)}^{(r)}$ which is an extended version of $\mathbf{B}^{(r)}$ by padding zeros as shown in Eq.(6.30).

$$\sum_{m,m',r} \underline{\mathbf{C}}_{m,m',r} = \sum_{k,k',r} \underline{\mathbf{B}}_{k,k',r}. \quad (6.38)$$

Therefore, the total number of Gaussian co-occurrences in $\underline{\mathbf{C}}$ is equal to the total number of state co-occurrences in $\underline{\mathbf{B}}$. Because of the orthogonality in the mode-3 slices of $\underline{\mathbf{B}}$, the estimated transition matrix \mathbf{T} from the NTD by using Eq.(6.31) has the same number of state co-occurrences as in $\underline{\mathbf{B}}$. We have finally shown that,

$$\sum_{i,j,n} \underline{\mathbf{X}}_{m,m',n} = \sum_{m,m',r} \underline{\mathbf{C}}_{m,m',r} = \sum_{k,k',r} \underline{\mathbf{B}}_{k,k',r} = \sum_{k,k'} T_{k,k'}. \quad (6.39)$$

So the total number of co-occurrence counts in the estimated transition matrix \mathbf{T} from NTD is the same as the total number of Gaussian co-occurrences in the training data which is equal to the total number of state co-occurrences from the BW training. By their definitions, $T_{k,l} = \Pr(S_k, S_l)$ and $A_{i,k} = \Pr(\mathcal{G}_i | S_k)$, while in the BW training, the estimate of $A_{i,k}$ is the accumulated co-occurrences of Gaussian i and state k , i.e. $\Pr(\mathcal{G}_i, S_k)$. Correspondingly, we re-scale the emission matrix \mathbf{A} estimated from NTD by $A_{i,k} \leftarrow A_{i,k} \sum_l T_{k,l}$ to transfer the conditional probability $\Pr(\mathcal{G}_i | S_k)$ to the joint probability $\Pr(\mathcal{G}_i, S_k)$. Now the obtained \mathbf{A} and \mathbf{B} from both NTD and BW should have the same scale.

With the optimization algorithm, the joint objective function in Eq.(6.32) will increase monotonously. However, in the above procedures, the iterations in NTD and HMM are *synchronous*, i.e. both of them are updated with the same number of iterations. This may not be the best choice according to our experience where NTD normally requires much more iterations to convergence (e.g. hundreds or thousands in Chapter 3, Chapter 4 and Chapter 5) than the iterations in the EM training of an HMM. So though this method ensures non-decrease of the joint objective function, the synchronous update may be too slow for practical purposes.

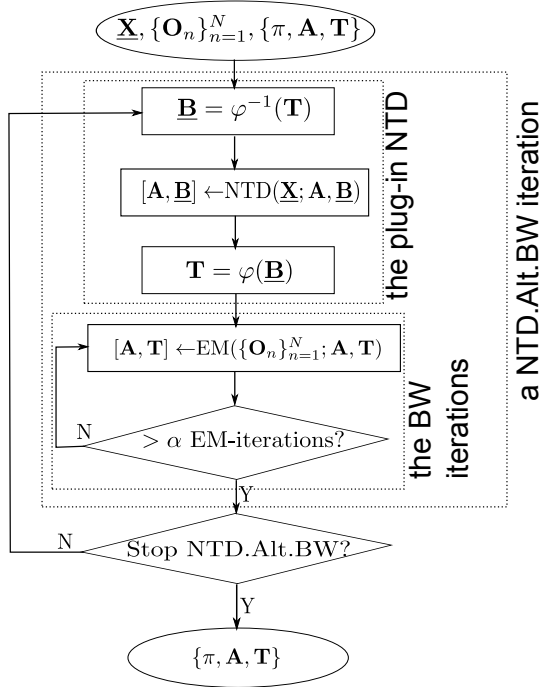


Figure 6.4: The alternating training framework of NTD.Alt.BW. The current HMM parameters are transformed to their tensor form by φ^{-1} to initialize the structured tensor decomposition. Then the tensor parameters are transformed back to HMM parameters by φ and α Baum-Welch iterations are performed. This process is repeated several times.

6.3.2 Alternative Training of NTD and BW (NTD.Alt.BW)

A method to combine NTD and BW in an asynchronous manner is to plug the NTD into the EM iterations as is illustrated in Figure 6.4 where NTD and BW switch at every α EM training iterations.

The inputs of the training scheme are the observation utterance group $\{\mathbf{O}_n\}_{n=1}^N$, the non-negative tensor $\underline{\mathbf{X}}$ and the initializations $\Lambda = \{\mathbf{A}, \mathbf{T}, \boldsymbol{\pi}\}$. $\boldsymbol{\pi}$ is initialized to an uniform distribution over all initial states and is set to zero for all other states. \mathbf{A} is initialized to *i.i.d* random variables with an uniform distribution between zero and one and is then normalized to have columns of unity ℓ_1 norm. \mathbf{T} is initialized as a transition matrix of the HMM in Figure 6.1. The loop and transition probabilities in \mathbf{T} are first generated from a uniform distribution

between zero and one. The self-loop and outgoing probabilities of each state are subsequently normalized to sum to unity.

The initial HMM parameter estimates $\Lambda = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{T}\}$ are transformed to the initial estimates of the NTD parameters: the matrix \mathbf{A} and the core tensor $\underline{\mathbf{B}}$. Subsequently NTD as presented in section 6.2 is performed. The updated $\underline{\mathbf{B}}$ is transformed back to its state transition form \mathbf{T} and together with the through NTD updated emission matrix \mathbf{A} , several EM iterations are performed. The updated $\{\mathbf{A}, \mathbf{T}\}$ are used to initialize the NTD. The process is iterated until the stopping criterion is met.

The NTD block in Figure 6.4 is computed as presented in section 6.2. The block of EM training is approached by the BW training [87, 5]. The stopping criterion of the EM training and the overall NTD.Alt.BW training could be a sufficient number of iterations over the training data or an insufficient increase of the data likelihood. Much like is common practice in EM training of HMMS for speech processing, we have opted for a fixed number of EM iterations α and a fixed number of NTD.Alt.BW iterations β . The total number of EM iterations is thus $\alpha * \beta$.

6.4 Experiments on TIDIGITS

For the experiments on TIDIGITS, we first compare our algorithms and several baselines for the training of DDHMMs. The proposed algorithms are then extended to CDHMM and KLDHMM. Finally, we will give discussions and suggestions on the algorithms. Details on the TIDIGITS database are given in section 3.3.3.

6.4.1 Data Preparation

Like in conventional ASR systems, speech is chopped into overlapping frames to compute the short-term spectral features known as Mel-Frequency Cepstral Coefficients (MFCC) [52]. The window length is 25 ms with a frame shift of 10 ms. For each frame, a 12 dimensional MFCC is extracted from a bank of 30 Mel-scaled filters. Each frame is represented by its MFCC plus the frame's log-energy. First and second order differences are computed and concatenated to form a 39-dimensional feature vector. A Gaussian mixture of $M=1000$ components is trained without supervision on the training set using maximum likelihood EM training.

- For DDHMM, a training utterance, indexed by n , is characterized by two representations: an observation sequence $\mathbf{O}^{(n)}$ and a co-occurrence matrix $\underline{\mathbf{X}}_{(3)}^{(n)}$. The sequence is obtained by quantizing each frame as the Gaussian identity with the highest posterior probability. The co-occurrence matrix is computed from the accumulated co-occurrences of Gaussian symbols of this utterance, as presented in Eq.(6.26).
- For SCDHMM, the n -th training utterance, is also characterized by two representations: an observation sequence $\mathbf{O}^{(n)}$ and a co-occurrence matrix $\underline{\mathbf{X}}_{(3)}^{(n)}$. The sequence is just the MFCC+ Δ + $\Delta\Delta$ vectors. The co-occurrence matrix is computed from the accumulated co-occurrences of Gaussian posterior probabilities of this utterance, as formulated in Eq.(6.26).
- For KLDHMM, a training utterance, indexed by n , is again characterized by two representations: an observation sequence $\mathbf{O}^{(n)}$ and a co-occurrence matrix $\underline{\mathbf{X}}_{(3)}^{(n)}$. The sequence is obtained by computing the posterior probabilities on the Gaussians for each frame, i.e. a Gaussian posteriorgram. The co-occurrence matrix is computed from the accumulated co-occurrences of Gaussian posterior probabilities of this utterance according to Eq.(6.26).

In all three cases, the complete training data is hence described as a set of sequences $\{\mathbf{O}_n\}_{n=1}^N$ and a non-negative tensor $\underline{\mathbf{X}}$ with count data, but note there are different contents in \mathbf{O}_n .

6.4.2 Experiments Using DDHMM

The goal of the experiments on DDHMM is to examine if the proposed methods outperform the baselines.

In the first experiments, we choose $R = 30$ sub-HMMs, each with $L = 10$ states. For training the NTD regularized BW (NTD.Reg.BW), the regularization parameter is selected to be $\lambda = 1$ and the number of EM iterations is 25. In the alternating training of NTD and BW (NTD.Alt.BW), the BW iteration is initialized from an NTD and is then interrupted by the plug-in NTD every $\alpha=5$ iterations where the NTD uses 100 iterations. The total number of these NTD interruptions is $\beta=5$. Hence the total number of EM iterations in HMM is also 25 and the total number of iterations in NTD is 500. Notice that NTD iterations require far less computational resources than EM iterations because there is no need to process every data frame, but only the co-occurrence statistics which are computed once.

Since we are dealing with non-convex optimization, initialization is important. We generate five groups of initial HMM parameter values randomly, which are each used as a starting point for all methods. For each method, we thus obtain five performance metrics of which the variance is observed, which constitutes an analysis of the sensitivity of the methods initialization.

Baselines

Three methods for unsupervised learning of HMMS are experimented with as baselines:

- **BW** is Baum-Welch training [88], which estimates the HMM parameters such that the data likelihood is maximized given the HMM topology of Figure 6.1. Like in the proposed methods, 25 iterations are performed starting from the same HMM.
- **BW+SA** is BW training with simulated annealing (SA) [84], in which the HMM parameter estimates are randomly perturbed by an amount that is proportional to $\exp(-0.1i)$ where i is the iteration number. Because of the perturbations, the number of iterations was set to 100 here. At the end of the cooling, an additional 25 perturbation-free iterations are conducted to ensure the final convergence.
- **JLH** is Joint Label and HMM parameter estimation as used in [95] to construct so-called self-organized units to represent speech as is explained in section 6.1.

All three baseline methods are initialized from the same five random HMM parameter groups.

Evaluation metrics

The models are evaluated by the following criteria [104].

(1) Data likelihood

We report the final log likelihoods $\sum_n \log \Pr(\mathbf{O}_n; \mathbf{\Lambda})$ for each of the methods. Monotonous increase of the likelihood was observed in all the models except BW+SA and NTD.Alt.BW. Since NTD maximizes a different criterion, the data likelihood on the HMM decreases, but subsequent EM iterations quickly recover as is illustrated in the drawing of NTD.Alt.BW in the top panel of Figure 6.5. The middle panel shows that the NTD interruptions are actually

Table 6.1: Evaluation of BW, BW+SA, NTD.Reg.BW and NTD.Alt.BW for unsupervised learning of discrete density HMMs: log-likelihood of the training data, segmentation accuracy in %, model purity in % and recognition accuracy in %.

	BW	BW+SA	NTD.Reg.BW ($\lambda = 1$)	NTD.Alt.BW
log-likelihood ($\times 10^6$)	-5.62 ± 0.04	-5.49 ± 0.05	-5.58 ± 0.04	-5.74 ± 0.05
segmentation	61.6 ± 3.0	65.0 ± 1.1	64.0 ± 2.3	78.6 ± 1.5
purity	76.1 ± 2.6	79.6 ± 3.5	78.3 ± 3.3	81.0 ± 2.0
recognition	62.2 ± 5.8	69.6 ± 5.3	69.8 ± 6.6	85.7 ± 1.4

beneficial to the accuracy (as defined below) compared to only one initializing NTD prior to the EM iterations (NTD.Init.BW). The benefit is also reflected in the final log likelihoods, which are -5.707 and -5.711 ($\times 10^6$) for NTD.Alt.BW and NTD.Init.BW respectively. All the training schemes were observed to converge with final likelihoods listed in the first row of Table 6.1.

BW+SA successfully found solutions with high data likelihood, but note that due to the nature of the method, 125 EM iterations were run here while 25 iterations were performed for the other methods. Comparing the likelihoods with the other evaluation metrics below, we conclude that data likelihood may not be a perfect choice for measuring the quality of unsupervised learning methods.

(2) Speech segmentation

A second evaluation criterion is segmentation accuracy. The reference segmentation is obtained by segmenting the training utterances with a HMM that was trained on the same data but using the orthographic transcriptions as supervisory information. The accuracy metric used here compares the reference segmentation boundaries $b_i, 1 \leq i \leq I$ of an utterance with its boundaries $\tilde{b}_j, 1 \leq j \leq J$ obtained with the unsupervised methods through a dynamic programming alignment of both boundary lists. First, all boundary pairs are compared with a threshold ξ to generate a distance matrix $d_{i,j}$: their distance is zero if below or equal to the threshold and one otherwise. Then the path of lowest cost from $d_{1,1}$ to $d_{I,J}$ is sought allowing only diagonal, horizontal or vertical transitions. The total cost is summed over all utterances and divided by the total number of boundaries in the reference segmentations of the whole training corpus. The segmentation accuracy is subsequently computed by subtracting the above cost value from 100.

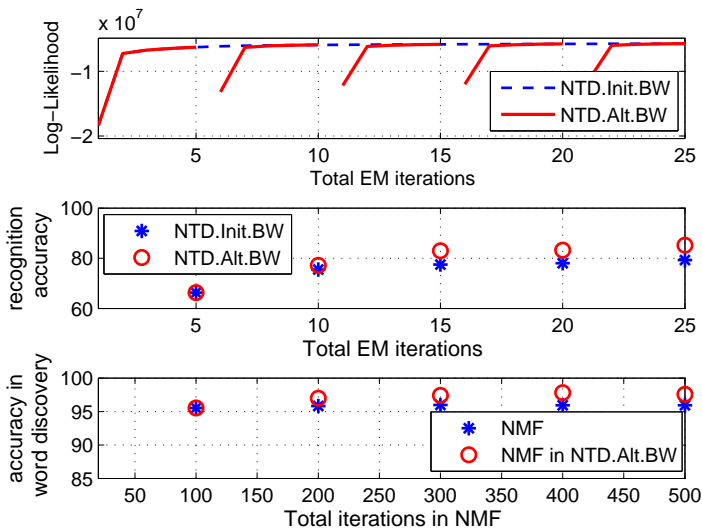


Figure 6.5: Convergence analysis. Top: log-likelihood of the training data for NTD.Alt.BW and initializing BW with only one NTD (NTD.Init.BW). Middle: recognition accuracy in % of the intermediate HMMs. Bottom: accuracy in % of the intermediate NMF models.

The results with the 5 random initializations with $\xi=5$ frames (50 ms) are shown in Table 6.1. The discovered sequential patterns in all the models have strong relations with the digits, but the patterns of NTD.Alt.BW are better than those trained only by BW (first column). Simulated annealing (second column) and NTD.Reg.BW (third column) improve slightly but consistently over BW.

(3) Average purity of the discovered patterns

Because the sequential patterns have been learned without supervision, they are anonymous, i.e. not identified as one of the words in the vocabulary. In order to analyze the discovered patterns, a mapping matrix \mathbf{Q} between the digits and the sub-HMMs is first constructed. Hereto, \mathbf{Q} is constructed as the co-occurrence matrix between ground truth digits and recognized sub-HMMs. The ground truth is approximated by a segmentation generated with a supervised HMM as described above, yielding a digit label for every data frame. $Q_{j,r}$ is then the number of frames that are labeled as digit j ($1 \leq j \leq J$) in the ground truth and as sub-HMM r ($1 \leq r \leq R$) at the end of the unsupervised training. $J = 12$ since we also define a silence word.

Some examples of mapping matrices obtained with different methods using the same initialization are shown in Figure 6.6. Here the columns (sub-HMMs) have been sorted according to the digits they represent. Most sub-HMMs are strongly related to one digit, while some are mixtures of different digits (e.g. pattern 24-30 in the top left panel). Some digits are modeled by multiple patterns (e.g. digit 5 in all the panels), which is not undesired as this constitutes a model of variability that is not related to semantics, such as gender or pronunciation variation. Pattern 30 in the top panels is probably a sub-word unit of “s” because the pattern is strongly related to both “six” and “seven” which share the same phone “s”. As can be seen in Figure 6.6, the NTD.Alt.BW training methods show good performance to extract pure patterns.

To make the visual evaluation of Figure 6.6 more quantitative, the purity of sub-HMM r is defined as $\max_{l'} Q_{l',r} / \sum_{l'} Q_{l',r}$. The best achievable purity is 100%, where each sub-HMM is strictly mapped to one digit or silence. The results are reported in Table 6.1. The NTD.Alt.BW training outperforms the other methods in finding patterns with high purity. The models obtained with simulated annealing also score well in terms of purity.

(4) Recognition accuracy on the test set

In this section, the learned sub-HMMs are used for recognition using Viterbi decoding. In order to be able to score the recognition result, each sub-HMM must be associated to a digit or silence. We rely on the mapping matrices \mathbf{Q} obtained above to achieve this. More sophisticated methods such as inducing a finite state grammar from a lattice recognition output [78] could be proposed. To implement the co-occurrence method, we first normalize the columns of the mapping matrix $Q_{j,r} \leftarrow Q_{j,r} / \sum_{j'} Q_{j',r}$ and then relate each sub-HMM to either a digit or garbage (i.e. silence, mixture or sub-word that does not relate clearly to any digit). A threshold η is utilized to identify the garbage patterns $\{r | \max_{l'} Q_{l',r} < \eta\}$, for example pattern 24 to 30 in the top left panel of Figure 6.6. If some digit is not modeled by any sub-HMM, we use the garbage model with maximal joint probability with the digit. In this way, in the top left panel of Figure 6.6, digit “six” is modeled by pattern 30 and digit “eight” is modeled by pattern 25.

The learned HMM is subsequently applied as a word recognizer on the test set. For each method, the word entry penalty corresponding to the minimal error rate on a fixed subset of 50 utterances from the training set is selected. Since all methods benefit from the same tuning advantage, none is favored over the other. The results are shown in the bottom row of Table 6.1 for $\eta=0.6$. The accuracy is computed by subtracting the word error rate (including insertions, deletions and substitution) from 100. Occurrences of silence and garbage in the recognized string are removed for scoring. The NTD.Alt.BW training improves

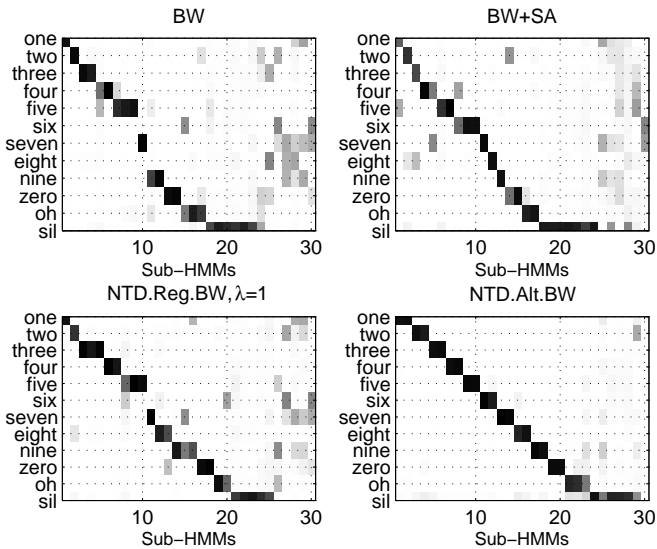


Figure 6.6: Mapping matrices between the sub-HMMs and the digits for four different methods initialized equally ($R = 30$, $L = 10$).

the baselines significantly. $BW+SA$ and $NTD.Reg.BW$ have also improved over BW in this respect. For comparison, a HMM trained with supervision with 25 Viterbi iterations yields an accuracy of 97.4%.

Analysis and discussion

(1) Baselines

JLH with different initializations: Table 6.2 summarizes the impact of performing JLH iterations initialized from the BW or $NTD.Alt.BW$ models, as explained in section 6.4.2. The performance of JLH relies heavily on its initialization and is unable to consistently improve over its initialization. Note that in this table, the results of BW and $NTD.Alt.BW$ are obtained by running 25 extra iterations on the ones in Table 6.1 so the total numbers of EM iterations of all the four methods in Table 6.2 are the same.

$BW+SA$: $BW+SA$ performs better than BW , especially in recognition accuracy. However, SA requires more training time. Because of the random perturbations, results are more difficult to reproduce than for other methods. $BW+SA$ is successful in finding a solution that has better likelihood on the training data.

Table 6.2: Evaluation of the JLH method: 25 JLH iterations, initialized from the BW and NTD.Alt.BW models respectively ($R = 30, L = 10$).

	BW	JLH init. by BW	NTD.Alt.BW	JLH init. by NTD.Alt.BW
log-likelihood ($\times 10^6$)	-5.56 ± 0.03	-5.67 ± 0.03	-5.58 ± 0.02	-5.77 ± 0.04
segmentation	61.8 ± 3.0	61.1 ± 2.7	79.7 ± 2.8	77.7 ± 2.0
purity	77.1 ± 2.3	76.3 ± 2.4	85.4 ± 1.5	83.8 ± 1.3
recognition	61.3 ± 3.7	51.6 ± 7.4	87.4 ± 3.8	89.0 ± 3.0

However, it’s sub-HMMs are not as pure as what is obtained with NTD.Alt.BW as shown in Table 6.1 and Figure 6.7. Closer inspection of segmentation results shows that the sub-HMMs found by BW+SA contain both phone-like sub-words and larger word-sized units that are not pure. In the top right panel of Figure 6.6 for instance, four sub-HMMs are mapped to the digit “six”. Upon inspection of segmentations, we observe they often occur in the same order. The first sub-HMM (the 8-th column), however, is also observed as the first sub-HMM of “four”, i.e. it could be modeling the phones “s” and “f”. Inspection of segmentations also shows that the 15-th sub-HMM is modeling both “two” and “zero” as a whole-word model. For NTD.Alt.BW such phenomena appear not as often as is witnessed by Figure 6.6 and the high values of segmentation accuracy, purity and recognition accuracy.

(2) The role of NTD in HMM training

Consistent improvement is observed from BW to NTD.Reg.BW for all five initializations and for the performance metrics of segmentation, purity and recognition, though the gain is not as large as for NTD.Alt.BW. NTD.Reg.BW has a well-formulated optimization framework, but performs worse than NTD.Alt.BW in terms of segmentation accuracy, purity and recognition accuracy. The reason might be attributed to the synchronous updating of NTD and BW. From past experience, we know that NTD requires hundreds of iterations to converge. In NTD.Reg.BW the KLD between the data tensor and its approximate factorization modifies the EM update rules. Since we compare all methods on the same number of EM iterations, there is a serious concern that the tensor factorization has insufficiently converged in NTD.Reg.BW. More iterations do lead to slight performance improvement, but NTD.Reg.BW never reaches the performance of NTD.Alt.BW.

The middle panel in Figure 6.5 shows how the recognition accuracy of

the HMMS evolves through the iteration process. The intermediate HMMS in NTD.Alt.BW perform better than the BW training with the same initializations and with the same number of iterations. Similarly, the quality of the NMF models in Eq.(6.30) during the iteration can be evaluated with the unordered word error rate (UWER) as described in Chapter 2. We compare the performance of the intermediate NMF models with a plain NMF model that is initialized identically but only performs the decomposition described in Chapter 4. The bottom panel of Figure 6.5 shows 100-UWER for the NMF model in NTD.Alt.BW and for the plain NMF model as a function of the total number of NMF iterations. Also here we observe an improvement. Hence we can conclude that alternating optimization of the NTD and HMM benefits both representations.

(3) Parameter sensitivity

In the unsupervised training, no prior information or grounding labels can be utilized to tune the parameters. Therefore a good model should be robust to the selection of the user-specified parameters. The number of sub-HMMs R and the number of hidden states per sub-HMM L are the two most important parameters in the configuration. We thus report the parameter sensitivity w.r.t. them in Figure 6.7. In general, ten states per sub-HMM generate better performance than five states on word discovery on the TIDIGITS database. NTD.Alt.BW is more robust than BW+SA for sensitivity to the number of states. Only for the performance metric of purity and with 10 states, BW+SA approaches the results of NTD.Alt.BW. However, 125 BW iterations were conducted for BW+SA while only 25 iterations for NTD.Alt.BW, making the former a lot slower to train.

The performance of the NTD.Reg.BW with respect to different scales of the regularization parameter λ is shown in Figure 6.8. As is expected, $\lambda = 1$ makes a good balance between HMM and NTD in NTD.Reg.BW and $\lambda > 1$ yields significantly different results from BW. The likelihood of NTD.Reg.BW decreases with increasing λ , which is not unexpected since more weight is attributed to NTD which does not maximize the likelihood of the sequential data directly. With large regularization parameters, NTD.Reg.BW is observed to outperform BW and BW+SA on segmentation and recognition accuracy. The set of sub-HMMs still contains members that are capable of modeling the digits accurately. However, inspection of the mapping matrices also reveals that more garbage sub-HMMs are found as a consequence of relaxing the frame transition constraints, which explains the lower purity in Figure 6.8.

(4) Statistical significance

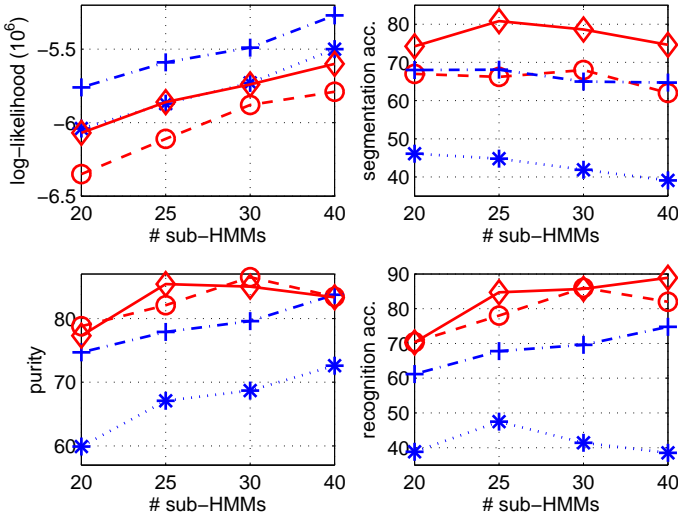


Figure 6.7: The performance of BW+SA and NTD.Alt.BW with respect to numbers of hidden states and number of sub-HMMs. *: BW+SA with 5 states, \circ : NTD.Alt.BW with 5 states, +: BW+SA with 10 states and \diamond : NTD.Alt.BW with 10 states.

Some of the standard deviations shown in Table 6.1 may cast doubt on the conclusion that the proposed algorithms yield better solutions. For each of the five attempts, the same initialization was used for all the four algorithms, therefore paired t -tests were conducted in this subsection to check the significance of the differences. The results are listed in Table 6.3. From the table, we see that NTD.Reg.BW indeed outperforms BW but does not outperform BW+SA and that NTD.Alt.BW outperforms all other algorithms. The test rejects the hypothesis that BW+SA improves BW with a high p value.

6.4.3 Comparison of DDHMM, SCDHMM and KLDHMM

After the extensive analysis of DDHMM learning with NTD methods in the previous sections, we extend our analysis to the SCDHMM and KLDHMM variants. As in the DDHMM, the HMM parameters are initialized randomly. The algorithm will discover some patterns from the continuous speech data containing repeated words. The patterns are subsequently evaluated by a frame-by-frame matching to supervised segmentations to compute their purities as

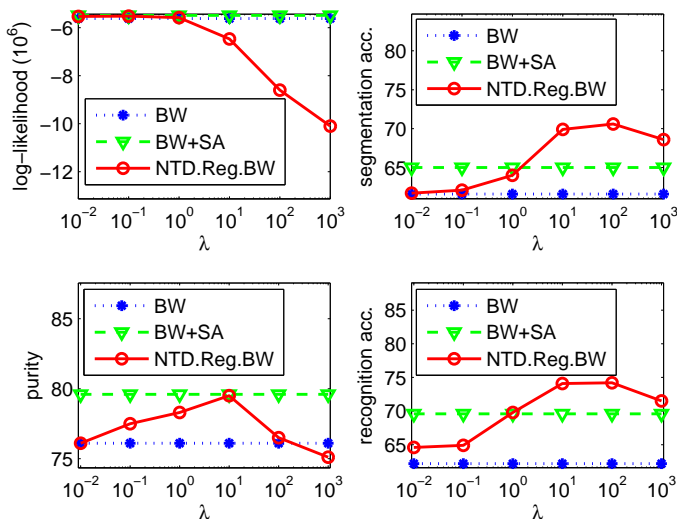


Figure 6.8: The performance of NTD.Reg.BW with respect to the regularization parameter λ . The number of sub-HMMs is $R = 30$ and the number of states per sub-HMM is $L = 10$. The NTD.Reg.BW and BW are with 25 BW iterations, while the BW+SA is with 125 iterations.

presented in section 6.4.2. High purity or other meaningful explanations (e.g. phones) are expected for a good HMM. The mapping matrices obtained from SCDHMM and KLDHMM are shown below.

SCDHMM

The mapping matrices from unsupervised training of SCDHMM with two different random initializations (attempts) are illustrated in Figure 6.10. Since a continuous density is used to describe the emission density of a hidden state, a relatively small number $M = 500$ of Gaussians are used instead of 1000 Gaussians in the above experiments. $R=25$ sub-HMMs each of which has $L = 10$ hidden states are created and initialized randomly. The BW training uses 25 EM passes, while 5 NTD passes and 5 EM passes are applied in NTD.Alt.BW. Compared with Table 6.1 and Figure 6.6, patterns discovered by the BW training of SCDHMM have higher purity than the ones from the BW training of DDHMM. The NTD.Alt.BW training of SCDHMM yields the best results regarding the pure patterns corresponding to digits.

Table 6.3: Details of the two-tailed paired t -test: $H_0 : \mu_0 = \mu_1$ v.s. $H_1 : \mu_0 \neq \mu_1$. h is the result of the test: $h=0$ indicates that the null hypothesis cannot be rejected at the 5% significance level, while $h=1$ indicates that the null hypothesis can be rejected at the 5% level. p is the probability of observing the given result or a more extreme value, if the null hypothesis is true. Small values of p cast doubt on the validity of the null hypothesis. ci is a 95% confidence interval for the true mean, or of $\mu_1 - \mu_0$ for a paired test.

μ_1	BW+SA	NTD.Reg.BW	NTD.Alt.BW
μ_0	BW	BW	BW
	$h=0$	$h=1$	$h=1$
	$p=0.08$	$p=6.910^{-3}$	$p=6.610^{-4}$
	$ci=[-1.55, 17.62]$	$ci=[3.22, 10.82]$	$ci=[17.14, 31.12]$
μ_1	NTD.Reg.BW	NTD.Alt.BW	NTD.Alt.BW
μ_0	BW+SA	BW+SA	NTD.Reg.BW
	$h=0$	$h=1$	$h=1$
	$p=0.82$	$p=4.710^{-4}$	$p=4.310^{-3}$
	$ci=[-12.65, 10.62]$	$ci=[11.83, 20.36]$	$ci=[8.97, 25.25]$

KLDHMM

The KLDHMM uses the same $M = 500$ Gaussians, the HMM structure with $R = 25$ sub-HMMs and $L = 10$ states each, and the same initializations as in SCDHMM. The mapping matrices from unsupervised training of KLDHMM with two different initializations (attempts) are illustrated in Figure 6.10.

From the mapping matrices, we observe that both BW and NTD.Alt.BW training of KLDHMM perform worse than the unsupervised training of DDHMM and SCDHMM in the sense of high purity. Much confusion is observed between the sub-HMMs obtained from the unsupervised training of KLDHMM. One possible reason might be the BW algorithm in KLDHMM brings too much uncertainty. Careful inspections of the acoustic model in KLDHMM were made to check if it is suitable to use Eq.(6.6) as an emission density. By expanding Eq.(6.6), we obtain the relation between the observation posteriors \mathbf{X}_t at frame t and the emission probabilities \mathbf{A}_k of state k as presented in Eq.(6.6) where $0 \leq X_{m,t} \leq 1$, $0 < A_{m,k} < 1$, $\sum_m X_{m,t} = 1$

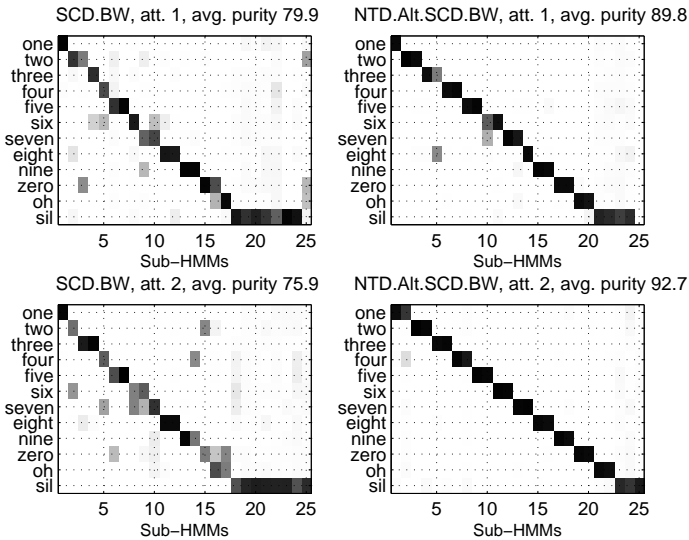


Figure 6.9: Mapping matrices from the BW training and the NTD.Alt.BW training of SCDHMM. The BW training is with 25 EM passes, while the NTD.Alt.BW training methods are with 5 NTD passes and 5 BW passes.

and $\sum_m A_{m,k} = 1$. It can be decomposed into three cases:

$$\left(\frac{A_{m,k}}{X_{m,t}}\right)^{X_{m,t}} = 1, \text{ if } X_{m,t} = 0, \quad (6.40)$$

$$\left(\frac{A_{m,k}}{X_{m,t}}\right)^{X_{m,t}} > 1, \text{ if } 0 < X_{m,t} < A_{m,k}, \quad (6.41)$$

$$\left(\frac{A_{m,k}}{X_{m,t}}\right)^{X_{m,t}} < 1, \text{ if } X_{m,t} > A_{m,k}. \quad (6.42)$$

Eq.(6.41) and Eq.(6.42) are a bit counterintuitive since a small $X_{m,t}$ amplifies the total likelihood in Eq.(6.41) while a large $X_{m,t}$ in Eq.(6.42) contracts the total likelihood. Then during the estimation of the posterior probabilities on hidden states of the frames in the forward-backward algorithm, irrelevant hidden states could obtain a high probability from Eq.(6.41), while relevant states could obtain a low probability according to Eq.(6.41).

Since too much confusions between sub-HMMs are observed in the patterns with BW training of KLDHMM, Viterbi alignment is implemented by only

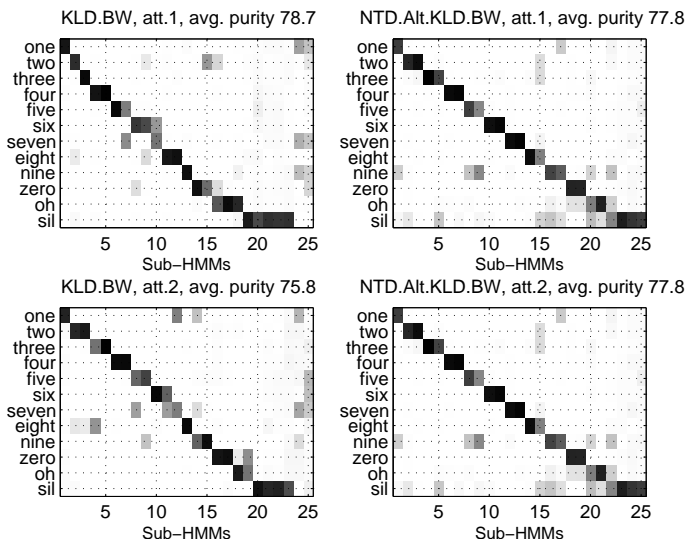


Figure 6.10: Mapping matrices from the BW training and the NTD.Alt.BW training of KLDHMM. The BW training is with 25 EM passes, while the NTD.Alt.BW training methods are with 5 NTD passes and 5 BW passes.

taking the best path instead of retaining all the paths in BW. Similarly, the alternative training of Viterbi alignment and NTD is called NTD.Alt.Vit. The mapping matrices from Viterbi training and NTD.Alt.Vit are shown in Figure 6.11. The parameters and initializations are exactly the same as in the BW training of KLDHMM. Word patterns with high purity are obtained. We also tried the unsupervised training of KLDHMM with only iterative Viterbi alignment. However, the $R = 25$ sub-HMMs collapsed to one single sub-HMM after a couple of EM iterations and no meaningful patterns have been discovered.

6.4.4 Learning from a Few of Labeled Examples and Many Unlabeled Continuous Utterances

In vocabulary acquisition by children it is not uncommon that care-givers show objects and name them using isolated utterances. These words are later used in continuous utterances. In robot language acquisition, a similar learning paradigm may be implemented. In this section, we will show that this scenario is very beneficial to the accuracy, even though only two labeled

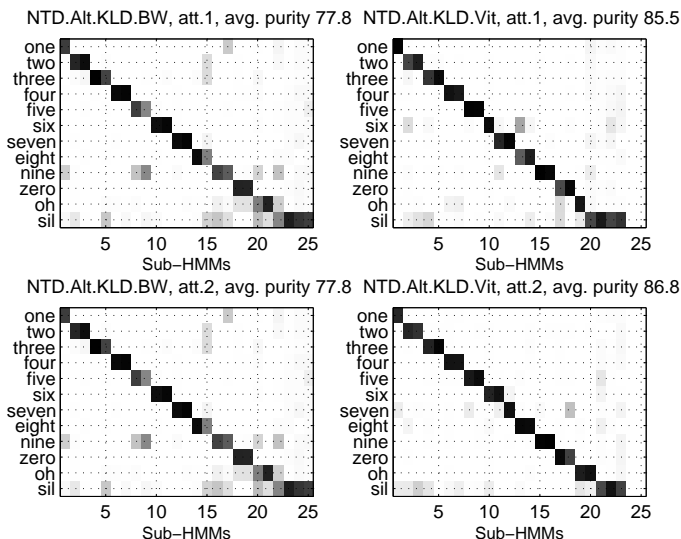


Figure 6.11: Mapping matrices from the Viterbi training and the NTD.Alt.Vit training of KLDHMM. The parameters and initializations are the same as the BW training of SCDHMM and KLDHMM. The Viterbi training is with 25 EM passes, while the NTD.Alt.Vit training methods are with 5 NTD passes and 5 Viterbi passes.

isolated examples will be used in conjunction with a large amount of unlabeled continuous speech.

Secondly, we will further compare NTD.Reg.BW learning for DDHMMs, SCDHMMs and KLDHMMs and compare their performance with this improved initialization method.

The task is to learn words with few of labeled utterances which is operated as follows: initialize the algorithm by some utterances of each word where one utterance only contains one word, train the algorithm using continuous speech which contains the words and evaluate the learned word models by recognizing the words in unseen utterances. In the following experiments on this task, each digit is initialized by two utterances: one from some male speaker and the other from some female speaker. The utterance is uniformly cut into L pieces to initialize the L states of the sub-HMM of the digit the utterance contains. The initializations from the male and female utterances of the same digit are accumulated. We call the initialization *oracle initialization*.

Table 6.4: The performance on TIDIGITS of NTD.Reg.BW training of DDHMM. $M = 1000$ Gaussians, $R = 12$ sub-HMMs, $L=10$ states for each sub-HMM and 25 EM iterations. First column: Baum-Welch training, second and third column: NTD.Reg.BW with listed regularization weight.

	BW	$\lambda = 10^{-2}$	$\lambda = 10^0$
log-likelihood ($\times 10^6$)	-5.95	-5.93	-6.17
# Sub.	71	21	20
# Ins.	38	8	12
# Del.	38	16	13
# total	3257	3257	3257
WER (%)	4.51	1.38	1.38
optimal word penalty	-300	-60	-45

Table 6.5: The performance on TIDIGITS of NTD.Reg.BW training of SCDHMM. $M = 1000$ Gaussians, $R = 12$ sub-HMMs, $L=10$ states for each sub-HMM, 25 EM iterations. First column: Baum-Welch training, second and third column: NTD.Reg.BW with listed regularization weight.

	BW	$\lambda = 10^{-1}$	$\lambda = 10^0$
log-likelihood ($\times 10^6$)	-5.58	-5.58	-5.61
# Sub.	11	9	63
# Ins.	4	2	25
# Del.	9	9	154
# total	3257	3257	3257
WER (%)	0.74	0.61	7.43
optimal word penalty	-30	-30	-50

The results of DDHMM, SCDHMM and KLDHMM with oracle initializations are reported in Table 6.4, Table 6.6 and Table 6.5. For SCDHMM, the Gaussians are NOT updated to keep the data tensor unchanged during the iterations of NTD.Reg.BW, otherwise the objective function is not well-defined.

The models are evaluated by the lowest word error rate (WER) when optimizing over the word entrance penalty (WEP) (line search from -300 to 100 with a stepsize 5) on the test data. All the learning algorithms are observed to converge. For DDHMM, the NTD.Reg.BW outperforms BW significantly. However, for KLDHMM and SCDHMM, NTD.Reg.BW seems unable to improve over BW significantly. For the learning task, NTD.Reg.BW training of

Table 6.6: The performance on TIDIGITS of NTD.Reg.BW training of KLDHMM. $M = 1000$ Gaussians, $R = 12$ sub-HMMs, $L=10$ states for each sub-HMM, 25 EM iterations.

	BW	$\lambda = 10^{-1}$	$\lambda = 10^0$
log-likelihood ($\times 10^6$)	-5.29	-5.32	-5.45
# Sub.	11	10	11
# Ins.	4	6	8
# Del.	9	8	11
# total	3257	3257	3257
WER (%)	0.74	0.74	0.92
optimal word penalty	-20	-15	-10

SCDHMM holds the best performance. As future work, more research towards learning words from a few of labeled data and a large amount of unlabeled data as the problem defined in [3] deserves more attention.

6.5 Conclusion

In this chapter, a non-negative Tucker decomposition (NTD) model was presented for the unsupervised training of an HMM for sequential pattern discovery, segmentation and recognition. Two training schemes to jointly learn the parameters of the NTD and the HMM were proposed: NTD-regularized Baum-Welch training of the HMM and alternated training of the NTD and the HMM. For DDHMM training, the second method outperformed the first method, Baum-Welch training and simulated annealing on the task of spoken pattern discovery from real-word speech data. The strength of combined NTD and HMM learning is that NTD explains the data with co-occurrence patterns and therewith provides a global view on the data, while the HMM models the local sequential aspects. A parameter sensitivity study showed some prior knowledge of the number of patterns and their length is required, but that the exact values are not critical. The algorithms are extended to SCDHMM and KLDHMM by which vocabularies patterns with high purity are discovered.

Chapter 7

Conclusion

In this chapter, we give an overview of the original contributions of the dissertation and discuss future directions.

7.1 Original Contributions

This work aims at learning methods for vocabulary acquisition by investigating new speech representations. Based on bag-of-features (BoF) representations of speech, algorithms for non-negative matrix factorizations (NMF) with constraints are proposed and evaluated for accurate representations of spoken words. In summary, the main contribution of this work is having improved BoF representations in the following aspects.

- Multi-view BoF.
The conventional BoF representation is a frequency count of feature occurrence. In Chapter 3, we have improved the BoF representation by using *multiple views (or groups) of features*. Bags of high-order statistics, like *co-occurrences/triplets of features*, are also implemented in the NMF learning framework. Those representations have shown good performance on vocabulary acquisition.
- BoF on a manifold.
Some feature relations are not easily represented with co-occurrences or triplets because of the exponential nature of high order statistics, because features show similarity or proximity (e.g. in images). Similarity of

features can be reflected in an adjacency matrix of a graph whose vertices are the features. The adjacency matrix is solely determined by the graph Laplacian which is the discretized manifold Laplacian. Therefore, the proposed graph regularized NMF in Chapter 4 actually looks for BoF representations respecting the structure of features on the manifold. The graph regularized NMF is a general tool and has manifested its good performance on the analysis of speech, documents and images.

- Bag of BoFs.
Hidden states play an important role in linking acoustic observations and linguist labels (phones or words) in speech recognizers using HMM. In Chapter 5, sub-word units, similar to hidden states, are learned by non-negative matrix tri-factorization (NMTF). Consequently, vocabulary patterns discovered by NMF are in a form of *bag of sub-word units* where every sub-word unit itself is in a BoF format. Hence, a word is actually represented in bag of BoFs. This representation yields better results on vocabulary acquisition than the one using raw acoustic observations in the sense of low unordered word error rates and fast word learning rates.
- Bag of (sub-)HMMs.
Inspired by the NMTF learning of Chapter 5, unsupervised learning of HMMs is investigated by combining HMM EM training and non-negative Tucker decomposition (NTD). In NTD, every utterance is decomposed into bag of sub-HMMs where a sub-HMM represents a spoken unit (a word in our example). Joint learning algorithms have been proposed for optimizing both NTD and HMM. The discovered sequential patterns have shown strong relations to words.

The above contributions are general tools for not only speech processing but could also be applied to other fields such as the analysis of text-based documents, video, gene sequences, gestures and handwritings.

7.2 Future Research Directions

7.2.1 An Incremental Learning Framework

An incremental learning framework should be designed towards the acquisition of large vocabulary. At the beginning, the robot first acquires a small set of words which are linked to other input modalities or grounding labels. When confronted with unseen data, the robot should be able to detect new recurring spoken patterns from continuous speech.

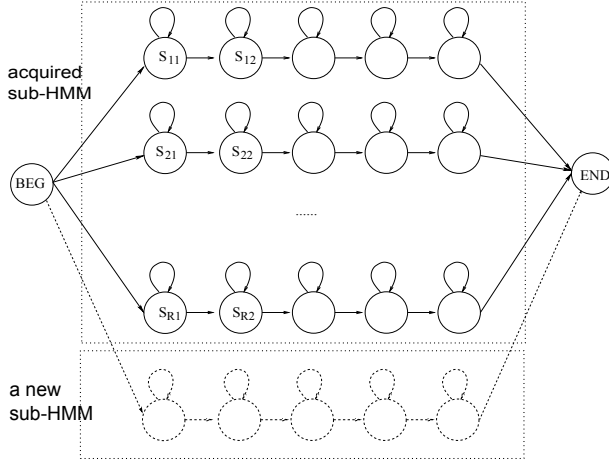


Figure 7.1: Incremental word learning by adding sub-HMMs. New spoken patterns are first detected, subsequently modeled by a sub-HMM and added to the overall HMM as a new branch.

This process can be realized by introducing new branches in the HMM of Figure 7.1. The sub-HMMs with full lines refer to the vocabulary the robot has obtained and the sub-HMMs with dashed lines refer to the new learned words which should be linked to some meaning later. Based on the alternative learning framework of NTD (realized by NMF and NMTF) and HMM, the newly included sub-HMMs can be first modeled by their bag of co-occurrences of observation symbols in $\mathbf{W}^{(new)}$ of Eq.(7.1) which is the incremental dictionary learning discussed in Chapter 2.

$$\mathbf{V} \approx [\mathbf{W}^{(old)} \ \mathbf{W}^{(new)}] \begin{bmatrix} \mathbf{H}^{(old)} \\ \mathbf{H}^{(new)} \end{bmatrix} \tag{7.1}$$

The representations of new words are in co-occurrences of observation symbols (like Gaussians in the previous chapters). NMTF is subsequently applied to learn a sub-HMM for each column of $\mathbf{W}^{(new)}$. The learned sub-HMMs are subsequently added to the HMM as illustrated in Figure 7.1. Alternative training of NMF, NMTF and HMM can be operated to refine the newly included sub-HMMs.

A grounded version of the above incremental process can be utilized to decide if adding new sub-HMMs or not:

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{V} \end{bmatrix} \approx \begin{bmatrix} \mathbf{Q}^{(\text{old})} & \mathbf{Q}^{(\text{new})} \\ \mathbf{W}^{(\text{old})} & \mathbf{W}^{(\text{new})} \end{bmatrix} \begin{bmatrix} \mathbf{H}^{(\text{old})} \\ \mathbf{H}^{(\text{new})} \end{bmatrix} \quad (7.2)$$

where \mathbf{G} is the grounding matrix of the new input data and $\mathbf{Q}^{(\text{old})}$ and $\mathbf{Q}^{(\text{new})}$ are mapping matrices between the (old and new) sub-HMMs and the ground truth labels. In the incremental learning, $\mathbf{Q}^{(\text{new})}$, $\mathbf{W}^{(\text{new})}$, $\mathbf{H}^{(\text{old})}$ and $\mathbf{H}^{(\text{new})}$ are updated. The number of new sub-HMMs, i.e. the number of columns of $\mathbf{Q}^{(\text{new})}$ or $\mathbf{W}^{(\text{new})}$, is determined by \mathbf{G} . Therefore, for this model to work the system has to handle uncertainties in the grounding information (i.e. to estimate \mathbf{G}) and be able to determine if an acoustic pattern is novel, i.e. unseen, or already represented.

With a large vocabulary, the computing of NMF and the storage of $\mathbf{W}^{(\text{old})}$ and $\mathbf{W}^{(\text{new})}$ will have high computational complexity. Low rank storage of $\mathbf{W}^{(\text{old})}$ and $\mathbf{W}^{(\text{new})}$ via NMTF is necessary. Sparse activations of words should be considered when solving Eq.(7.1). Usually only a couple of words can be activated in an utterance and we only need to consider the relevant ones for solving Eq.(7.1). Towards a small but information-rich data matrix \mathbf{V} for acquiring new words, techniques for out-of-vocabulary detection [43][82] can be useful for detecting and training with the speech segments which only contain unknown words.

7.2.2 Hierarchical HMMs

Inspired by the success of deep learning in ASR [77][127], a hierarchical HMM with multiple layers can be constructed as illustrated in Figure 7.2. The unsupervised learning algorithms are useful for the initialization of the hierarchical system.

The bottom layer is a set of Gaussians from a Gaussian mixture model (GMM) and the first layer contains hidden states which are linear combinations of the Gaussians. The two layers can be initialized by the unsupervised training of (semi-)CDHMM as presented in Chapter 6. For the layers higher than layer 1, a hidden state in layer h is a linear combination of the hidden states in layer $h-1$ with weights $\Pr(S_{k^{[h-1]}}^{[h-1]} | S_{k^{[h]}}^{[h]})$ where $k^{[h-1]}$ and $k^{[h]}$ are the state indices of layer $h-1$ and layer h correspondingly. The weights can be learned by unsupervised training of a KLDHMM between layer h and layer $h-1$ as presented in Chapter 6. This is because the KLD metric is suitable for comparing the multinomial distributions in the posteriorgram and of the emission probabilities the hidden

states. For state $S_{k^{[h]}}^{[h]}$, the observation likelihood on frame \mathbf{O}_t can be computed by,

$$\begin{aligned} & \Pr(\mathbf{O}_t | S_{k^{[h]}}^{[h]}) \\ = & \sum_{m=1}^M \Pr(\mathbf{O}_t | \mathcal{G}_m) \sum_{k^{[1]}} \Pr(\mathcal{G}_m | S_{k^{[1]}}^{[1]}) \\ & \sum_{h'=2}^{h-1} \sum_{k^{[h']}} \Pr(S_{k^{[h'-1]}}^{[h'-1]} | S_{k^{[h']}}^{[h']}) \Pr(S_{k^{[h-1]}}^{[h-1]} | S_{k^{[h]}}^{[h]}), \end{aligned} \quad (7.3)$$

where M is the number of Gaussians in the bottom layer, h is the number of layers with hidden states. In this framework, the transitions between hidden states of layer h' are not utilized when training HMMs on the layers higher than h' , which makes the computational complexity lower than the hierarchical HMM proposed in [32].

To conclude, we think the proposed methods would be able to work as general tools for the research in sequential data analysis. The methods can be expanded towards the acquisition of large vocabulary and can also be deepened towards hierarchical representations of speech as new acoustic models for ASR.

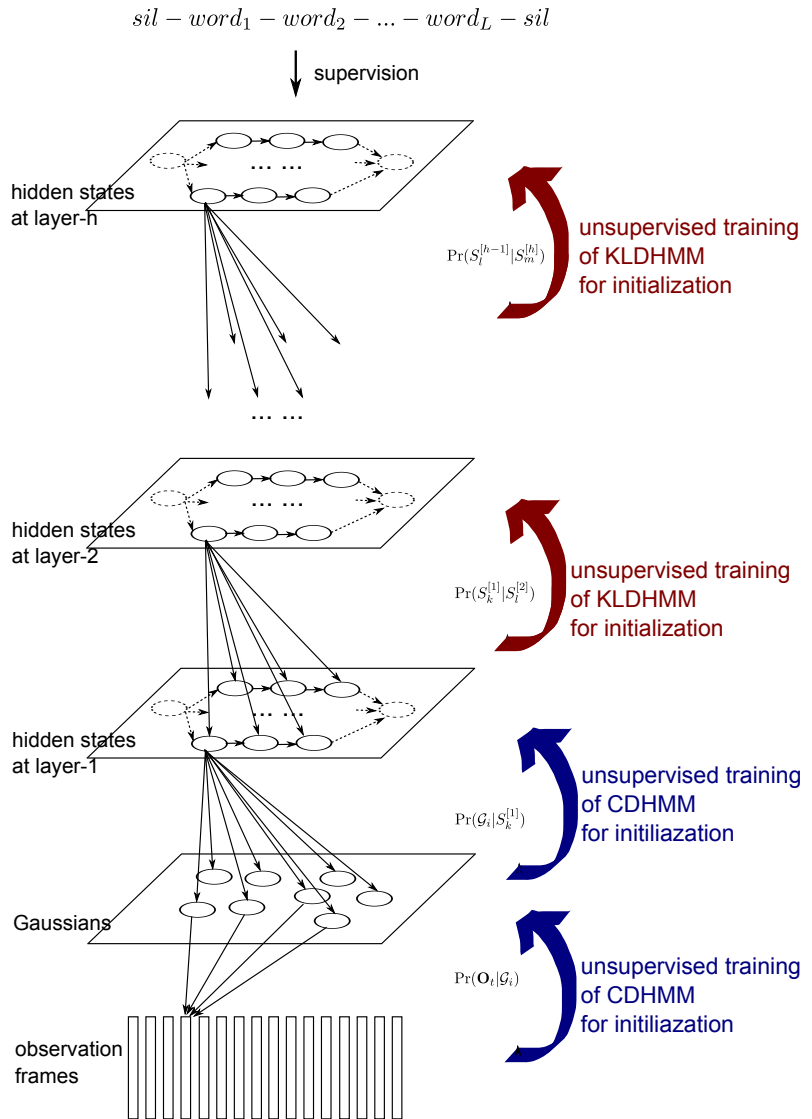


Figure 7.2: A hierarchical HMM for speech recognition. The model has multiple layers of hidden states. CDHMM and KLDHMM can be implemented for initializing the bottom layers. Supervision is only available for the top layer which is trained with transcribed data.

Appendix A

Convergence of the Contraction Mapping in L1GNMF

Let $\alpha_i = 2\lambda\mathbf{D}_{ii}$ and $\gamma_i = \mathbf{B}_{ik} + 2\lambda\mathbf{E}_{ik}$, the problem to prove that Eq.(4.16) is the solution of Eq. (4.13) boils down to proving that Eq. (A.3) is the solution of Eq. (A.1) with unknown variables $\{x_i\}$.

$$\alpha_i x_i^2 + \left(\sum_s (\gamma_s - \alpha_s x_s^2)\right)x_i - \gamma_i = 0. \quad (\text{A.1})$$

Eq. (A.2) is used to solve the above equation iteratively for any x_i .

$$\alpha_i (x_i^{(n+1)})^2 + \left(\sum_s (\gamma_s - \alpha_s (x_s^{(n)})^2)\right)x_i^{(n+1)} - \gamma_i = 0. \quad (\text{A.2})$$

So given the solution of x_i at the n -th iteration, $x_i^{(n)}$, the positive root of the quadratic equation, is chosen as the updated $x_i^{(n+1)}$, where $\beta^{(n)} = \sum_s (\gamma_s - \alpha_s (x_s^{(n)})^2)$.

$$x_i^{(n+1)} = \frac{-\beta^{(n)} + \sqrt{(\beta^{(n)})^2 + 4\alpha_i \gamma_i}}{2\alpha_i} \quad (\text{A.3})$$

The following four lemmas are equivalent.

- Lemma 1** $\{x_i^{(n)}\}$ converges.
- Lemma 2** $\lim_{n \rightarrow \infty} x_i^{(n)}$ is the solution of Equation (A.1).
- Lemma 3** $\lim_{n \rightarrow \infty} \sum_i x_i^{(n)} = \sum_s x_s = 1$.
- Lemma 4** $\{\beta^{(n)}\}$ converges.

It is obvious to derive Lemma 2 from Lemma 1 by computing the limit of Eq. (A.2). By summing all the equations (w.r.t. the index i) of Eq. (A.1), we obtain $\sum_s x_s = 1$. Hence Lemma 3 holds given Lemma 2. By summing all the Eq.s (A.2) w.r.t. i and using the definition of $\beta^{(n)}$, we obtain the relations between $\beta^{(n+1)}$ and $\beta^{(n)}$,

$$\beta^{(n+1)} = \beta^{(n)} \sum_s x_s^{(n+1)}, \tag{A.4}$$

which implies that Lemma 4 holds given Lemma 3. Lemma 4 to Lemma 1 can be easily seen from Eq. (A.3). So we only need to prove any of the four lemmas to prove them all.

Lemma 5 $\{\beta^{(n)}\}$ is monotonous.

From Eq. (A.3), we obtain,

$$\begin{aligned} & x_i^{(n+1)} - x_i^{(n)} \\ = & -\frac{1}{2\alpha_i} \left(1 - \frac{\beta^{(n)} + \beta^{(n-1)}}{\sqrt{(\beta^{(n)})^2 + 4\alpha_i\gamma_i} + \sqrt{(\beta^{(n-1)})^2 + 4\alpha_i\gamma_i}} \right) \\ & (\beta^{(n)} - \beta^{(n-1)}) \end{aligned} \tag{A.5}$$

This implies $(x_i^{(n+1)} - x_i^{(n)})(\beta^{(n)} - \beta^{(n-1)}) \leq 0$. The difference of two successive $\beta^{(n)}$'s yields,

$$\beta^{(n+1)} - \beta^{(n)} = \sum_s \alpha_s (x_i^{(n)} + x_i^{(n+1)})(x_i^{(n)} - x_i^{(n+1)}). \tag{A.6}$$

So by using $(x_i^{(n+1)} - x_i^{(n)})(\beta^{(n)} - \beta^{(n-1)}) \leq 0$, we have proved $(\beta^{(n+1)} - \beta^{(n)})(\beta^{(n)} - \beta^{(n-1)}) \geq 0$. Hence Lemma 5 holds.

By summing Eq. (A.5) w.r.t. i , we see the inverse trend of the change of $\beta^{(n)}$ with respect to the change of $\sum_s x_s^{(n)}$,

$$\left(\sum_s x_s^{(n+1)} - \sum_s x_s^{(n)} \right) (\beta^{(n)} - \beta^{(n-1)}) \leq 0. \tag{A.7}$$

Also, $\beta^{(n)}$ doesn't change its sign during the iterations from Eq. (A.4). We hereby discuss the convergence in four cases.

Case 1 $\beta^{(n)}\{\geq 0, \uparrow \text{ (increasing)}\}$ $\sum_s x_s^{(n+1)} = \frac{\beta^{(n+1)}}{\beta^{(n)}} \geq 1$, and from Eq. (A.7) $\sum_s x_s^{(n+1)} \leq \sum_s x_s^{(n)}$, hence $\{\sum_s x_s^{(n)}\}$ converges to a constant $\eta \geq 1$. If η would be greater than 1, $\beta^{(n)}$ would have to tend to ∞ . But $\beta^{(n)} \leq \sum_s \gamma_s$ is bounded. So $\lim_{n \rightarrow \infty} \sum_n x_n = \eta = 1$. We have proved Lemma 2.

Case 2 $\beta^{(n)}\{\geq 0, \downarrow\}$ $\{\beta^{(n)}\}$ converges, and Lemma 4 holds.

Case 3 $\beta^{(n)}\{\leq 0, \uparrow\}$ $\{\beta^{(n)}\}$ converges, and Lemma 4 holds.

Case 4 $\beta^{(n)}\{\leq 0, \downarrow\}$ $\sum_s x_s^{(n+1)} = \frac{\beta^{(n+1)}}{\beta^{(n)}} \geq 1$ and since $\beta^{(n)} \leq 0$, Eq. (A.7) implies that $\sum_s x_s^{(n+1)} \geq \sum_s x_s^{(n)}$. In this case, if $\sum_s x_s^{(n)} > 1$ for some n , and $\beta^{(n)}$ will diverge. We will now show that an additional normalization step to ensure $\sum_s x_s^{(n)} = 1$ will convert Case 4 into Case 1, Case 2 or Case 3, after which unnormalized iterations can continue.

Define $\tilde{x}_i^{(n)} = \frac{x_i^{(n)}}{\sum_s x_s^{(n)}}$, $\tilde{\beta}^{(n)} = \sum_s \gamma_s - \alpha_s (\tilde{x}_s^{(n)})^2$, thus the new equation to be solved is,

$$\alpha_i (x_i^{(n+1)})^2 + \tilde{\beta}^{(n)} x_i^{(n+1)} - \gamma_i = 0. \tag{A.8}$$

The solution is $x_i^{(n+1)} = \frac{-\tilde{\beta}^{(n)} + \sqrt{(\tilde{\beta}^{(n)})^2 + 4\alpha_i \gamma_i}}{2\alpha_i}$. By summing the equations w.r.t. i in Eq. (A.8), we obtain the relations between $\beta^{(n+1)} (= \sum_s (\gamma_s - \alpha_s (x_s^{(n+1)})^2))$ and $\tilde{\beta}^{(n)}$,

$$\beta^{(n+1)} = \tilde{\beta}^{(n)} \sum_s x_s^{(n+1)}. \tag{A.9}$$

It implies $\tilde{x}_i^{(n+1)} = x_i^{(n+1)} \frac{\tilde{\beta}^{(n)}}{\beta^{(n+1)}}$. The relation between two successive $\tilde{\beta}^{(n)}$ is,

$$\begin{aligned} & \tilde{\beta}^{(n+1)} - \beta^{(n+1)} \\ &= \sum_s (\gamma_s - \alpha_s (\tilde{x}_s^{(n+1)})^2) - \beta^{(n+1)} \\ &= \sum_s (\gamma_s - \alpha_s (x_s^{(n+1)} \frac{\tilde{\beta}^{(n)}}{\beta^{(n+1)}})^2) - \beta^{(n+1)} \\ &= \sum_s (\gamma_s - \alpha_s (x_s^{(n+1)})^2) - \beta^{(n+1)} \\ & \quad + \sum_s \alpha_s (x_s^{(n+1)})^2 (1 - (\frac{\tilde{\beta}^{(n)}}{\beta^{(n+1)}})^2) \\ &= -(\sum_s \alpha_s (x_s^{(n+1)})^2) \frac{\beta^{(n+1)} + \tilde{\beta}^{(n)}}{(\beta^{(n+1)})^2} (\tilde{\beta}^{(n)} - \beta^{(n+1)}) \\ &= \left(1 - \frac{\sum_s \gamma_s}{\beta^{(n+1)}}\right) \left(1 + \frac{\tilde{\beta}^{(n)}}{\beta^{(n+1)}}\right) (\tilde{\beta}^{(n)} - \beta^{(n+1)}) \end{aligned} \tag{A.10}$$

From Eq. (A.9), we know $1 + \frac{\tilde{\beta}^{(n)}}{\beta^{(n+1)}} \geq 1$ because $x_i^{(n+1)}$ is always non-negative. According to the definition of Case 4, for some n , $\tilde{\beta}^{(n)} \leq 0$ and $\beta^{(n+1)} \leq \tilde{\beta}^{(n)} \leq 0$, e.g. $\tilde{\beta}^{(0)} = \beta^{(0)} \leq 0$. If $\beta^{(n+1)} \leq \tilde{\beta}^{(n)} \leq 0$ is not true, Case 4 is converted into Case 3 ($0 \geq \beta^{(n+1)} \geq \tilde{\beta}^{(n)}$) or Case 1 and 2 ($\beta^{(n+1)} \geq 0$) where no normalization of $\{x_i^{(n+1)}\}$ is required. So $1 - \frac{\sum_s \gamma_s}{\beta^{(n+1)}} \geq 1$ and $\tilde{\beta}^{(n)} - \beta^{(n+1)} \geq 0$ hold in Case 4. By making further derivation of Eq. (A.10), we obtain a simple relation between $\tilde{\beta}^{(n+1)} - \beta^{(n+1)}$ and $\tilde{\beta}^{(n)} - \beta^{(n+1)}$ in Case 4,

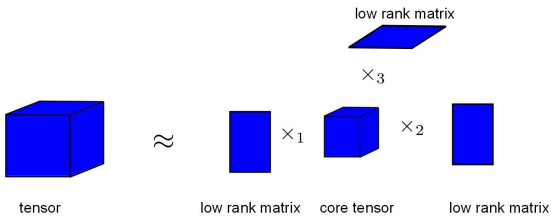
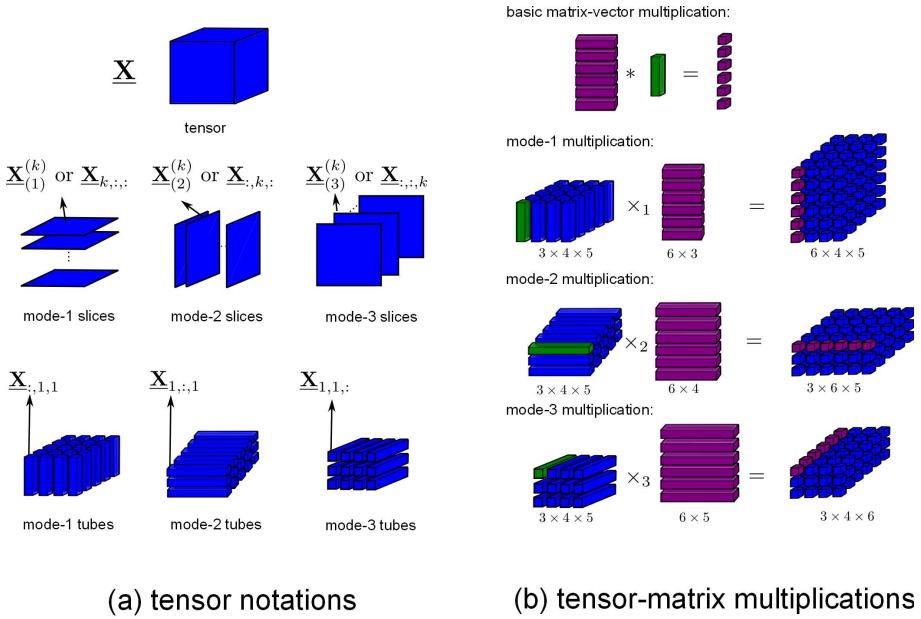
$$\begin{aligned} & \tilde{\beta}^{(n+1)} - \beta^{(n+1)} \\ = & \left(1 - \frac{\sum_s \gamma_s}{\beta^{(n+1)}}\right) \left(1 + \frac{\tilde{\beta}^{(n)}}{\beta^{(n+1)}}\right) (\tilde{\beta}^{(n)} - \beta^{(n+1)}) \\ \geq & \tilde{\beta}^{(n)} - \beta^{(n+1)} \end{aligned} \tag{A.11}$$

Hence $\tilde{\beta}^{(n+1)} \geq \tilde{\beta}^{(n)}$. So by transforming $\{\beta^{(n)}\}$ to $\{\tilde{\beta}^{(n)}\}$, the normalization of $\{x_i^{(n)}\}$ actually switches Case 4 to Case 3 if $\tilde{\beta}^{(n)} \leq 0, \forall n$ or to Case 1 and 2 if $\tilde{\beta}^{(n')} \geq 0, \exists n'$. The convergence is ensured for any case by using the algorithm in Table 4.1.

Appendix B

Tensor Notations and Operations

We consider 3-dimensional tensors in this paper, i.e. an array of numbers with three slices. The dimensions of the array are also referred to as *modes*. Mode-1, mode-2 and mode-3 slices and tubes are shown in Figure B.1(a). The multiplications of a tensor and a matrix are illustrated in Figure B.1(b). Mode- i multiplication entails replacing a mode- i tube by the multiplication of the matrix and this mode- i tube in the resulting tensor.



(c) non-negative Tucker decomposition

Figure B.1: Tensor notations and operations.

Appendix C

Derivation of the Symmetric NMTF Algorithm

In this section, we derive the algorithm for solving $\mathbf{Z} \approx \mathbf{X}\mathbf{Y}\mathbf{X}^T$. To obtain the tri-factorization of the joint probabilistic distribution \mathbf{Z} where $\sum_{i,j} Z_{i,j} = 1$, the chosen cost function is the Kullback-Leibler divergence between \mathbf{Z} and $\mathbf{X}\mathbf{Y}\mathbf{X}^T$ with normalization constraints $\sum_i X_{i,k} = 1$ and $\sum_{k,k'} Y_{k,k'} = 1$.

$$\mathcal{F}(\mathbf{X}, \mathbf{Y}) = \sum_{i,j} Z_{i,j} \log \frac{Z_{i,j}}{(\mathbf{X}\mathbf{Y}\mathbf{X}^T)_{i,j}} \quad (\text{C.1})$$

To derive the updating algorithm of \mathbf{X} , an auxiliary function is defined firstly with respect to the variable \mathbf{X} with fixed \mathbf{Y} . $\mathcal{A}(\mathbf{X}, \mathbf{X}^t)$ is an auxiliary function for $\mathcal{F}(\mathbf{X})$ if

$$\mathcal{A}(\mathbf{X}, \mathbf{X}^t) \geq \mathcal{F}(\mathbf{X}), \quad \mathcal{A}(\mathbf{X}^t, \mathbf{X}^t) = \mathcal{F}(\mathbf{X}^t) \quad (\text{C.2})$$

where \mathbf{X}^t is the solution from the previous iteration. With the convexity of $-\log(\cdot)$, it is easy to see that \mathcal{A} in Eq.(C.3) is an auxiliary function of $\mathcal{F}(\mathbf{X})$ and that $\mathcal{A}(\mathbf{X}, \mathbf{X}^t)$ is a convex function.

$$\begin{aligned} & \mathcal{A}(\mathbf{X}, \mathbf{X}^t) \\ &= \sum_{i,j} \left[Z_{i,j} \log Z_{i,j} - \sum_{k,l} Z_{i,j} \frac{X_{i,k}^t Y_{k,l} X_{j,l}^t}{(\mathbf{X}^t \mathbf{Y} (\mathbf{X}^t)^T)_{i,j}} (\log(X_{i,k} Y_{k,l} X_{j,l}) - \log \frac{X_{i,k}^t Y_{k,l} X_{j,l}^t}{(\mathbf{X}^t \mathbf{Y} (\mathbf{X}^t)^T)_{i,j}}) \right] \end{aligned} \quad (\text{C.3})$$

So the optimization problem with respect to \mathbf{X} is,

$$\begin{aligned} \min_{\mathbf{X}} \quad & \mathcal{A}(\mathbf{X}, \mathbf{X}^t) \\ \text{s.t.} \quad & \sum_i X_{i,k} = 1 \end{aligned} \quad (\text{C.4})$$

By using the Lagrange multiplier $\lambda_k(\sum_i X_{i,k} - 1)$, the optimization problem is transformed to solve $\frac{\partial \mathcal{A}(\mathbf{X}, \mathbf{X}^t)}{\partial X_{i',k'}} + \lambda_k = 0$ from Eq.(C.5).

$$\begin{aligned} X_{i',k'}^t \sum_{i,k} \frac{Z_{i,i'}}{(X^t Y (X^t)^T)_{i,i'}} X_{i,k}^t Y_{k,k'} + \frac{Z_{i',i}}{(X^t Y (X^t)^T)_{i',i}} Y_{k',k} X_{i,k}^t + \lambda_k X_{i',k'}^t &= 0 \\ \text{s.t.} \quad \sum_{i'} X_{i',k'} &= 1 \end{aligned} \quad (\text{C.5})$$

By summing the equations w.r.t. i' , one obtains,

$$\lambda_k = - \sum_{i',k} \frac{Z_{i,i'}}{(X^t Y (X^t)^T)_{i,i'}} X_{i,k}^t Y_{k,k'} + \frac{Z_{i',i}}{(X^t Y (X^t)^T)_{i',i}} Y_{k',k} X_{i,k}^t. \quad (\text{C.6})$$

So the final update of $X_{i',k'}$ is,

$$X_{i',k'}^t = X_{i',k'}^t \frac{\sum_{i,k} \frac{Z_{i,i'}}{(X^t Y (X^t)^T)_{i,i'}} X_{i,k}^t Y_{k,k'} + \frac{Z_{i',i}}{(X^t Y (X^t)^T)_{i',i}} Y_{k',k} X_{i,k}^t}{\sum_{i,i',k} \frac{Z_{i,i'}}{(X^t Y (X^t)^T)_{i,i'}} X_{i,k}^t Y_{k,k'} X_{i',k'}^t + \frac{Z_{i',i}}{(X^t Y (X^t)^T)_{i',i}} Y_{k',k} X_{i,k}^t X_{i',k'}^t} \quad (\text{C.7})$$

The update of \mathbf{Y} is essentially the same as in NMF [65] by reformulating $\mathbf{Z} \approx \mathbf{X} \mathbf{Y} \mathbf{X}^T$ to be $\text{vec}(\mathbf{Z}^T) \approx (\mathbf{X} \otimes \mathbf{X}) \text{vec}(\mathbf{Y}^T)$. vec is to transform a matrix to a vector by concatenating its columns, and \otimes is the Kronecker product. Following the property from NMF, given $\sum_{\mu} (X \otimes X)_{\mu,\gamma} = 1$, $\sum_{\mu} (\text{vec}(Z^T))_{\mu} = \sum_{\kappa} (\text{vec}(Y^T))_{\kappa}$ holds. $\sum_{\mu} (X \otimes X)_{\mu,\gamma} = 1$ is guaranteed by $\sum_i X_{i,k} = 1$, thus $\sum_{k,l} Y_{k,l} = \sum_{i,j} Z_{i,j} = 1$. The update of \mathbf{Y} in the form of tri-factorization is given in Eq.(C.8) where \mathbf{Y}^t is from the previous iteration.

$$Y_{k,l} = Y_{k,l}^t \sum_{i,i'} \frac{Z_{i,i'}}{(X^t Y (X^t)^T)_{i,i'}} X_{i,k} X_{i',l} \quad (\text{C.8})$$

Bibliography

- [1] http://en.wikipedia.org/wiki/Language_acquisition#Emergentism. pages 2
- [2] ARADILLA, G., BOURLARD, H., AND MAGIMAL-DOSS, M. Using kl-based acoustic models in a large vocabulary recognition task. *Idiap-RR Idiap-RR-14-2008, IDIAP, 0 2008*. pages 136
- [3] AYLON CLEMENTE, I., HECKMANN, M., AND WREDE, B. Incremental word learning: Efficient hmm initialization and large margin discriminative adaptation. *Speech Communication 54* (Nov. 2012), 1029–1048. pages xvi, 18, 19, 166
- [4] BALLARD, D. H., AND YU, C. A multimodal learning interface for word acquisition. In *ICASSP (2003)*, pp. v–784–7 vol.5. pages 6
- [5] BAUM, L. E., PETRIE, T., SOULES, G., AND WEISS, N. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist. 41(1)* (1970), 164–171. pages 17, 150
- [6] BENGIO, Y. Learning deep architectures for ai. *Foundations and Trends in Machine Learning 2, 1* (2009), 1–127. pages 6, 137
- [7] BLEI, D. M., AND MORENO, P. J. Topic segmentation with an aspect hidden markov model. In *SIGIR (2001)*, pp. 343–348. pages 134
- [8] BLEI, D., N. A. J. M. Latent dirichlet allocation. *Journal of Machine Learning Research (2003)*. pages 30
- [9] BOURLARD, H., DUPONT, S., AND RIS, C. Multi-stream speech recognition. *CC AI The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology 15, 3* (1998), 215–234. pages xvii, 52

- [10] BOVES, L., AND DEN OS, E. Acquisition of communication and recognition skills (ACORNS). <http://www.acorns-project.org/>, 2006-2009. pages 47
- [11] BOVES, L., TEN BOSCH, L., AND MOORE, R. Acorns - towards computational modeling of communication and recognition skills. In *International Conference on Cognitive Informatics* (2007), pp. 349–356. pages 2, 4, 38, 47
- [12] BRANDL, H., JOUBLIN, F., WREDE, B., AND GOERICK, C. A self-referential childlike model to acquire phones, syllables and words from acoustic speech. In *7th International Conference on Development and Learning* (2008), IEEE, IEEE, pp. 31–36. pages xvi, 16, 17
- [13] BRIDLE, J., AND BROWN, M. A date-adaptive frame rate technique and its use in automatic speech recognition. In *Proceedings of Institute of Acoustics Autumn Conference* (1982), pp. C2.1–C2.6. pages 52
- [14] CAI, D. <http://www.cad.zju.edu.cn/home/dengcai/data/gnmf.html>. pages 83, 98, 99, 100
- [15] CAI, D., HE, X., HAN, J., AND HUANG, T. Graph regularized non-negative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), 1548–1560. pages 22, 76, 77, 78, 83, 89, 92, 93, 98
- [16] CANO, P., BATLLE, E., KALKER, T., AND HAITSMA, J. A review of audio fingerprinting. *VLSI Signal Processing* 41 (2005), 271–284. pages 6
- [17] CHRISTOUDIAS, C., URTASUN, R., AND DARRELL, T. Multi-view learning in the presence of view disagreement. In *Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)* (2008), pp. 88–96. pages 51
- [18] CHUNG, F. R. K. *Spectral graph theory*. American Mathematical Soc., Providence, RI 02904-2294 USA, 1997. pages 77
- [19] CICHOCKI, A., ZDUNEK, R., PHAN, A. H., AND AMARI, S. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009. pages 143
- [20] COOKE, M., AND SCHARENBERG, O. The interspeech 2008 consonant challenge. In *Interspeech* (2008). pages 53, 55, 56, 57

- [21] CYBENKO, G., AND CRESPI, V. Learning hidden markov models using non-negative matrix factorization. *IEEE Transactions on Information Theory* 57(6) (2011), 3963–3970. pages 110, 112, 113
- [22] C.YU, AND D.H.BALLARD. A multimodal learning interface for grounding spoken language in sensory perceptions. *ACM Transactions on Applied Perception* 1 (2004), 57–80. pages 2, 3
- [23] DING, C., HE, X., AND SIMON, H. D. On the equivalence of nonnegative matrix factorization and spectral clustering. 606–610. pages 28
- [24] DING, C., LI, T., AND JORDAN, M. I. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *ICDM* (2008), pp. 183–192. pages 79, 80
- [25] DING, C. H. Q., LI, T., AND PENG, W. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics and Data Analysis* 52, 8 (2008), 3913–3927. pages 30
- [26] DONOHO, D. L., AND STODDEN, V. When does non-negative matrix factorization give a correct decomposition into parts? In *NIPS* (2003). pages 39
- [27] DRIESEN, J. *Discovering words in speech using matrix factorization*. PhD thesis, ESAT,KULeuven, 2012. pages 37
- [28] DRIESEN, J., GEMMEKE, J. F., AND VAN HAMME, H. Weakly supervised keyword learning using sparse representations of speech. In *ICASSP* (2012), pp. 5145–5148. pages 6
- [29] DRIESEN, J., GEMMEKE, J. F., AND VAN HAMME, H. Weakly supervised keyword learning using sparse representations of speech. In *ICASSP* (2012), pp. 5145–5148. pages 142
- [30] DRIESEN, J., AND VAN HAMME, H. Modelling vocabulary acquisition, adaptation and generalization in infants using adaptive bayesian pls. *Neurocomputing* 74(11) (2011), 1874–1882. pages 35
- [31] FEVOTTE, C., AND CEMGIL, A. T. Nonnegative matrix factorizations as probabilistic inference in composite models. In *EUSIPCO* (2009). pages 77
- [32] FINE, S., AND SINGER, Y. The hierarchical hidden markov model: Analysis and applications. 41–62. pages 171

- [33] FINESSO, L., GRASSI, A., AND SPREIJ, P. Approximation of stationary processes by hidden markov models. *Math. Control Signals Syst.* 22 (2010), 1–22. pages 112
- [34] FRANK, M. C., GOODMAN, N. D., AND TENENBAUM, J. B. A bayesian framework for cross-situational word-learning. pages 2
- [35] G.AIMETTI, BOSCH, L., AND R.K.MOORE. The emergence of words: Modelling early language acquisition with a dynamic systems perspective. In *INTERSPEECH* (2009). pages 2, 13
- [36] GAO, S., TSANG, I. W.-H., CHIA, L.-T., AND ZHAO, P. Local features are not lonely - laplacian sparse coding for image classification. In *CVPR* (2010), pp. 3555–3561. pages 76, 97
- [37] GAUSSIER, E., AND GOUTTE, C. Relation between plsa and nmf and implications. In *SIGIR* (2005), pp. 601–602. pages 30
- [38] GHAHRAMANI, Z. Unsupervised learning. In *Advanced Lectures on Machine Learning* (2003), pp. 72–112. pages 6
- [39] GHOSHAL, A. Hidden markov models for automatic annotation and content-based retrieval of images and video. In *In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval* (2005), pp. 544–551. pages 18
- [40] GU, Q., DING, C., AND HAN, J. On trivial solution and scale transfer problems in graph regularized nmf. In *The 22nd International Joint Conference on Artificial Intelligence (IJCAI)* (2011), pp. 1288–1293. pages 76, 78
- [41] GUAN, N., TAO, D., LUO, Z., AND YUAN, B. Manifold regularized discriminative non-negative matrix factorization with fast gradient descent. *IEEE Transactions on Image Processing* 20 (July 2001), 2030–2248. pages 79
- [42] HAKKANI-TUR, D., TUR, G., RAHIM, M., AND RICCARDI, G. Unsupervised and active learning in automatic speech recognition for call classification. In *ICASSP* (2004), pp. I–429–32 vol.1. pages 6
- [43] HANNEMANN, M., KOMBRINK, S., KARAFIÁT, M., AND BURGET, L. Similarity scoring for recognizing repeated out-of-vocabulary words. In *INTERSPEECH* (2010), pp. 897–900. pages 170
- [44] HANT, J., AND ALWAN, A. A psychoacoustic-masking model to predict the perception of speech-like stimuli in noise. *Speech Communication* 40, 3 (2001), 291–313. pages 51

- [45] HE, X., OGURA, T., SATOU, A., AND HASEGAWA, O. Developmental word acquisition and grammar learning by humanoid robots through a self-organizing incremental neural network. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 37 (2007), 1357–1372. pages 2
- [46] HINTON, G. E. Learning multiple layers of representation. *Trends in Cognitive Sciences* 11 (2007), 428–434. pages 6, 103, 106, 137
- [47] HINTON, G. E., OSINDERO, S., AND TEH, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* 18 (July 2006), 1527–1554. pages 128
- [48] HOFFMAN, T. Probabilistic latent semantic analysis. In *Uncertainty in Artificial Intelligence* (1999). pages 30, 34
- [49] HOYER, P. O., AND DAYAN, P. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research* 5 (2004), 1457–1469. pages 75
- [50] HSU, D., KAKADE, S. M., AND ZHANG, T. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences* 78(5) (2011), 1460–1480. pages 110, 141
- [51] HU, J., BROWN, M. K., AND TURIN, W. Hmm based on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (1996), 1039–1045. pages 134
- [52] HUANG, X., ACERO, A., AND HON, H.-W. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001. pages 3, 45, 150
- [53] HUANG, X. D., HON, H. W., AND LEE, K. F. Large-vocabulary speaker-independent continuous speech recognition with semi-continuous hidden markov models. In *Proceedings of the workshop on Speech and Natural Language* (1989), HLT '89, pp. 276–279. pages 38, 41
- [54] IWAHASHI, N. Robots that learn language: Developmental approach to human-machine conversations. In *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication* (2006), pp. 143–167. pages xvi, 2, 16, 17
- [55] JANSEN, A., AND CHURCH, K. Towards unsupervised training of speaker independent acoustic models. In *INTERSPEECH* (2011), pp. 1693–1692. pages 142

- [56] JANSEN, A., CHURCH, K., AND HERMANSKY, H. Towards spoken term discovery at scale with zero resources. In *INTERSPEECH* (2010), pp. 1676–1679. pages 4, 142
- [57] JUSCZYK, P. W. How infants begin to extract words from speech. *Trends in Cognitive Sciences* 3 (1999), 323–328. pages 20
- [58] KEMP, T., AND WAIBEL, A. Unsupervised training of a speech recognizer using tv broadcasts. In *ICSLP* (1998), pp. 2207–2210. pages 4
- [59] K.F.LEE, AND H.W.HON. Large-vocabulary speaker-independent continuous speech recognition using hmm. In *ICASSP* (1988), pp. 123–126. pages 44, 60
- [60] KHARBE, A. *English Language and Literary Criticism*. Discovery Publishing House, 2009. pages 21
- [61] KIMBALL, O., KAO, C.-L., IYER, R., ARVIZO, T., AND MAKHOUL, J. Using quick transcriptions to improve conversational speech models. In *INTERSPEECH* (2004), pp. 2265–2268. pages 4
- [62] LAKSHMINARAYANAN, B., AND RAICH, R. Non-negative matrix factorization for parameter estimation in hidden markov models. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)* (2010), pp. 89–94. pages 110, 113
- [63] LAZEBNIK, S., SCHMID, C., AND PONCE, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR* (2006), pp. 2169–2178. pages 74
- [64] LEE, D. D., AND SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature* 401 (1999), 788–791. pages 6, 28, 77, 78
- [65] LEE, D. D., AND SEUNG, H. S. Algorithms for non-negative matrix factorization. In *NIPS* (2000), pp. 556–562. pages 26, 27, 39, 78, 79, 80, 82, 115, 146, 147, 180
- [66] LI, S. Z., HOU, X. W., ZHANG, H. J., AND CHENG, Q. S. Learning spatially localized, parts-based representation. In *CVPR* (2001), pp. 207–212. pages 76
- [67] LIN, C.-J. Projected gradient methods for nonnegative matrix factorization. *Neural Computation* 19 (2007), 2756–2779. pages 27

- [68] LOWE, D. Object recognition from local scale-invariant features. In *ICCV* (1999), pp. 1150–1157. pages 93
- [69] MACWHINNEY, B. *The Emergence of Language*. Lawrence Erlbaum Associates, 1999. pages 2
- [70] MARKOV, K., AND NAKAMURA, S. Never-ending learning with dynamic hidden markov network. In *INTERSPEECH* (2007), pp. 1437–1440. pages xvi, 16
- [71] MATSOUKAS, S., GAUVAIN, J.-L., ADDA, G., COLTHURST, T., KAO, C.-L., KIMBALL, O., LAMEL, L., LEFEVRE, F., MA, J., MAKHOUL, J., NGUYEN, L., PRASAD, R., SCHWARTZ, R., SCHWENK, H., AND XIANG, B. Advances in transcription of broadcast news and conversational telephone speech within the combined ears bbn/limsi system. *IEEE Transactions on Audio, Speech, and Language Processing* 14 (2006), 1541–1556. pages 4
- [72] MAVRIDIS, N., AND ROY, D. Grounded situation models for robots: where words and percepts meet. pp. 4690–4697. pages 2, 3
- [73] LE CERF, P., AND VAN COMPERNOLLE, D. A new variable frame rate analysis method for speech recognition. *IEEE Signal Processing Letters* 1 (1994), 185–187. pages 52
- [74] VAN GERVEN, M., CSEKE, B., OOSTENVELD, R., AND HESKES, T. Bayesian source localization with the multivariate laplace prior. In *NIPS* (2009), pp. 1901–1909. pages 75
- [75] VAN HAMME, H. Integration of asynchronous knowledge sources in a novel speech recognition framework. In *Proceedings of ITRW on Speech Analysis and Processing for Knowledge Discovery* (2008), vol. 401. pages 15, 32, 45, 52, 53, 142
- [76] VAN SEGBROECK, M., AND VAN HAMME, H. Unsupervised learning of time-frequency patches as a noise-robust representation of speech. *Speech Communication* 51 (2009), 1124–1138. pages 38
- [77] MOHAMED, A.-R., DAHL, G. E., AND HINTON, G. E. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech and Language Processing* 20, 1 (2012), 14–22. pages 170
- [78] MOHRI, M., PEREIRA, F., AND RILEY, M. Weighted finite-state transducers in speech recognition. *Computer Speech and Language* 16, 1 (2002), 69–88. pages 155

- [79] MØRUP, M., HANSEN, L. K., AND ARNFRED, S. M. Algorithms for sparse non-negative tucker. *Neural Computation* 20 (2008), 2112–2131. pages 143
- [80] MYSORE, G. J., SMARAGDIS, P., AND RAJ, B. Non-negative hidden markov modeling of audio with application to source separation. In *LVA/ICA* (2010), pp. 140–148. pages 74
- [81] NGIAM, J., CHEN, Z., KOH, P., AND NG, A. Y. Learning deep energy models. In *ICML* (2011), pp. 1105–1112. pages 103, 106
- [82] PARADA, C., SETHY, A., DREDZE, M., AND JELINEK, F. A spoken term detection framework for recovering out-of-vocabulary words using the web. In *INTERSPEECH* (2010), pp. 1269–1272. pages 170
- [83] PARK, A. S., AND GLASS, J. R. Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech and Language Processing* 16, 1 (2008), 186–197. pages 4, 13, 142
- [84] PAUL, D. B. Training of hmm recognizers by simulated annealing. In *ICASSP* (1985), pp. 13–16. pages 140, 152
- [85] PEDERSEN, J. S., AND HEIN, J. Gene finding with a hidden markov model of genome structure and evolution. *Bioinformatics* 19 (2003), 219–227. pages 134
- [86] POEPPPEL, D., IDSARDI, W. J., AND VAN WASSENHOVE, V. Speech perception at the interface of neurobiology and linguistics. *Philosophical Transactions of the Royal Society B: Biological Sciences* 363 (2008), 1071–1086. pages 20
- [87] RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2) (1989), 257–286. pages 134, 150
- [88] ROWEIS, S. <http://cs.nyu.edu/~roweis/code.html>, 1999. pages 152
- [89] ROY, D. Grounded spoken language acquisition: Experiments in word learning. *IEEE Transactions on Multimedia* 5(2) (2003), 197–209. pages xv, 2, 4, 13, 14, 35
- [90] SAFFRAN, J., WERKER, J., AND WERNER, L. The infant’s auditory world: Hearing, speech and the beginnings of language. *Handbook of Child Psychology, Volume 2: Cognition, Perception and Language* (2006), 55–108. pages 20

- [91] SAKOE, H., AND CHIBA, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing* 26 (1978), 43–49. pages 12
- [92] SALVI, G., MONTESANO, L., BERNARDINO, A., AND SANTOS-VICTOR, J. Language bootstrapping: learning word meanings from perception-action association. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 42 (2012), 660–671. pages 2, 3
- [93] SCHARENBERG, O., AND COOKE, M. Comparing human and machine recognition performance on a vcv corpus. In *Proceedings of the workshop on Speech Analysis and Processing for Knowledge Discovery* (2008), pp. CD-ROM. pages 51, 55, 56, 57, 59
- [94] SHASHANKA, M., RAJ, B., AND SMARAGDIS, P. Probabilistic latent variable models as non-negative factorizations. *Computational Intelligence and Neuroscience 2008* (2008), 1–8. pages 77
- [95] SIU, M.-H., GISH, H., LOWE, S., AND CHAN, A. Unsupervised audio patterns discovery using hmm-based self-organized units. In *INTERSPEECH* (2011), pp. 2333–2336. pages 141, 152
- [96] SMARAGDIS, P., SHASHANKA, M. V. S., RAJ, B., AND MYSORE, G. J. Probabilistic factorization of non-negative data with entropic co-occurrence constraints. In *ICA* (2009), pp. 330–337. pages 76
- [97] SONG, L., SIDDIQI, S. M., GORDON, G. J., AND SMOLA, A. J. Hilbert space embeddings of hidden markov models. In *ICML* (2010), pp. 991–998. pages 141
- [98] STOUTEN, V., DEMUYNCK, K., AND VAN HAMME, H. Discovering phone patterns in spoken utterances by non-negative matrix factorisation. *IEEE Signal Processing Letters* 15 (2008), 131–134. pages xvi, 15
- [99] SUN, M., AND VAN HAMME, H. Coding methods for the nmf approach to speech recognition and vocabulary acquisition. In *IMCIC* (2011). pages 42
- [100] SUN, M., AND VAN HAMME, H. Consonant recognition and articulatory feature classification from signal analysis at multiple time scales. In *IMCIC* (2011). pages 51
- [101] SUN, M., AND VAN HAMME, H. Image pattern discovery by using the spatial closeness of visual code words. In *ICIP* (2011), pp. 205–208. pages 97

- [102] SUN, M., AND VAN HAMME, H. A two-layer non-negative matrix factorization model for vocabulary discovery. In *Symposium on Machine Learning in Speech and Language Processing* (2011). pages 125
- [103] SUN, M., AND VAN HAMME, H. Unsupervised vocabulary discovery using non-negative matrix factorization with graph regularization. In *ICASSP* (2011), pp. 5152–5155. pages 86
- [104] SUN, M., AND VAN HAMME, H. Joint training of non-negative tucker decomposition and discrete density hidden markov models. *Computer Speech and Language* (2012), DOI: 10.1016/j.csl.2012.09.006. pages 152
- [105] SUN, M., AND VAN HAMME, H. Large scale graph regularized non-negative matrix factorization with ℓ_1 normalization based on kullback-leibler divergence. *IEEE Transactions on Signal Processing* 60(7) (2012), 3876–3880. pages 76
- [106] SUN, M., AND VAN HAMME, H. Tri-factorization learning of sub-word units with application to vocabulary acquisition. In *ICASSP* (2012), pp. 5177–5180. pages 119
- [107] SUNDARAM, S., AND BELLEGARDA, J. R. Latent perceptual mapping: a new acoustic modeling framework for speech recognition. In *INTERSPEECH* (2010), pp. 881–884. pages 141
- [108] TEN BOSCH, L., VAN HAMME, H., AND BOVES, L. A computational model of language acquisition: focus on word discovery. In *INTERSPEECH* (2008), pp. 2570–2573. pages 4
- [109] THAMBIRATNAM, K., AND SRIDHARAN, S. Rapid yet accurate speech indexing using dynamic match lattice spotting. *IEEE Transactions on Audio, Speech and Language Processing* 15 (2007), 346–357. pages 6
- [110] TJOA, S. K., AND LIU, K. J. R. Multiplicative update rules for nonnegative matrix factorization with co-occurrence constraints. In *ICASSP* (2010), pp. 449–452. pages 76
- [111] TUYTELAARS, T., LAMPERT, C. H., BLASCHKO, M. B., AND BUNTINE, W. Unsupervised object discovery: A comparison. *International Journal of Computer Vision* 88, 2 (2010), 284–302. pages 93
- [112] VALLABHA, G., MCCLELLAND, J., PONS, F., WERKER, J., AND AMANO, S. Unsupervised learning of vowel categories from infant-directed speech. *Proceedings of the National Academy of Sciences* 104 (2007), 13273. pages 6

- [113] VAN HAMME, H. Hac-models: a novel approach to continuous speech recognition. In *INTERSPEECH* (2008), pp. 2554–2557. pages xvi, 2, 4, 15, 34, 39, 41, 42, 74, 141, 142
- [114] VANLUYTEN, B., WILLEMS, J., AND MOOR, B. D. Structured nonnegative matrix factorization with applications to hidden markov realization and clustering. *Linear Algebra and Its Applications* 429 (2008), 1409–1424. pages 110
- [115] VARADARAJAN, B., KHUDANPUR, S., AND DUPOUX, E. Unsupervised learning of acoustic sub-word units. In *ACL* (2008), pp. 165–168. pages 142
- [116] VEDALDI, A., AND FULKERSON, B. VLFeat: An open and portable library of computer vision algorithms. <http://www.vlfeat.org/>, 2008. pages 97
- [117] VIRTANEN, T. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech and Language Processing* 15, 3 (2007), 1066–1074. pages 75, 79
- [118] VITERBI, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13 (1967), 260–269. pages 131
- [119] WALLACH, H. M. Topic modeling: beyond bag-of-words. In *NIPS 2005 Workshop on Bayesian Methods for Natural Language Processing* (2005). pages 74
- [120] WANG, C., SONG, Z., YAN, S., ZHANG, L., AND ZHANG, H. Multiplicative nonnegative graph embedding. In *CVPR* (2009), pp. 389–396. pages 76
- [121] WANG, X., AND GRIMSON, E. Spatial latent dirichlet allocation. In *NIPS* (2007). pages 97
- [122] YANG, J., YAN, S., FU, Y., LI, X., AND HUANG, T. Non-negative graph embedding. In *CVPR* (2008), pp. 1–8. pages 76
- [123] YANG, Z., AND OJA, E. Quadratic non-negative matrix factorization. *Pattern Recognition* (2011), 1–11. pages 110
- [124] YOO, J., AND CHOI, S. Probabilistic matrix tri-factorization. In *ICASSP* (2009), pp. 1553–1556. pages 111, 116

- [125] YOUNG, S. J., KERSHAW, D., ODELL, J., OLLASON, D., VALTCHEV, V., AND WOODLAND, P. *The HTK Book Version 3.4*. Cambridge University Press, 2006. pages 3, 38, 56, 65, 69, 123, 139
- [126] YU, C., AND BALLARD, D. H. A multimodal learning interface for grounding spoken language in sensory perceptions. *ACM Transactions on Applied Perception 1* (2004), 57–80. pages 6
- [127] YU, D., AND DENG, L. Deep-structured hidden conditional random fields for phonetic recognition. In *INTERSPEECH* (2010), pp. 2986–2989. pages 6, 170
- [128] ZHANG, T., FANG, B., TANG, Y. Y., HE, G., AND WEN, J. Topology preserving non-negative matrix factorization for face recognition. *IEEE Transactions on Image Processing 17*, 4 (2008), 574–584. pages 76, 79
- [129] ZHANG, Y., AND GLASS, J. Towards multi-speaker unsupervised speech pattern discovery. In *ICASSP* (2010), pp. 4366–4369. pages 4, 13, 142
- [130] ZHUANG, X., HUANG, J. T., AND HASEGAWA-JOHNSON, M. Speech retrieval in unknown languages: a pilot study. In *Proceedings of the Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies* (2009), CLIAWS3 '09, pp. 3–11. pages 4

Curriculum Vitae



Meng Sun was born on 1 December 1984, Shandong, China. He received his bachelor degree of science in applied mathematics from the Department of Mathematics and System Science, National University of Defense Technology of China in 2006. From 2006 to 2008, he studied as a master-doctoral student in the same department. In December 2008, he joined the speech group of ESAT-PSI of KULeuven as a PhD candidate with the financial support from the China Scholarship Council and KULeuven. He has been working on the EU project ACORNS and the KULeuven project VASI. His research interests are speech recognition, machine learning, data mining and mathematical modeling.

List of Publications

Articles in International Journals

- [1] **Meng Sun**, Hugo Van hamme. *Joint training of non-negative Tucker decomposition and discrete density hidden Markov models*. Computer Speech and Language, in press, doi: 10.1016/j.csl.2012.09.006.
- [2] **Meng Sun**, Hugo Van hamme. *Large scale graph regularized non-negative matrix factorization with l1 normalization based on Kullback-Leibler divergence*. IEEE Transactions on Signal Processing, pp. 3876-3880, vol.60(7), 2012.

Articles in International Conferences

- [1] **Meng Sun**, Hugo Van hamme. *Tri-factorization learning of sub-word units with application to vocabulary acquisition*. In Proc. International Conference on Acoustics, Speech and Signal Processing, pp. 5177-5180, Kyoto, Japan, March 2012.
- [2] **Meng Sun**, Hugo Van hamme. *A two-layer non-negative matrix factorization model for vocabulary discovery*. Symposium on machine learning in speech and language processing, Bellevue, Washington, USA, June 2011.
- [3] **Meng Sun**, Hugo Van hamme. *Unsupervised vocabulary discovery using non-negative matrix factorization with graph regularization*. In Proc. International Conference on Acoustics, Speech and Signal Processing, pp. 5152-5155, Prague, Czech Republic, May 2011.

- [4] **Meng Sun**, Hugo Van hamme. *Image pattern discovery by using the spatial closeness of visual code words*. In Proc. International Conference on Image Processing, pp. 205-208, Brussels, Belgium, September 2011.
- [5] **Meng Sun**, Hugo Van hamme. *Coding methods for the NMF approach to speech recognition and vocabulary acquisition*. In Proc. International Multi-Conference on Complexity, Informatics and Cybernetics, Orlando, Florida, USA, March 2011. **Session's Best Paper Award**.
- [6] **Meng Sun**, Hugo Van hamme. *Consonant recognition and articulatory feature classification from signal analysis at multiple time scales*. In Proc. International Multi-Conference on Complexity, Informatics and Cybernetics, Orlando, Florida, USA, March 2011.

Arenberg Doctoral School of Science, Engineering & Technology

Faculty of Engineering

Department of Electrical Engineering (ESAT)

Center for Processing Speech and Images (PSI)

Kasteelpark Arenberg 10, box 2441

B-3001 Heverlee

KATHOLIEKE UNIVERSITEIT
LEUVEN

ASSOCIATIE
K.U. LEUVEN