EXPLORATION OF SPARSE REPRESENTATION TECHNIQUES IN LANGUAGE RECOGNITION

OM PRAKASH SINGH

EXPLORATION OF SPARSE REPRESENTATION TECHNIQUES IN LANGUAGE RECOGNITION

A

Thesis submitted

for the award of the degree of

DOCTOR OF PHILOSOPHY

By

OM PRAKASH SINGH

DEPARTMENT OF ELECTRONICS AND ELECTRIC ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781 039, ASSAM, INDIA

APRIL 2019

Certificate

This is to certify that the thesis entitled "**EXPLORATION OF SPARSE REPRESENTATION TECHNIQUES IN LANGUAGE RECOGNITION**", submitted by **Om Prakash Singh**, Roll No. 08610209, a research scholar in the *Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati*, for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion, has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dated:

Guwahati.

Dr. Rohit Sinha
Professor
Dept. of Electronics and Electrical Engg.
Indian Institute of Technology Guwahati
Guwahati - 781 039, Assam, India.

To

My dear parents

Shri S. P. Singh and Smt. S. Devi

for their love, support and encouragement

&

My guide

Prof. Rohit Sinha

for his guidance and support

&

My dear wife **Pinki**, and my lovely kids **Sankalp** and **Mokshit**

for the inspiration that you gave and the sacrifices you have all made during this period

Acknowledgements

First and foremost, I feel it as a great privilege in expressing my deepest and most sincere gratitude to my thesis supervisor Prof. Rohit Sinha, for his excellent guidance and constant encouragement throughout my study. His insightful feedbacks have helped me greatly in improving the quality of my thesis. My heartfelt thanks to you sir for the unlimited support and patience shown to me. I would particularly like to thank him for all his help in patiently and carefully correcting all my manuscripts.

I am very much thankful to my doctoral committee members Prof. S. R. Mahadeva Prasanna, Prof. Pradip Kr. Das and Dr. K. Karthik for their encouragement, valuable suggestions on my work and for sparing their precious time for evaluating the progress of my work. I would also like to thank Prof. P. K. Bora, Prof. S. Dandapat, Prof. Ratnajit Bhattacharjee, Prof. Shaik Rafi Ahamed, and Dr. A. Rajesh and other faculty members of the Department of Electronics and Electrical Engineering, IIT Guwahati for their care and support.

I am deeply indebted to All India Council for Technical Education, Govt. of India for providing me the research grant under Research Promotion Scheme (RPS), vide Grant No. 8023/RID/RPS-30/(NER)/2011-2012. The 2007 NIST Language recognition evaluation (LRE) speech corpus was procured from this grant. I am also very much thankful to Linguistic Data Consortium (LDC) for providing me the 2009 NIST LRE speech corpus under Data scholarship program in Spring 2016. I would like to thanks Dr. L. N. Sharma, Senior Scientific officer for his enormous help whenever required. I am also very much thankful to my seniors Mr. Sanjib Das, Dr. P. Krishnamoorthy, Dr. H. S. Jayanna, and Dr. Debadata Pati for their kind help and support. I am also grateful to all other technical and non-teaching staffs of the department for their help.

I would also like to express my sincere gratitude to Director, Sikkim Manipal Institute of Technology, Sikkim for allowing me to pursue the Ph.D. work at Department of Electrical and Electronics, IIT Guwahati, and grant the study leave whenever I required. I am also thankful to Prof. Rabindra Nath Bera, Head of the Department, Electronics and Communication Engineering, SMIT for his constant support and encouragement during my entire Ph.D. journey. My special thanks to all my colleagues who have shared my academic loads during the sanctioned leave from my parent institute.

I am indebted to my friend Dr. Haris BC for providing me the multi-lingual speech corpus, his practical advice, research discussion that helped to enrich the knowledge. I am thankful to my friends

Dr. Rajib Panigrahi, Dr. Kuntal Deka, Dr. Nagesh Ch, Dr. Ashish Kumar Namdeo, Dr. Sanjoy Mondal, Dr. Utkal V Mehta, Dr. Ramesh C. Mishra, Dr. Sayantan Hazra, Dr. Rajib Jana, Dr. Sunil Y., Dr. Gayadhar Pradhan, Dr. Syed Shahnawazuddin, Dr. Deepak K. T., Dr. Biswajit Dev Sharma, Dr. Nagraj Adiga, Dr. Rohan Kumar Das, Dr. Anurag Singh, Dr. Sibashankar Padhy, Dr. Banriskhem K Khonglah, Dr. Santosh Kumar Yadav, Dr. Kukil Khanikar, Mr. Mridupawan Sonowal, Mohammed Nasir Ansari, Mr. Nagendra Kumar, Mr. Sarfaraz Jelil, Mr. Ganji Sreeram, Mr. Ramesh Kumar Bhukya, Mr. Tilendra Choudhary, Ms. Vineeta Das, Mr. Eedara Prabhakar Rao, Mr. Samarjeet Das and all other members in the EMST lab for their love and care.

My deepest gratitude goes to my parents and sibling (Mr. Chandra Prakash Singh and Mrs. Sanju Sinha) for their blessings, continuous love and support throughout my studies. I specially thanks to my elder brother Mr. Uday Kumar Singh for his enormous love and sacrifices made to ensure my bright future. At last, but not the least, my special thanks go to my dear wife Pinki, and my lovely kids Sankalp and Mokshit for the inspiration that you gave and the sacrifices you have all made during this long journey.

Om Prakash Singh

Abstract

Language recognition (LR) refers to the task of identifying the language being spoken in a speech utterance with the help of machines. Following the success of the i-vector representation in speaker recognition task, it is also noted to yield state-of-the-art performances in LR task. The i-vectors form a popular low-rank representation of speech utterances and are derived from the Gaussian mixture model (GMM) mean supervectors using factor analysis. The i-vector extraction involves a low-rank "total variability matrix" which is hypothesized to incorporate both language and the session/channel variabilities. The session/channel variabilities are compensated by appropriate techniques like linear discriminant analysis (LDA), within class covariance normalization (WCCN), etc. The concept of total variability space as a single space is an extension of the joint factor analysis (JFA), where the language and the session/channel variabilities are modeled separately. In the recent years, sparse representation classification (SRC) has been successfully exploited in many pattern recognition tasks including LR. The existing SRC based LR approaches use an exemplar dictionary created by concatenating the i-vectors corresponding to the training utterances. The i-vector representation for the test utterance are then sparse coded over the exemplar dictionary. Based on the indices of the non-zero coefficients in the sparse coded vector (s-vector), the language of the spoken test utterance is determined. With increase in the number of training examples being available, the size of the exemplar dictionary also increases, and the computational burden of sparse coding over such highly redundant dictionary becomes quite prohibitive.

Towards addressing that challenge, this thesis explore various dictionary learning approaches for designing compact dictionaries for LR task. These compact dictionaries not only reduce the computational burden but also result in enhanced LR performances. For dictionary learning, the K-SVD, the D-KSVD, the LC-KSVD, and the online learning (OL) dictionary approaches employing l_0-norm, l_1-norm, and combined l_1 and l_2-norm regularizations

are investigated. Though the i-vector approach is found to be very successful, it suffers from high computational complexity involved in its extraction as well as high memory requirement in the storage of the total variability matrix. For addressing that, an ensemble of random subspaces of GMM mean supervector employing LDA followed by WCCN based session/channel compensation in the subspace has been investigated. The proposed approach does not require any learning for creating the subspaces and has very low memory requirements. In typical LR applications, the number of languages involved is not very large. As a result of that, unlike speaker recognition task, the overall complexity of the JFA based system turns out to be lower than that of the i-vector approach. Exploiting this fact, the combination of the proposed ensemble approach with JFA is also explored. Furthermore, the JFA compensated maximum a-posteriori (MAP) point estimates (i.e., latent vector) are themselves used as the spoken utterance representations. Compared to the i-vector, the JFA latent vector has considerably lower dimensionality. The sparse coding algorithms are then applied to the JFA latent vector, and the performances are compared with the i-vector based LR system.

We also proposed an ensemble of learned-exemplar (class-wise concatenation of the learned dictionary) approach in order to reduce the latency and to achieve some diversity gain. In this approach, multiple small-sized dictionaries are created, and each of the dictionaries is used for extracting the sparse representation of a particular test utterance. In this way, multiple s-vectors are obtained for that test utterance. The language-specific coefficients in each of the s-vectors are averaged to have multiple score vectors. Finally, these score vectors are used to decide the spoken language of an unknown utterance. The recently proposed bottleneck features based i-vector representation is also explored in the sparse representation framework.

Keywords: Sparse representation, GMM mean supervector, i-vector, JFA latent vector, learned dictionary, language recognition.

Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial neural network
ASR	Automatic speech recognition
BP	Basis pursuit
BPDN	Basis pursuit denoising
BNF	Bottleneck features
CDS	Cosine distance scoring
CF	Channel factor
CMN	Cepstral mean normalization
CMVN	Cepstral mean and variance normalization
CTS	Conversational telephone speech
DCT	Discrete cosine transform
D-KSVD	Discriminative-KSVD
DLSI	Dictionary learning with structured incoherence
DNN	Deep neural networks
EM	Expectation maximization
ENet	Elastic net
FA	False alarm
FAR	False acceptance rate
FDDL	Fisher discrimination dictionary learning
FFT	Fast Fourier Transform
FRR	False rejection rate
GB	Gaussian backend
GG	Generative Gaussian
GLDS	Generalized linear discriminant sequence

GMM	Gaussian mixture model
GMM-MMI	GMM-maximum mutual information
GMM-UBM	GMM-universal background model
G-PLDA	Gaussian-PLDA
HMM	Hidden Markov model
IDR	Identification rate
i-vector	Identity vector
JFA	Joint factor analysis
K-SVD	K-singular value decomposition
LAR	Least angle regression
Lasso	Least absolute shrinkage and selection operator
LC-KSVD	Label consistent-KSVD
LD	Language detection
LDA	Linear discriminant analysis
LF	Language factor
LID	Language identification
LPC	Linear predictive coefficients
LR	Language recognition
LRE	Language recognition evaluation
LSTM	Long short-term memory
MAP	Maximum *a-posteriori*
MFCC	Mel frequency cepstral coefficients
MLLR	Maximum likelihood linear regression
MLR	Multi-class logistic regression
MMI	Maximum mutual information
NAP	Nuisance attribute projection
NCA	Neighborhood component analysis
OGI	Oregon Graduate Institute
OL	Online learning
OLR	Oriental language recognition

OMP	Orthogonal matching pursuit
PCA	Principal component analysis
PLP	Perceptual linear prediction
PLDA	Probabilistic linear discriminant analysis
PLLR	Phone log-likelihood ratios
PPR	Parallel phone recognition
PRLM	Phone recognition followed by language modeling
PPRLM	Parallel PRLM
PTN	Phonetic temporal network
RASTA	Relative spectral processing
RBF	Radial basis function
RP	Random Projection
RR	Ridge Regression
RNN	Recurrent neural network
SDC	Shifted delta cepstral
SNR	Signal to noise ratio
SRC	Sparse representation classification
s-vector	Sparse representation vector
SVM	Support vector machine
TDNN	Time delay neural network
UBM	Universal background model
VAD	Voice activity detection
VOA	Voice of America
VOP	Vowel onset point
VQ	Vector quantization
VSM	Vector space modeling
VTLN	Vocal tract length normalization
WCCN	Within-class covariance normalization

1

Introduction

Contents

Language recognition (LR) refers to the task of determining the identity of the language spoken in a speech utterance with the help of machines. The LR task can be based on written or spoken resources. In this work, we will always refer LR in the context of spoken language, i.e. spoken language recognition. The LR can be broadly divided into two different variants: language identification (LID) and language detection (LD). In LID task, the language of the given spoken segment is determined from a set of target languages. While, in the LD task, it is determined whether the unknown speech segment belongs to the claimed language of interest or not, thus posing a two class problem. The language recognition task can be a closed-set or an open-set. In closed-set, the test speech segments are limited to target languages which are used in the training process. However, in the open-set task, the non-target languages are also included in testing. The industries like travel and tourism, telecommunication and emergency services typically require an interface in multiple languages, thus employ human resources who have a working knowledge of multiple languages. With the expansion of such industries and the need for providing round the clock services, the cost of maintaining such services becomes exorbitantly high. To address this challenge, a number of automated multi-lingual services have been developed in the past. The success of such multi-lingual services depends on the fast and accurate determination of the spoken language by the LR system. The LR is a promising technology for various multi-lingual speech processing applications like spoken language translation, multi-lingual speech recognition, and spoken document retrieval.

1.1 Perceptual cues

The perceptual cues that human listeners use to recognize the languages always form the source of inspiration for automatic spoken LR [5] tasks. It has been noted from the human listening experiments that perceptual cues [6, 7] can be broadly categorized into pre-lexical and the lexical semantics [5]. The pre-lexical group includes acoustic-phonetics, phonotactics, and prosodic information while lexical provides words and syntax based knowledge [1,8]. Acoustic-phonetic deals with the acoustic aspects of speech sounds. The information about distinct speech sounds (phones) and the frequency of occurrence of phones helps to recognize the spoken language. Phonotactics refers to the rules that govern the combination of different phones in a language. Prosody, in general, refers to suprasegmental features, such as stress, rhythm, and intonation. The lexical group deals with the words or morphemes and a syntactic system that governs how words and morphemes are combined to form phrases and utterances.

Among these perceptual cues, acoustic-phonetic and prosodic features are commonly used in LR task due to the simplicity of feature extraction as well as comparable performance to the lexical cues based LR system. The Mel frequency cepstral coefficients (MFCC) [9] and shifted delta cepstral (SDC) [10,11] features are commonly used to capture the acoustic information. However, the prosodic cues such as stress, rhythm, and intonation are captured by the energy contour, the syllable duration, and the pitch contour, respectively. The extracted features are a compact and efficient representation of the speech utterance while incorporating all the most important aspects of the speech characteristics and excluding the redundant information [8].

1.2 Literature survey on language recognition

Over the last few decades, many LR systems have been developed. In this section, a brief review of LR systems is presented on the basis of different time intervals.

1.2.1 Very early works

The researchers at Texas Instruments have made the sustained efforts in LR in between the year 1973 and 1980 [12–15]. In [12], the authors have explored the frequency occurrence of certain reference sounds in different languages. The automatic segmentation to the reference sounds has been done based on spectral change. The decision on the spoken language of the speech utterances has been made on the basis of time average log-likelihood of the languages. This work has been extended in [13], where sequences of several phoneme-like segments were used for classification. In [14], the human interactive approach has been used for the generation of reference sounds. The reference sounds were selected manually followed by automatic isolation of the representative occurrences of these sounds from the speech data. The work has been further extended in [15], where more emphasis was given on properly specifying the reference sounds as well as providing more flexibility in selection of reference sounds. House and Neuberg work's [16] was based on manually phonetic transcribed data for LR. In this work, the hidden Markov model (HMM) [17] was trained on broad phonetic labeled data derived from phonetic transcription. The perfect discrimination of eight languages was shown and demonstrated that excellent LR can be achieved by exploiting phonotactic information. In [18], an automatic acoustic-phonetic segmentation based approach has been used for developing the LR system. The segmented data were grouped into six classes: (i) syllable nuclei, (ii) non-vowel sonorant, (iii) vocal murmur, (iv) voiced frication, (v) voiceless frication, and (vi) silence/low energy segments.

Based on the segmented data, two statistical models were considered: segmental and syllable. The accuracy of 80% has been achieved in five languages. In [19], a polynomial classifier was designed on 100-elements linear predictive coefficients (LPC) [20] derived acoustic feature vectors consisting of 15 autocorrelation coefficients, 15 cepstral coefficients, 15 filter coefficients, 15 area functions, 15 log area ratios, 15 reflection coefficients, first-five formant frequencies, and their bandwidths. The overall accuracy of 84% was achieved in eight languages. Foil [21] has developed two LR approaches. In the first approach, a classical quadratic classifier was applied to the seven prosodic features extracted from the pitch and energy contours. The second approach exploits the frequency of occurrence of characteristic sounds using the formant frequencies and the language decisions were made on the basis of vector quantization (VQ) [22] distortion measure. With the formant cluster approach, 64% recognition rate with 11% false rejection was reported on the three language database collected from the radio with a 5 dB signal to noise ratio (SNR). Goodman *et al.* [23] extended the work described in [21] by incorporating additional and modified parameter sets, a new voicing method, and various distance metrics for classification.

In summary, the works in LR task in the early 1990's were based on the frequency occurrence of certain reference sounds, phonotactic information, LPC derived acoustic features, and prosodic features. For speech segmentation, both manual and automatic approaches were reported. The statistical approaches like log-likelihood, VQ, HMM and polynomial classifiers were also investigated. The interested reader is referred to the review work carried out by Muthusamy [24].

1.2.2 Works in last decade of 20th century

Muthusamy *et al.* [25] have built a multi-language, neural network-based segmentation and broad classification algorithm using seven broad phonetic categories. The neural networks were trained using back propagation with conjugate gradient optimization [26]. The experiments were carried out in four languages and 82.3% accuracy was noted on the test set. In [27], authors have investigated the VQ technique on the LPC derived features. Two strategies were considered for developing LR systems. In the first approach, separate VQ codebook was created for each of the languages. While, the second approach was based on single universal (common) VQ codebook for all languages, and its occurrence probability histogram. The recognition rates of 65% and 80% were reported in the first and second approaches, respectively. The results are reported with only 8 sentences of unknown speech (about 64 seconds). In [28], four statistical methods: VQ, discrete HMM, continuous density HMM and

Gaussian mixture model (GMM) [29] were compared. The 10 mel-cepstrum coefficients along with its dynamic coefficients were used as a feature vector. These mel-cepstrum coefficients were obtained by transforming 14 cepstrum coefficients (derived using 14^{th} order LPC analysis) on the mel scale. Lamel and Gauvain [30] have conducted cross-lingual experiments with phone recognition in both French and English. Three-state left-to-right continuous density HMM with GMM observation density has been used to build the phone models. In that work, BREF, a large read-speech corpus for French [31] and DARPA Wall Street Journal (WSJ) corpus [32] were used for French and English data, respectively. It was found that the recognition of French language at phone level (phone accuracy for BREF is 76.4% vs. 69% for WSJ) is easier but harder to recognize at the lexical level due to a larger number of homophones. It was also reported that the 4 kHz signal bandwidth is sufficient for the French language whereas 8 kHz is needed for the English language. In [33], phone-based acoustic likelihoods technique has been applied for LR task. The basic idea was to train the language-specific phone model sets in parallel, which were used to process the unknown speech. The experiments were done on the Oregon Graduate Institute (OGI) multilingual telephone corpus [34]. Muthusamy and Berkling [35], have compared three distinct techniques based on (a) raw acoustic features, (b) broad phonetic categories, and (c) fine phonetic categories for LR. The distinction task was limited to English and Japanese language. The acoustic representation used was 7^{th}-order perceptual linear prediction (PLP) features [36]. It was shown that the combination of fine phonetic uni-gram and bi-gram features using a neural network classifier performs better than that of Viterbi search. Zissman [37], has developed LR techniques employing continuous-observation, ergodic HMMs with tied Gaussian observation probability densities. In this approach, the MFCCs and first-order delta-cepstrum vectors have been used. In [38], the authors presented an analysis of the phoneme-based features for LR system. The features considered in this work were derived from a superset of phonemes of all three languages (German, English and Japanese). This study conclude that, using known pair-wise contrastive mono-phonemes, it is possible to limit the number of features incase of large number of languages. In [39], automatic segmentation based sub-word representation has been used to discriminate the languages within same language group. The techniques based on acoustic differences between the phonemes in each language, relative frequencies of phonemes in each language and the combination of the two sources of information have been examined. In [40,41], a comparison between four LR approaches: (i) GMM, (ii) single language phone recognition followed by language modeling (PRLM), (iii) parallel PRLM (PPRLM) which uses multiple single-language phone

recognizers, each trained in a different language, and (iv) language-dependent parallel phone recognition (PPR) have been performed. The PPRLM provides the best performance among the four approaches. Kadambe [42], has used the differences in phone/phoneme inventory and phonemotactics to identify the languages. The phoneme recognition based on tri-phonemes and tri-gram phonemotactic models have been achieved using continuous density, second order ergodic variable duration hidden Markov models. This work has been extended by incorporating the lexical access module to the phoneme recognition system [43]. Schultz *et al.* [44] studied the effects of phonetic, phonotactic, lexical and syntactic-semantic knowledge sources to large vocabulary continuous speech recognition based LR system. It was shown that incorporating lexical and linguistic knowledge leads to significant improvements in recognition performance. Navratil [45], has investigated modified bi-grams with a context-mapping matrix and language models based on binary decision trees exploiting the language information from a wide phonetic context. The proposed approaches were found to be more efficient compared to standard bi-gram in the phonotactics approach.

In summary, the majority of LR works in last decade of 20th century were based on acoustic features (e.g., PLP and MFFC) and phonotactic information. Apart from this, lexical and syntactic-semantic knowledge sources were also used to enhance the LR performance. The statistical modeling techniques like neural network, VQ, HMM, GMM, PRLM, PPRLM and PPR were also explored.

1.2.3 Works in first decade of 21st century

Wong *et al.* [46] investigated GMM-universal background model (GMM-UBM) for language modeling as a speed enhanced alternative to standard GMM system. The MFCC along with delta coefficients have been used for speech parametrization. On comparing with standard GMM based LR approach, a slight degradation was noted. In [47], the authors have presented the phonotactic-acoustic features based LR system that combines extended phonotactic and acoustic features using neural network classifier as well as phonotactic background normalization method for rejecting the unknown language. Singer [48] has described three LR systems based on Lincoln laboratory implementation of the phone recognition, GMM, and support vector machine (SVM). The LR performances have been evaluated on the 1996 and 2003 NIST test sets. In [49, 50], SVM classifier using generalized linear discriminant sequence (GLDS) kernel has been used on SDC features. For contrast purpose, the author has also developed the GMM based LR system, and found a slight degradation in the performance of the proposed system. Nagarajan *et al.* [51,52] has proposed syllable-like unit instead of the phoneme as a basic

sound unit in a framework similar to PPR, but without using annotated corpora. In this approach, similar syllable segments are clustered and syllable models are trained incrementally. The determination of the language identity from the unknown test utterances is done with the help of language dependent syllable models. In [53], the acoustic and phonetic speech information has been utilized for automatic spoken LR system. The GMM-UBM based LR system employing various acoustic features has been developed. In this work, the vocal tract length normalization (VTLN) [54, 55] which is used to reduce the speaker variation is also studied for LR task. The different aspects of phonetic speech information were also explored for the PPRLM based LR system. In [56], rhythmic units related to syllables are segmented using vowel detection algorithm. These rhythmic units are used to extract the various parameters and are modeled using GMM. In [57], a set of Legendre polynomials are used to approximate a segment of pitch contour. The coefficients of the polynomials form a feature vector. The extracted feature vectors are modeled using GMM. The LR work discussed in [58] uses SVM classifier in which Louradour sequence kernel with the background GMM was used to map the variable length sequence of acoustic feature vectors consisting of MFCC along with their with delta coefficients to the fixed dimensional space. On comparing with GLDS kernel, the Louradour sequence kernel resulted in improved LR performance. Campbell *et al.* [59] proposed GMM supervectors with SVM classifier for speaker verification task. This approach has been further extended to LR domain [60] along with adaptive relevance factor to take care of the duration mismatch. A GMM supervector is a high dimensional representation of a speech utterance which is derived by concatenating the mixture means of the language adapted GMM-UBM. In that approach, 56-dimensional SDC feature vectors (formed by concatenating 7 static MFCC plus 49 dynamic features using 7-1-3-7 delta-shift pattern) were used after voice activity detection (VAD). In [61], the vector space modeling (VSM) was proposed for automatic spoken LR task. This approach leads to a discriminative classifier backend, which was demonstrated to give superior performance over likelihood-based n-gram language modeling backend for long utterances. Ma *et al.* [62] adopted distributed output coding strategy in ensemble classifier design for vector space spoken LR. In this approach, multi-class LR problem is decomposed into many binary classification tasks, and the results of the classifiers are combined to form an output code as a hypothesized solution to the overall LR problem. Matejka *et al.* [63] describes Brno University of Technology (BUT) LR system for 2007 NIST Language recognition (LRE) evaluation. A total of 13 LR systems were fused (4 acoustic and 9 phonotactic). The 4 acoustic LR systems used were (i) GMM

system with 2048 Gaussians per language and eigen-channel adaptation, (ii) discriminatively trained GMM using maximum mutual information (GMM-MMI), (iii) GMM-MMI with channel compensated features, and (iv) SVM classifier on GMM super-vectors. The phonotactic systems were based on three phone recognizers: two ANN/HMM hybrids and one based on GMM/HMM context-dependent models. In [64], an alternative approach to PPRLM system, which aims at improving the acoustic diversification among its parallel subsystems by using multiple acoustic models has been proposed. Mary and Yegnanarayana [65] explored the vowel onset point (VOP) based prosodic features for LR task. The location of VOPs is used to obtain the syllable-like regions in continuous speech. The prosodic cues such as stress, rhythm, and intonation are characterized by energy contour, duration, and pitch contour, respectively. The quantitative parameters of energy contour, pitch contour and duration are computed from the syllable like regions and forms the prosodic features. For classification, a multilayer feedforward neural network is trained using these features. In [66], an Eigen-language space in analogy to the eigenvoice modeling approach has been proposed. The estimated language factors are applied as input features to SVM classifiers. In [67], PCA-based feature extraction for phonotactic LR has been proposed. Using PCA, the dimensionality of fixed length feature vectors from lattices obtained using phone recognizer is reduced, and are used with the SVM-based system. Stolcke *et al.* [68] has demonstrated the superior performance of language-independent phone recognizers in both PRLM and PPRLM based LR systems. The authors have also investigated the ways to use speaker adaptation transforms estimated by maximum likelihood linear regression (MLLR) as a complementary feature for language characterization. Torres-Carrasquillo *et al.* [69] have presented the description of MIT Lincoln laboratory LR system submitted to the NIST 2007 LRE. In this, a fusion of four LR systems namely a GMM-MMI spectral system, an SVM classifier on GMM supervector, an SVM language classifier using the lattice output of an English tokenizer, and a language model classifier using the lattice output of a Hungarian tokenizer. The similar approaches were used on the NIST 2009 LRE database [70]. The per system score calibration has been done using single discriminatively trained Gaussian, followed by score fusion with logistic regression. In [71], the dimensionality reduction techniques like PCA or random projection (RP) for using higher order n-grams in SVM-based phonotactic LR have been investigated. Richardson and Campbell [72] have applied a nuisance attribute projection (NAP) to token based LR system. The NAP formulation was done in such a way that it scales well to high dimensional sparse feature vectors. In [73], an attempt has been made to develop a two-level LR

system for Indian languages using acoustic features. In the first level, the system identifies the family of the spoken language, and then it is fed to the second level which aims at identifying the particular language in the corresponding family. In [74], a wide range of prosodic attributes has been considered for improving the LR performance. These attributes are modeled by the bag of n-grams approach with SVM.

In summary, the major contributions in the first decade of the 21st century in terms of features are the SDC features, VOP-based prosodic features, GMM-supervector as the utterance representation. In terms of statistical modeling techniques, GMM-UBM, and SVM played a major role in achieving the improved performances. The VTLN and NAP were also found helpful in improving the LR performance. A good introductory tutorial on the fundamentals of the theory and the state-of-the-art solutions can be found in [1].

1.2.4 Works in present decade

In this subsection, we present a brief review of recent works in the LR domain. On surveying the literature, we find that those works can be categorized into three broad themes: low-rank modeling, sparse domain modeling, and deep neural network based modeling. This review serves two purposes. First, it highlights the important contributions made in the LR domain and also provides the basis for the work carried out in this thesis.

1.2.4.1 Low-rank modeling

One of the most popular concept in the last decade is i-vector based utterance representation, a modification to joint factor analysis (JFA) [75–77]. These approaches were originally proposed for speaker verification task. In JFA, speaker and session/channel subspaces are being modeled separately. In this work, the presence of speaker-specific information was noted in the session/channel subspace. In order to address this problem, the i-vector [78] modeling was proposed where both speaker and session/channel variabilities are captured jointly by constructing a low- rank projection matrix referred to as total variability matrix. The i-vector is considered as the low dimensional representation of GMM mean-supervector derived using factor analysis. Martinez *et al.* [79] have investigated three different classifiers (linear generative model, SVM and logistic regression) on the i-vector as front-end features and these systems were noted to outperform the JFA based one. In [80], dimensionality reduction techniques like linear discriminant analysis (LDA), neighborhood component analysis (NCA),

and their combination with within class covariance normalization (WCCN) were proposed in order to compensate for the session/channel variabilities present in the i-vector. The SVM classifier was directly applied to compensated low-dimensional i-vectors, followed by score calibration. In [81], a language-dependent total variability matrix for each of the target languages was used for the extraction of multiple i-vectors and then concatenated i-vectors are used in SVM based classification. Soufifar *et al.* [82] have presented a method for extracting i-vectors by a means of subspace multinomial modeling of the n-gram counts for phonotactic LR. The SVM and logistic regression based techniques were used for scoring. In [83], speaker adaptation vectors from sub-space GMMs [84] were used as the features for LR. Although the performance was found inferior to the state-of-the-art acoustic i-vector system, it provides complementary information and useful in the fusion. In [85], prosodic features based i-vector with a generative classifier was developed for LR task. Martinez *et al.* [86] have combined prosodic and formant features based i-vectors are used to build a generative LR system. Zhang *et al.* [87], have investigated diverse acoustic features in the i-vector framework. For classification, generative Gaussian (GG) and cosine distance scoring (CDS) classifiers have been used. In [88], an LR system for mixed-language TV broadcast has been proposed. The JFA was used to separately model the language and channel variabilities, represented by the eigenvoice matrix V and the eigenchannel matrix U, respectively. The corresponding language and channel factors are extracted simultaneously by concatenating the V and U into a single matrix and by following the standard procedure of i-vector extraction. Only language factors are considered and used to obtain the scores using Gaussian and CDS classifiers. In [89], phonetic tokenizations and tandem features were investigated in the i-vector framework for speaker verification and LR task. The tokens for computing zero-order statistics were extended from the MFCC trained GMM components to phonetic phonemes, 3-grams and tandem feature trained GMM components using phoneme posterior probabilities. It was shown that for LR, the tandem feature is superior to MFCC in terms of first-order statistic calculation. Diez *et al.* [90] have investigated phone log-likelihood ratios (PLLR) features in the i-vector framework for LR task. This work was extended in [91] where PLLR features are projected into a subspace that enhances the information retrieved from the system [92]. The dimensionality of the projected features is reduced using PCA and used for the computation of shifted deltas. The fast and memory efficient approaches for i-vector extractions are described in [93, 94]. In [95], simplified and supervised i-vector modeling framework for robust and efficient speaker verification and LR was proposed.

1.2.4.2 Sparse domain modeling

The sparse representation of signals for classification is an active research area in much signal processing tasks. The sparse representation technique relies on the underlying assumption that a signal can have a compact representation as a linear combination of few columns (atoms) of an overcomplete dictionary. In an ideal condition, the selected dictionary atoms belong to the same class label as that of a query signal. Motivated by this fact, the sparse representation classification (SRC) has been originally proposed for face recognition [96] and thereafter it has gained attention in various applications such as speaker identification [97], speaker verification [98, 99] and LR [100] due to its superior performance compared to well-known classifiers like CDS and SVM. In [101] the sparse representation of GMM mean shifted supervectors over a K-singular value decomposition (K-SVD) learned dictionary for speaker verification has been investigated. In [102], a speaker verification system employing sparse representation of GMM mean shifted supervectors over discriminatively learned dictionary (supervised K-SVD) was proposed. In [103], sparse coding has been performed in order to determine the sparse vectors corresponding to SDC feature vectors. For classification, a linear SVM classifier was applied on the pooled sparse vectors. The work has been further extended with an adaptive sparse coding, which uses maximum *a-posteriori* (MAP) adaptation scheme based on online learning. A good survey on the sparse representation can be found in [104].

1.2.4.3 Deep neural network modeling

Most of the works employing deep neural network (DNN) was used for the direct classification of input features and the indirect way of computing frame-level features. The LR system discussed in [105] deals with DNN as a classifier which uses PLP features. The indirect methods of computing the features include the bottleneck features (BNF) from a DNN with a bottleneck layer or DNN posteriors features from the outer layer. In [106], an i-vector representation based on BNF for LR was proposed. The 43-dimensional features including 39-dimensional MFCC features and 4-dimensional pitch based features are spliced across 10 frames and applied to the input of DNN. A stack of restricted Boltzmann machines is trained using unsupervised generative pre-training procedure to initialize weight matrices. After pre-training, the standard error back-propagation algorithm is used to fine tune the DNN in a supervised manner. In [107], DNN replaces the GMM to compute the posterior of the frames with respect to each of the classes in the model. In the case of DNN, the classes are senones (tied triphone

states) which are achieved using a standard decision tree for automatic speech recognition (ASR). While in the GMM case, the classes correspond to the individual Gaussian in the mixture model. The DNN posteriors are then used to compute sufficient statistics for the i-vector. Richardson *et al.* [108] have investigated speaker and language recognition systems under the i-vector framework employing BNF along with DNN posteriors. The performance of the proposed system was found slightly inferior to BNF along with GMM posteriors. In that work, an LR system is developed to distinguish among 24 different languages, but the DNN was trained on the transcribed English speech data from the Switchboard-I corpus. Later in [109], multilingual transcribed data was used to train the DNN and found helpful to get an improved performance. Most of the works exploiting DNN for LR task require transcribed speech and pronunciation dictionaries which is expensive to obtain, thus limiting the applicability of this approach. Recently, the work [110] explored the DNN based LR system which uses acoustic features in the input and replaces the ASR based output labels with those learned from a Bayesian nonparametric model. This learns an appropriate set of sub-word units automatically from the speech data. The i-vector extraction using DNN is computationally expensive for both acoustic modeling and bottleneck feature extraction. Motivated by this, DNN was used as a classifier in the i-vector space [111], i.e. input to the DNN was the i-vectors, unlike the spectral features. In [112], a discriminative fine-tuning of a PLDA model in a low-dimensional PLDA latent subspace is proposed for the LR task. In [113], the performance of the feed-forward neural network based LR employing line spectral frequencies and MFCC front-end features are compared. Tang *et al.* [4] proposed the phonetic temporal network which makes use of phonetic information captured by time delay neural network (TDNN) [114], and applied them as input to long short-term memory (LSTM) [115, 116] for LR. In [117], DNN is used to map the sequences of speech features to fixed-dimensional embeddings called x-vectors, and are used for LR task. Various structures of DNN and their details can found in [118, 119]. The specifications and performances of salient language recognition system reported in the literature are summarized in Table 1.1.

1.3 Motivation of the study

- In LR domain, the idea of sparse representation was first explored in [100]. The authors derived the s-vector over an exemplar dictionary created using state-of-the-art i-vector front-end features. With the increase of a number of training examples, the size of the exemplar dictionary increases, and the computational burden of sparse coding over such dictionary be-

Table 1.1: Summary of specifications and performances of salient language recognition system reported in the literature.

Ref. No.	Year of Publication	Salient specifications of LR system	Evaluation data/ No. of languages/ Test segment durations	Reported Results Measure, Duration: Value
[79]	2011	Utt. Rep. : JFA, **i-vector** Channel Comp. : LDA and NAP Classifiers : **GG**, SVM and MLR Score Calibration : GB+MLR	NIST LRE 2009/ 23 Languages/ 3, 10, and 30 seconds	In $100 \times C_{avg}$, 3s : 14.10 10s : 4.04 30s : 1.88
[80]	2011	Utt. Rep. : i-vector Channel Comp. : **LDA**, WCCN, and NCA Classifiers : SVM Score Calibration : GB+MLR	NIST LRE 2009/ 23 Languages/ 3, 10, and 30 seconds	In % EER, 3s : 11.8 10s : 3.9 30s : 2.2
[100]	2012	Utt. Rep. : i-vector Channel Comp. : LDA and WCCN Classifiers : **SRC**, SVM and CDS Dictionary : Single Exemplar/Multiple Exemplar using Random subspace	NIST LRE 2007/ 14 Languages/ 3, 10, and 30 seconds	In % EER, 3s : 22.43 10s : 10.51 30s : 3.57
[120]	2013	Features : BNF using DNN Utt. Rep. : i-vector Channel Comp. : LDA and WCCN Classifiers : K-Nearest Neighbor	NIST LRE 2009/ 23 Languages/ 3, 10, and 30 seconds	In % EER, 3s : 9.71 10s : 3.47 30s : 1.98
[105]	2014	Features : PLP Classifiers : Fully connected feed-forward DNN Score Calibration : MLR	NIST LRE 2009/ 8 Languages/3s	In $100 \times C_{avg}$:12, In % EER, :9.58
			Google 5M/ 25 Languages + 9 dialects/ From 1 second up to 8 seconds	In $100 \times C_{avg}$: 5
[108]	2015	Features : SDC/**BNF** Utt. Rep. : i-vector based on **GMM**/DNN posteriors Classifiers : PLDA Score Calibration : Discriminative GB	NIST LRE 2009/ 23 Languages/ 3, 10, and 30 seconds	In $100 \times C_{avg}$, 3s : 15.9 10s : 6.55 30s : 2.76
[109]	2015	Features : Multilingual Stacked BNF Utt. Rep. : i-vector Channel Comp. : WCCN Classifiers : MLR Score Calibration : MLR	NIST LRE 2009/ 23 Languages/ 3, 10, and 30 seconds *Training data in 11 languages	In $100 \times C_{avg}$, 3s : 6.75 10s : 2.34 30s : 1.43
[103]	2016	Features : SDC Method: Adaptive sparse coding of SDC features Classifiers : SVM	NIST LRE 2015/ 9 Languages (2 clusters: Arabic (5) and Chinese (4))/ upto 30s	In $100 \times C_{avg}$, Arabic: 18.74 Chinese: 16.34
[117]	2018	Features : x-vectors trained on MFCC/BNF/Multilingual BNF, Utt. Rep. : i-vector Channel Comp. : WCCN Classifiers : Discriminative Gaussian	NIST LRE 2017/ 14 Languages (5 clusters)/ 3, 10, and 30 seconds	In $100 \times C_{Primary}$, xvect-mbnf : 13.0 xvect-bnf : 14.8 xvect-mfcc : 20.9

comes quite prohibitive. Thus, extracting the sparse representation of i-vector over simple learned/discriminatively learned dictionary may be the preferable choice. It helps to reduce the computational burden and expected to provide enhanced performance.

- The major drawback of the i-vector approach lies in high computational complexity in its extraction and the storage requirements. In order to be effective, the i-vector approach requires appropriate session/channel compensation. That indicates the application of compensation techniques in low dimensional space is very much required in order to get the good performance. This may be done by partitioning the GMM-mean supervector into subvectors (slices) followed by the session/channel compensation. This approach does not require any learning for creating the subspaces and has very low storage requirements.
- Unlike the speaker verification task, the number of languages handled by the LR system happens to be much smaller. As a result of this, in the LR task, the language-specific information in JFA can be captured without restricting the language space unlike done for the speaker space in speaker recognition tasks. Attributed to a small number of language factors involved, the complexity of JFA based LR system is expected to be quite low compared to the i-vector based one. In this thesis, we have used JFA modeling in two ways: more efficient session/channel compensation of GMM-mean supervector and deriving a relatively much lower dimensional feature than the i-vector feature in sparse representation based LR system
- As mentioned, the sparse coding over reduced size learned/discriminately learned dictionary may be the preferable choice to reduce the computational burden of the *s-vector* extraction over an exemplar dictionary based LR approach. The question arises can we do something to further reduce the latency. The ensemble of learned-exemplar (class-wise concatenation of the learned dictionary) may be a suitable option in order to reduce the latency as well as to achieve the diversity gain.
- Recently, DNNs have gained lots of attention in the speech technologies on account of yielding impressive gain in the respective performance measures. In general, DNN can be trained as a classifier for the intended task or can be used as a means of extracting features to be used by another classifier. The DNN-bottleneck features based i-vector has been reported to provide the state-of-the-art performances in LR task [108]. This motivated us to explore the sparse representation derived using the DNN-bottleneck based i-vector features.

1.4 Contributions of the thesis

The thesis mainly explores the use of sparse representation for LR task. Apart from invoking the SRC, the LR is also performed by applying the CDS classifier to the s-vectors. The salient contributions made in this thesis are summarized as follows:

- Exploration of learned dictionary based sparse representation in LR task. In this context, both simple and discriminative learned dictionaries have been created employing different kinds of regularization. Two different representations of speech utterances namely GMM-mean supervector and i-vector are used in this work.
- In order to address the computational complexity in the i-vector extraction along with run-time memory requirement, a low complexity LR approach exploiting ensemble of random subspaces of GMM-mean supervector is proposed. Furthermore, the JFA-based session/channel compensation is applied to the GMM mean supervectors to further improve the LR performances.
- To further enhance the recognition performances by exploiting the diversity, the sparse coding of i-vector/JFA latent vector over ensemble discriminative learned dictionaries for LR is also proposed.
- For contrast purpose, recently proposed bottleneck features based i-vector, TDNN, LSTM, and PTN based LR systems are developed. Motivated by those works, the use of the bottleneck features based i-vectors is also made in the sparse coding domain. The resulting LR systems noted to outperform the existing contrast systems.

1.5 Organization of the thesis

The rest of the thesis is organized as follows. Chapter 2 describes different modules of a typical language recognition system and reviews the most commonly used techniques for realizing each of the modules. The state-of-the-art i-vector-based language recognition is also described in detail, which is used as the primary contrast method in this thesis. The details of the speech corpus and the performance measure are also presented in this chapter.

Chapter 3 presents an in-depth exploration of the sparse representation approaches for LR task. It begins by describing the existing LR approach employing the i-vector representation over an exemplar dictionary using Lasso regularization based sparse coding. The work is further extended to employ OMP/ENet regularization in sparse coding and the use of GMM-mean supervector as utter-

ance representation. It is followed by an exploration of simple learned and discriminatively learned dictionaries for LR task. For session/channel compensation, the i-vectors are applied with well-known techniques like LDA, WCCN, and LDA followed by WCCN, while the GMM-mean supervectors are processed with JFA prior to dictionary learning. The i-vector based system with different classifiers is also implemented for the contrast purposes.

Chapter 4 deals with the implementation of low complexity LR system exploiting ensemble of random subspaces of GMM-mean supervector which includes LDA/WCCN based session/channel compensation in the subspace. This work is intended for addressing the high computational complexity and memory requirement associated with the i-vector-based LR systems. The proposed ensemble-based LR approach does not require any pre-learned projection matrix for obtaining the subspaces and has very low storage requirements. In typical LR applications, the number of languages required to be identified is limited to 10-30. As a result of that, the complexity of the JFA based system turns out to be lower than that of the i-vector based one. Exploiting this fact, the combination of the proposed ensemble approach with JFA is also explored.

In Chapter 5, an ensemble of the learned-exemplar dictionary (concatenation of class-based learned dictionaries) based LR approach is proposed which can handle large sized training data efficiently along with benefitting from the diversity gain. In this approach, each of the test vectors is sparse coded over multiple learned-exemplar dictionaries, unlike single learned-exemplar dictionary. The language-specific coefficients of each of the s-vectors are averaged and used as the score vectors. These score vector obtained from each of the dictionaries in the ensemble is first calibrated with Gaussian back-end and then all calibrated scores are fused together using the multiple-class logistic regression for the final decision.

In Chapter 6, we explore the sparse coding of DNN based i-vectors for LR task. For i-vector representation, DNN based bottleneck features are used, unlike the conventional MFCC features. For contrast purpose, the existing DNN based LR approaches are also developed.

Finally, Chapter 7 summarizes the work presented in this thesis and presents a few directions for future work. The details of the key algorithms used for sparse coding and dictionary learning are presented in Appendices A and B, respectively. The details of the tuning parameters of the dictionary learning approaches are presented in Appendix C.

2

Language Recognition System

Contents

2.1 Introduction

The primary focus of this chapter is to review the techniques involved in the task of automatic spoken language recognition. Like developing any other classifier, it involves two stages namely the training and the testing. In the training stage, speech utterances from the target languages are used for creating the corresponding language models while the testing stage depends upon the task, i.e., whether we need to perform language detection or language identification. The testing stage of language detection uses a segment of speech and a target language claim and performs the pattern matching with the claimed target language model towards accepting/rejecting the claim. However, the testing stage of language identification determines the language of the unknown speech segment by computing maximum similarity score with respect to target languages being modeled. The block diagram of the state-of-the-art language detection system exploiting the acoustic features and the i-vector paradigm is, shown in Figure 2.1. The different blocks involved in the training and testing stages are shown separately. The following sections describe the functionality of each of the blocks used to build the language detection system.

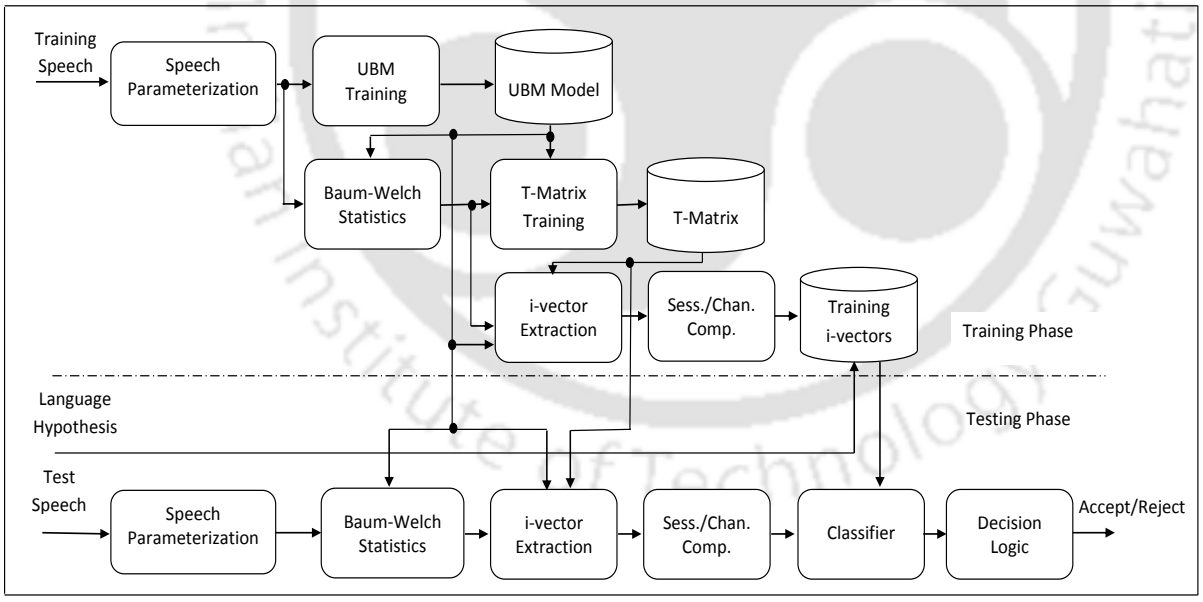

Figure 2.1: Block diagram of the i-vector based language detection system

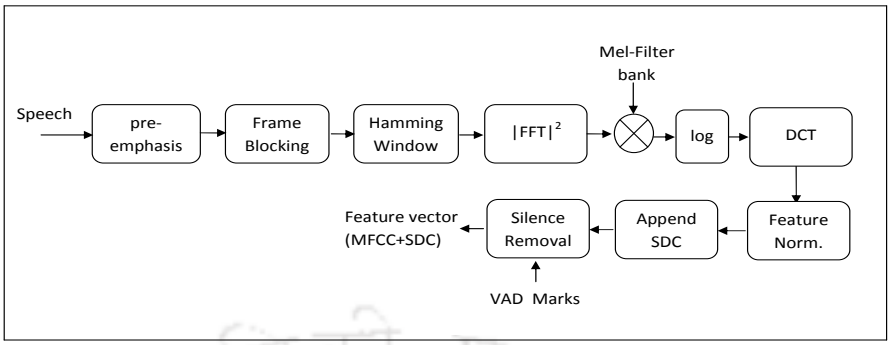

Figure 2.2: Block diagram of MFCC plus SDC feature extraction

2.2 Speech parametrization

The speech parameterization is the process of extracting the relevant information from speech signal. Speech is a non-stationary signal and therefore needs to be processed/analyzed over short duration speech frames. In general, a Hamming window of typically 20-30 ms is applied on the speech signal with a frameshift of 10 ms. These speech frames are assumed to be stationary and used to obtain the appropriate acoustic feature vectors from each of the frames. Prior to windowing, the pre-emphasis filter is usually applied to an utterance in order to enhance the high-frequency components. To remove the silence or non-speech regions from the utterances usually, VAD is also performed. The signal energy based VAD forms the simplest method for speech and silence regions and works satisfactorily for telephone (narrowband) speech [121]. In that method, the speech/non-speech frame is decided by comparing the frame energy to a threshold derived from the average energy of the utterance. When the energy of a given frame is more than the threshold value, that particular frame is marked as the speech frame or otherwise silence frame. The indices of speech/non-speech frames (VAD marks) are stored and used to remove the frames after the feature extraction. The majority of the acoustic feature extraction techniques used for LR task are based on short-term spectral features. MFCC [9] and PLP [36] are two dominant spectral features used for speech-based classification tasks, with MFCC being the most popular for LR task. Figure 2.2 shows the block diagram of the MFCC feature extraction procedure and it comprises of following salient stages:

(i) apply the fast Fourier transform (FFT) to the windowed speech and compute the periodogram estimate of the power spectrum, (ii) multiply Mel-scaled filterbank with the power spectrum, and

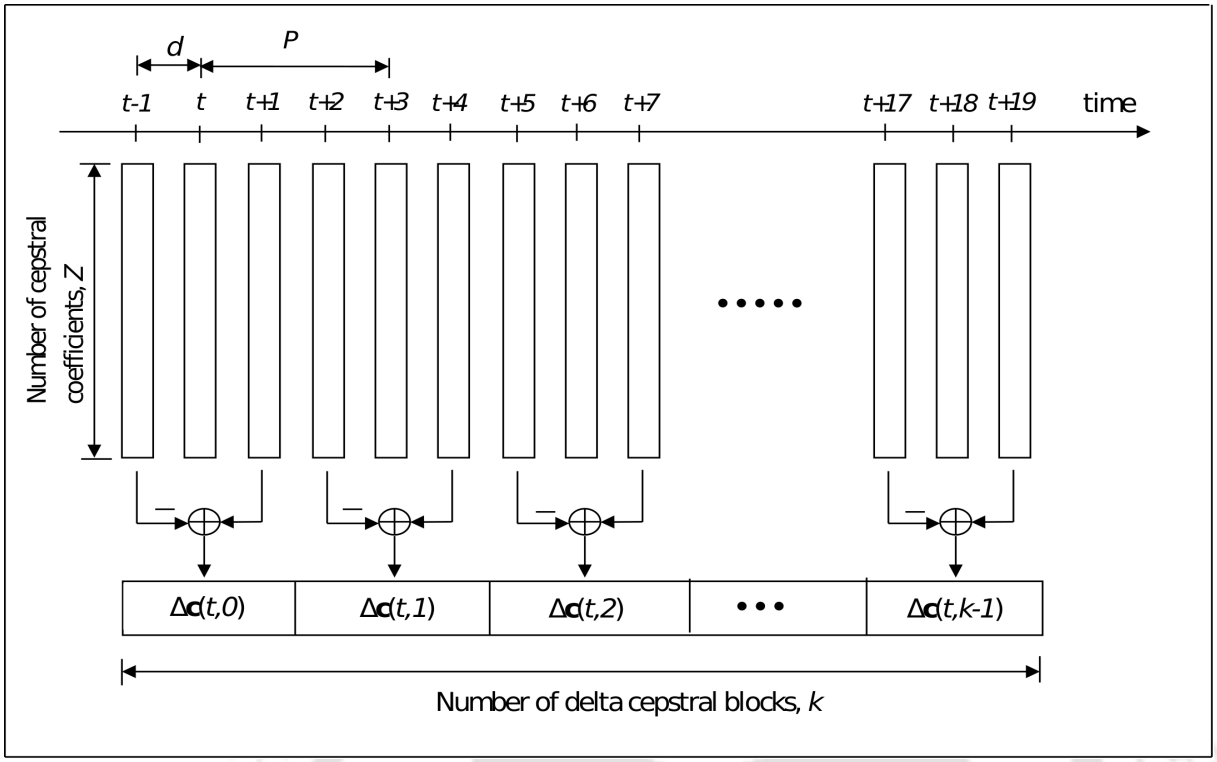

Figure 2.3: The illustration of extraction of the t-th frame of SDC feature for the choice of parameter set, Z-d-P-k = 7-1-3-7 [1]

then compute the energy output of each filter in the filterbank by adding up the weighted spectral coefficients, and (iii) finally obtain the cepstral domain representation by applying logarithmic compression to the filterbank energies, followed up by the discrete cosine transform (DCT). The cepstral coefficients only contain information about the particular frame, and hence usually referred to as the static features. Additional information about the temporal dynamics of the signal is captured by computing first and second derivatives of cepstral coefficients [122–124], and are referred to as delta (Δc) and delta-delta ($\Delta\Delta c$) coefficients, respectively. The delta coefficients provide information about the speech rate, while delta-delta coefficients give the information similar to the acceleration of speech. The traditional delta cepstra are computed over a few neighboring frames, thus the captured temporal context is limited. In order to capture the temporal dynamics to a greater extent, SDC [125] feature vectors are also proposed. The SDC feature vectors are created by stacking delta cepstra compounded across multiple speech frames as illustrated in Figure 2.3. The four parameters Z-d-P-k completely specify the SDC features. Z is the number of cepstral coefficients computed at each frame, d represents the time advance and delay for delta computation, P is the time shift between consecutive blocks, and

k is the number of delta-cepstral blocks whose coefficients are stacked to form a final feature vector. Using *Z-d-P-k* SDC feature extraction scheme, the Z static features are calculated as

$$\boldsymbol{c}(t) = [c_0(t), c_1(t), c_2(t), \ \ldots, c_{Z-1}(t)]' \tag{2.1}$$

and the ith block of delta-cepstral features is computed as

$$\boldsymbol{\Delta c}(t,i) = \boldsymbol{c}(t + iP + d) - \boldsymbol{c}(t + iP - d) \tag{2.2}$$

The Z static and kZ dynamic feature vectors at time t are computed and attributed to represent the static and dynamic characteristics of the vocal tract. The concatenation of SDC features with the static features is reported to yield the improved LR performance.

The cepstral mean normalization (CMN) [126] and the cepstral mean and variance normalization (CMVN) [127, 128] are relatively easy and effective feature normalization techniques. In CMN, the mean value of the cepstral coefficients computed over the whole utterance is subtracted from each of the cepstral coefficients. This helps to remove the time-invariant distortions (convolution) introduced by the transmission channel and the recording device. However, in CMVN, the cepstral coefficients are linearly transformed to have zero mean and unit variance. In real time application, normalizing a feature vector over the entire utterance is not a feasible solution [129], as it causes an unnecessarily long processing delay. To reduce the processing delay, the segmental CMVN [128] technique is also proposed. Here, cepstral features are normalized over a sliding window of 3-5 seconds duration. The feature vector to be normalized is located at the center of the sliding window. The VTLN is another widely used normalization approach which helps to reduce the inter-speaker variability that arises due to physiological differences in the vocal-tracts across the speakers. The speaker normalization is done by warping the frequency-axis of the spectra of speakers by appropriate warp factor [130]. The maximum likelihood search with respect to a GMM model is done to find the appropriate warp factor. In addition, relative spectral (RASTA) [131] filtering performed on temporal trajectories of critical band energies is also useful to enhance the speech features. It addresses the problem of the slowly time-varying linear channel. The RASTA filter is a band pass filter used to suppress the spectral components that change more slowly or quickly than the typical range of change of speech.

Apart from the acoustic-phonetic features, various levels of perceptual cues such as prosodic [65, 74, 85, 86] and phonotactic [82, 132] approaches are also been explored for the LR task. In [85], it

was reported that the i-vector representation over prosodic feature based LR system gives a degraded performance in comparison to the acoustic and phonotactic system. However, these features are used as meaningful complementary sources of information and fused to obtain the improved performance. In comparison with phonotactics level features, comparable performance was noted with acoustic-phonetic SDC features [82, 133]. The phonotactic system requires a good phoneme recognizer which needs to be trained on large transcribed speech corpus [134]. Only a few works have been reported utilizing the lexical information. The reason behind this may be due to high computational complexity and requirement of language-specific transcribed speech data [44,134,135]. Nowadays, many state-of-the-art LR systems still include or rely on acoustic modeling [70].

2.3 Statistical modeling techniques

Statistical modeling is a critical issue in scientific data analysis. Its purpose is to construct a model that approximates the true structure as accurately as possible through the use of available data. A statistical model is a probability distribution, and are used to represent stochastic structures, predict future behavior, and extract useful information from the available data [136]. The language detection task also relies on the statistical models (language models) as it depends on similarity calculation between test utterance and the reference model of the language. Thus, construction of the good model is critical for the success of LR system. The extracted SDC feature vectors of the training speech data in target languages are used to train the corresponding language models. In literature, various statistical modeling approaches for LR have been tried including VQ [22], HMM [17], artificial neural network (ANN) [137], GMM [29], GMM-UBM [138], SVM [139, 140], GMM mean supervector representation [59], i-vector representation [78], PRLM [41], PPRLM [41], and PPR-VSM [61]. A few of the most commonly used statistical modeling techniques are described in the following subsections.

2.3.1 Gaussian mixture model

A GMM is a parametric model of probability density function of the feature vectors represented as a weighted sum of C component Gaussian densities [29]. A GMM denoted by λ can be expressed as,

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{C} \eta_i b(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{2.3}$$

where $\mathbf{x}$ is a D-dimensional feature vector and $b(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), i = 1, 2, \ldots, C$ are the component Gaussian densities. The parameters $\eta_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ represent the mixture weight, mean vector and covariance matrix

of the ith mixture component. These parameters are collectively represented by the notation, $\lambda = \{\eta_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{C}$. Further, each component density is a D-variate Gaussian function of the form,

$$b(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{'}\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]. \tag{2.4}$$

The mixture weight η_i must satisfy the condition

$$\sum_{i=1}^{C} \eta_i = 1 \tag{2.5}$$

The covariance matrices $\boldsymbol{\Sigma}_i$ can be full rank or constrained to be diagonal. It depends on the number of Gaussian components and amount of data availability for estimating the GMM parameters. Usually, the diagonal covariance matrices are preferred on account of the computational constraint [29]. For using GMM as a language model, the model parameters $\{\eta_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{C}$ are required to be estimated using the corresponding language training data. These parameters are estimated following the maximum likelihood (ML) criteria with the help of expectation maximization (EM) algorithm [141].

2.3.2 GMM-universal background model

As the number of languages increases, building a language-specific GMM model having a large number of mixture components is not a good choice. This requires too much time to build all the language specific models and needs large memory storage space for saving all the models. To overcome this problem, GMM-UBM [138] is used. In the GMM-UBM system, training language model is derived by adapting training utterances with UBM and a form of Bayesian adaptation. UBM is language independent model and derived from large multilingual training dataset using EM algorithm. Following the MAP approach, a language-dependent GMM is obtained by adapting the parameters of the UBM from language-dependent training data. In general, only mean parameters are adapted as it is found to be effective.

Suppose we have access to a language independent GMM-UBM which is represented by weighted sum of C Gaussian mixture densities and denoted as $\lambda_{\text{UBM}} = \{\eta_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{C}$ and the language-dependent data vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$, the MAP adaption process of its mean parameters is performed as follows. The posterior probability $p(i|\mathbf{x}_k, \lambda_{\text{UBM}})$ of the ith mixture component for a data vector $\mathbf{x}_k$ with respect to λ_{UBM} is computed as,

$$p(i|\mathbf{x}_k, \lambda_{\text{UBM}}) = \frac{\eta_i b(\mathbf{x}_k|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{j=1}^{C} \eta_j b(\mathbf{x}_k|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2.6}$$

Once posterior probability $p(i|\mathbf{x}_k, \lambda_{\text{UBM}})$ is determined, the sufficient statistics for estimating the mean are computed as

$$0^{th} \text{ order statistics} : N_i = \sum_{k=1}^{K} p(i|\mathbf{x}_k, \lambda_{\text{UBM}}) \tag{2.7}$$

$$1^{st} \text{ order statistics} : \acute{\boldsymbol{F}}_i = \frac{1}{N_i} \sum_{k=1}^{K} p(i|\mathbf{x}_k, \lambda_{\text{UBM}}) \mathbf{x}_k \tag{2.8}$$

This is same as the "Expectation" step in the EM algorithm. Finally, the MAP adapted mean vector $\boldsymbol{\mu}_i^a$ corresponding to the ith mixture is computed as

$$\boldsymbol{\mu}_i^a = \alpha_i \acute{\boldsymbol{F}}_i + (1 - \alpha_i)\boldsymbol{\mu}_i \tag{2.9}$$

where α_i is the adaptation coefficient controlling the balance between old mean and adapted mean.

2.3.3 GMM mean supervector representation

GMM mean supervector is a high dimensional vector derived from adapted GMM-UBM [142] and forms the choicest representation in speaker and language recognition domains [60, 143]. The mean vectors corresponding to a language adapted GMM-UBM are concatenated to form a GMM mean supervector $\grave{\boldsymbol{s}}$. Let us consider $\boldsymbol{\mu}^a \in \mathcal{R}^{C \times N}$ be the GMM adapted mean matrix for an utterance u. Each row represents a mean vector corresponding to a particular Gaussians. The concatenation of mean vectors form a supervector and is given by

$$\grave{\mathbf{s}} = [\boldsymbol{\mu}_{1,:}^a \mid \boldsymbol{\mu}_{2,:}^a \mid \ \ldots \ \boldsymbol{\mu}_{C,:}^a]' \tag{2.10}$$

where $\boldsymbol{\mu}_{i,:}^a$ represents the mean vector of the i-th Gaussian in GMM. In order to reduce the bias due to UBM, the centered GMM mean supervectors are generally used and are derived as

$$\tilde{\boldsymbol{s}} = \grave{\boldsymbol{s}} - \boldsymbol{\mu} \tag{2.11}$$

where $\boldsymbol{\mu}$ is the GMM-UBM mean supervector.

2.3.4 i-vector representation

The concept of total variability space or i-vector approach was first applied to speaker verification [78]. Later, the same idea was applied for language identification in [80], [79]. The i-vectors are the reduced dimension representation of GMM mean supervectors derived using factor analysis and involve a low-rank projection matrix referred to as the *total variability matrix*. In context of language

identification, the total variability matrix models both the language and the channel variability. Given a speech utterance u, the language and session/channel-dependent GMM mean supervector $\grave{\boldsymbol{s}}$ is modeled as

$$\grave{\boldsymbol{s}} = \boldsymbol{\mu} + \boldsymbol{T}\boldsymbol{w} \tag{2.12}$$

where $\boldsymbol{\mu}$ is the language independent UBM mean supervector, $\boldsymbol{T}$ is the total variability matrix, and $\boldsymbol{w}$ is known as the identity vector or i-vector and assumed to have normal distribution $N(\boldsymbol{0}, \boldsymbol{I})$. The training procedure of total variability matrix $\boldsymbol{T}$ is exactly the same as learning the eigenvoice matrix [144] except the way speech utterances are treated. For an utterance u, the i-vector $\boldsymbol{w}$ is computed as,

$$\boldsymbol{w} = (\boldsymbol{I} + \boldsymbol{T}^{'}\boldsymbol{\Sigma}^{-1}\boldsymbol{N}(u)\boldsymbol{T})^{-1}\boldsymbol{T}^{'}\boldsymbol{\Sigma}^{-1}\tilde{\boldsymbol{F}}(u) \tag{2.13}$$

where $\boldsymbol{\Sigma}$ and $\boldsymbol{N}(u)$ are diagonal matrices whose diagonal blocks are $\boldsymbol{\Sigma}_i$ and $N_i\boldsymbol{I}$, respectively. $\tilde{\boldsymbol{F}}(u)$ is a super vector of size $CD \times 1$ obtained by concatenating all centralized first -order Baum-Welch statistics $\tilde{\boldsymbol{F}}_i$ for a given utterance u. The centralized first-order Baum-Welch statistics for ith UBM mixture component is given by

$$\tilde{\boldsymbol{F}}_i = \sum_{k=1}^{K} p(i|\mathbf{x}_k, \lambda_{UBM})(\mathbf{x}_k - \boldsymbol{\mu}_i) \tag{2.14}$$

where $i = 1, 2, \ldots, C$ is the Gaussian index and $p(i|\mathbf{x}_k, \lambda_{UBM})$ is the posterior probability of the ith Gaussian mixture component for a data vector $\mathbf{x}_k$ with respect to λ_{UBM}.

2.4 Session/Channel compensation

The recordings belonging to the same language obtained through different channels (different microphones, acoustic environments, and transmission channels) known as inter-session channel variability. The other variability may be due to the different environmental conditions while recording the training and testing data from different languages. These issues make the LR problem more challenging. Here, we present the brief idea of three channel compensation techniques which are popularly used in speaker verification and language recognition to minimize the session/channel effects. The JFA is used to remove the session/channel effect in high dimensional GMM supervector space while LDA/WCCN are commonly used in i-vector space.

2.4.1 Joint factor analysis

In JFA [75], the GMM mean shifted supervector $\tilde{\boldsymbol{s}}$ for a language is represented as the sum of three factors as,

$$\tilde{\boldsymbol{s}} = \boldsymbol{U}\boldsymbol{u} + \boldsymbol{V}\boldsymbol{v} + \boldsymbol{D}\boldsymbol{d} \tag{2.15}$$

where $\boldsymbol{U}$ is the session/channel subspace (eigenchannel matrix), $\boldsymbol{V}$ and $\boldsymbol{D}$ define a language subspace (eigenvoice matrix and diagonal residual matrix, respectively). The vectors $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{d}$ are session and language dependent factors in their respective subspaces, and each is assumed to be a random variable with normal distribution $N(\boldsymbol{0}, \boldsymbol{I})$. The session/channel compensated GMM mean shifted supervector is given by $\check{\boldsymbol{s}} = \boldsymbol{V}\boldsymbol{v} + \boldsymbol{D}\boldsymbol{d}$. In our implementation, we have used $\boldsymbol{V}\boldsymbol{v}$ factor only, ignoring the residual factor. The factor $\boldsymbol{V}\boldsymbol{v}$ is referred to as the JFA compensated supervector or in short the JFA-supervector. The MAP point estimate $\boldsymbol{v}$ corresponding to the eigenvoice matrix is referred to as the JFA latent vector in this work.

2.4.2 Linear discriminant analysis

LDA is used to perform session/channel compensation of i-vector representations by maximizing the separability between languages. In this, the i-vectors are projected down to a set of new orthogonal axes where the discrimination between different classes (languages) is maximum. The projection matrix is composed of eigenvectors corresponding to top $(L-1)$ eigenvalues, where L is the total number of languages used. The eigenvalues can be obtained by solving eigenanalysis equation

$$(\breve{\boldsymbol{W}}^{-1}\boldsymbol{B})\alpha = \boldsymbol{\lambda}\boldsymbol{\alpha} \tag{2.16}$$

where $\breve{\boldsymbol{W}}$ is the within-class covariance matrix, $\boldsymbol{B}$ is the between-class covariance matrix, $\boldsymbol{\alpha}$ is an arbitrary vector, and $\boldsymbol{\lambda}$ is the diagonal matrix of eigen values [78]. These matrices are learned on large set of training i-vectors. The matrix $\breve{\boldsymbol{W}}$ is defined as

$$\breve{\boldsymbol{W}} = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{n_l}\sum_{i=1}^{n_l}(\boldsymbol{\omega}_i^l - \overline{\boldsymbol{\omega}}_l)(\boldsymbol{\omega}_i^l - \overline{\boldsymbol{\omega}}_l)' \tag{2.17}$$

where $\overline{\boldsymbol{\omega}}_l = \frac{1}{n_l}\sum_{i=1}^{n_l}\boldsymbol{\omega}_i^l$ is the mean of i-vectors for each language, L is the total number of language and n_l is the number of i-vectors per language. The matrix $\boldsymbol{B}$ is computed as

$$\boldsymbol{B} = \frac{1}{L}\sum_{l=1}^{L}(\boldsymbol{\omega}_l - \overline{\boldsymbol{\omega}})(\boldsymbol{\omega}_l - \overline{\boldsymbol{\omega}})' \tag{2.18}$$

In factor analysis, the i-vectors have standard normal distribution $w \sim \mathcal{N}(0, I)$, hence the language population mean vector $\overline{\boldsymbol{\omega}}$ is equal to the null vector.

2.4.3 Within-class covariance normalization

WCCN is another linear transformation method widely used to mitigate the session/channel effects from the Gaussian mean supervector and the i-vector [78, 145]. In WCCN method, the data vectors are transformed using a matrix which minimizes the upper bounds on the classification error metric and hence minimizes the classification error. The transformation matrix $\boldsymbol{B}$ is obtained by Cholesky decomposition of the inverse of the within-class covariance matrix $\breve{\boldsymbol{W}}$ as,

$$\breve{\boldsymbol{W}}^{-1} = \boldsymbol{B}\boldsymbol{B}^{'} \tag{2.19}$$

2.5 Classifiers

In this section, five popularly used classifiers in speaker/language recognition domain are briefly discussed for the contrast purpose. The strengths and weakness of each classifier are summarized in Table 2.1. The algorithms are explained in terms of the i-vector based representation of the speech utterances.

2.5.1 Cosine distance scoring

The CDS provides the similarity between two i-vectors. Given the training and test i-vectors ($\boldsymbol{w}_{trn}$ and $\boldsymbol{w}_{tst}$), the score is computed as,

$$\text{Score}(\boldsymbol{w}_{trn}, \boldsymbol{w}_{tst}) = \frac{(\boldsymbol{w}_{trn})^{'}\boldsymbol{w}_{tst}}{\|\boldsymbol{w_{trn}}\|_2 \, \|\boldsymbol{w_{tst}}\|_2} \tag{2.20}$$

Using Equation 2.20, the scores against all training i-vectors are are computed for a particular test i-vector. Finally, the class-wise mean of the scores are computed for the final decision.

2.5.2 Support vector machine

The SVM [139] is a supervised binary classifier. In the training phase, a set of training instance-class label pairs $(\boldsymbol{w}_i, c_i),\ i = 1, 2, \ldots N,\ \boldsymbol{w}_i \in \mathbb{R}^m,\ c_i \in (-1, +1)$ is used to determine the best linear hyperplane H, which maximizes the margin between the two classes. The notations $\boldsymbol{w}_i$ and c_i represent ith i-vector and corresponding class label, respectively. The classification function f associated with

the optimal hyperplane H is given by,

$$f(\boldsymbol{w}) = \sum_{i=1}^{N} \alpha_i c_i k(\boldsymbol{w}, \boldsymbol{w}_i) + b \tag{2.21}$$

where α_i and b are the Lagrange multiplier and bias, the SVM parameters obtained from training step. Considering the linear kernel function $k(\boldsymbol{w}, \boldsymbol{w}_i) = \boldsymbol{w}_i^{'}\boldsymbol{w}$, Equation 2.21 can be re-written as

$$f(\boldsymbol{w}) = \left(\sum_{i=1}^{N} \alpha_i c_i \boldsymbol{w}_i\right)^{'} \boldsymbol{w} + b = \boldsymbol{a}^{'}\boldsymbol{w} + b \tag{2.22}$$

where $\boldsymbol{a} = \left(\sum_{i=1}^{N} \alpha_i c_i \boldsymbol{w}_i\right)$ is the weight vector. For the given test i-vector $\boldsymbol{w}_{tst}$, the classification is performed by noting the sign of the function $f(\boldsymbol{w}_{tst})$. The *one-vs-all* strategy is used to obtain the scores for all languages. The other three popular kernels used in SVM are as follows:

- radial basis function (RBF): $k(\boldsymbol{w}, \boldsymbol{w}_i) = \exp\left(-\gamma \|\boldsymbol{w} - \boldsymbol{w}_i\|^2\right),\ \gamma > 0$
- polynomial: $k(\boldsymbol{w}, \boldsymbol{w}_i) = (\gamma \boldsymbol{w}_i^{'}\boldsymbol{w} + r)^d,\ \gamma > 0$
- sigmoid: $k(\boldsymbol{w}, \boldsymbol{w}_i) = \tanh(\gamma \boldsymbol{w}_i^{'}\boldsymbol{w} + r),\ \gamma > 0$

where γ, r and d are the parameters of the kernel.

2.5.3 Generative Gaussian model

In linear generative Gaussian (GG) model [79], the class specific i-vectors are used to train a class dependent multivariate normal distribution $N(\boldsymbol{\mu}_l, \boldsymbol{\Sigma})$, where full covariance matrix $\boldsymbol{\Sigma}$ is shared across all classes. Given the test i-vector $\boldsymbol{w}$, the log-likelihood score for each class is computed as

$$\ln p(\boldsymbol{w}|l) \simeq \boldsymbol{w}^{'}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_l - \frac{1}{2}\boldsymbol{\mu}_l^{'}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_l \tag{2.23}$$

where $\boldsymbol{\mu}_l$ is the mean vector and $\boldsymbol{\Sigma}$ is the common covariance matrix. Equation 2.23 is linear in $\boldsymbol{w}$, and hence leads to a linear classifier.

2.5.4 Logistic regression

Given a set of training instance-class label pairs $(\boldsymbol{w}_i, c_i),\ i = 1, 2, \ldots N$ where $\boldsymbol{w}_i \in \mathbb{R}^m$ and $c_i \in (0, 1)$ are the ith i-vector and its class label, respectively. The probability distribution of the class label c given an i-vector $\boldsymbol{w}$ can be modeled using logistic regression [146], and given by

$$p(c = 1|\boldsymbol{w}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^{'}\boldsymbol{w}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^{'}\boldsymbol{w})} \tag{2.24}$$

where $\boldsymbol{\theta} \in \mathbb{R}^m$ are the parameters of logistic regression model and $\sigma(.)$ is the sigmoid function. The regularized logistic regression model can be described as

$$\arg\max_{\theta} \sum_{i=1}^{N} \log p(c_i|\boldsymbol{w}_i;\boldsymbol{\theta}) - \lambda R(\boldsymbol{\theta}) \tag{2.25}$$

where $R(\boldsymbol{\theta})$ is the regularization term. The parameter $\boldsymbol{\theta}$ can be computed using Equation 2.25. The value of $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$ and $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$ corresponds to l_1 and l_2 regularized logistic regression. Once $\boldsymbol{\theta}$ is computed, find the decision boundary for the classification. The decision boundary is defined as the line where

$$p(c = 1|\boldsymbol{w};\theta) = 0.5 \Longrightarrow \theta^{'}\boldsymbol{w} = 0 \tag{2.26}$$

For multi-class problem, *one-vs-all* strategy is used to obtain the scores for all languages.

2.5.5 Gaussian-PLDA

Assume $\boldsymbol{w}_k$ represents the kth i-vector of a language data, where $k = 1, 2, \ldots, N$. The Gaussian-probabilistic linear discriminant analysis (G-PLDA) [147], [148] model assumes that each i-vector $\boldsymbol{w}_k$ can be decomposed into language component $\boldsymbol{l}$ and channel component $\boldsymbol{c}$ and is expressed as

$$\boldsymbol{w}_k = \boldsymbol{l} + \boldsymbol{c} = (\boldsymbol{m} + \boldsymbol{\Phi}\boldsymbol{\phi}) + (\boldsymbol{\Psi}\boldsymbol{\psi}_k + \boldsymbol{\eta_k}) \tag{2.27}$$

The language and channel components describe the between-language and within-language variability, respectively. The former doesn't depend on the particular utterance while the later is utterance dependent. The columns of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ provides a basis for the language subspace (eigenvoice) and channel subspace (eigenchannel), respectively. The $\boldsymbol{\phi}$ and $\boldsymbol{\psi}_k$ are the corresponding latent identity vectors having standard normal distributions. The residual term $\boldsymbol{\eta_k}$ is assumed to be Gaussian with zero mean and diagonal covariance $\boldsymbol{\Sigma}$. The global offset is represented by $\boldsymbol{m}$. In this work, we have used the modified G-PLDA [149] model due to low dimensional i-vector. Here, the eigenchannels have been removed and $\boldsymbol{\Sigma}$ is considered as full covariance matrix. The modified G-PLDA model is given by

$$\boldsymbol{w}_k = \boldsymbol{m} + \boldsymbol{\Phi}\boldsymbol{\phi} + \boldsymbol{\eta_k} \tag{2.28}$$

The EM algorithm [141] is applied on the large development data for ML point estimates of the model parameters $\{\boldsymbol{m}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}\}$.

Table 2.1: Comparison among CDS, SVM, MLR, GG, and G-PLDA classifiers

S.No.	Classifiers	Pros	Cons
1	CDS	Simple and easy to implement.	1. Less optimal under spaces of lower dimension. 2. Don't work well for correlated classes.
2	SVM	1. SVM's can model non-linear decision boundaries and there are many kernels to choose from. 2. Fairly robust against over-fitting, especially in high-dimensional space.	1. Memory intensive,trickier to tune due to the importance of picking the right kernel. 2. Don't scale well to larger datasets.
3	MLR	1.Algorithm can be regularized to avoid over-fitting. 2.Logistic models can be updated easily with new data using stochastic gradient descent.	1. Unstable with well separated classes 2. Unstable with few examples. 3. Tends to under-perform when there are multiple or non-linear decision boundaries.
4	GG	1. A good algorithm to use for the classification of static postures and non-temporal pattern recognition.	1. For computational reasons, it can fail to work if the dimensionality of the problem is too high. 2. User must set the number of mixture models that the algorithm will try and fit to the training dataset.
5	G-PLDA	1. A probabilistic version of LDA and so inherits LDA' s discriminative nature. 2. A generative model which places a Gaussian prior on the underlying class variable, and so can model classes with very limited training data.	Lack of robustness to outliers in the language and channel subspaces.

2.6 Score calibration

In a language detection system, the scores from the detectors of different languages may have different score distributions. This necessitates the score adjustment in order to achieve good language recognition performance. Although it may be ideal to adjust the detection scores of different target languages separately, in practice, a global score transformation is performed [150]. A global score adjustment and calibration method often assumes the least prior knowledge. Gaussian backend scores and likelihood ratios are commonly adopted measures by LRE systems [48, 151]. The language scores computed using different classifiers may not be well calibrated. This may be due to the varying utterance length, unequal amount of training data used for target language modeling, etc. To calibrate the language scores, Gaussian backend (GB) [79] followed by multi-class logistic regression (MLR) [152] employing the FoCal toolkit [153] is used prior to final decision. The parameters for calibrating the score was trained in a cheating way, that is, using the evaluation scores themselves [105]. This was done to concentrate on the ability of the underlying models to discriminate between the given classes by not allowing any errors due to miscalibration.

Table 2.2: Number of training and test data segment for 2007 NIST LRE general test set condition.

Languages	Train		Test (30s)
	Seg. Count	Dur.(hrs)	Seg. Count
Arabic	1593	106.24	80
Bengali	1525	165.56	80
Chinese	6265	394.33	398
English	2438	259.09	240
Hindustani	1329	71.00	240
Spanish	730	56.36	240
Farsi	484	24.33	80
German	424†	4.29	80
Japanese	884	44.01	80
Korean	1362	65.39	80
Russian	2090	120.11	160
Tamil	843	130.11	160
Thai	1911	97.32	80
Vietnamese	2437	228.17	160

† Minimum number of training utterances among all the languages, and therefore only 420 utterances per language are considered during classification.

2.7 Databases

2.7.1 2007 NIST LRE

The 2007 NIST language recognition evaluation (LRE) [154] dataset contains 14 target languages in general test set condition. It consists of 7530 conversational telephone speech (CTS) utterances of 30, 10 and 3 seconds duration including the out-of-set data. The training data set includes the speech data extracted from multiple corpora: NIST speaker recognition evaluations (SREs) (2004, 2005, 2006, 2008), NIST 2007 supplementary training dataset, previous NIST LREs, OGI-multilingual, and Babel [155] data-sets. The experiments are performed in closed set condition on 30 seconds duration segments. The distributions of training and test data are shown in Table. 2.3. The number of speech segments along with overall speech durations for each of the target languages are shown for training data. We have also shown the number of test segments for each languages having approximately 30 seconds of duration.

Table 2.3: Distribution of training and test data for 2009 NIST LRE

Languages	Train	Test		
	Seg. Count	30s	10s	3s
Amharic	-	-	-	-
Bosnian	-	-	-	-
Cantonese	250	378	352	341
Creole(Haitian)	250	323	323	323
Croatian	-	-	-	-
Dari	-	-	-	-
English (American)	250	896	863	856
English (Indian)	-	-	-	-
Farsi	250	390	385	384
French	-	-	-	-
Georgian	250	399	399	399
Hausa	-	-	-	-
Hindi	250	667	618	637
Korean	250	463	451	451
Mandarin	250	1015	991	970
Pashto	-	-	-	-
Portuguese	-	-	-	-
Russian	250	511	492	483
Spanish	250	385	385	385
Turkish	250	394	392	394
Ukrainian	-	-	-	-
Urdu	-	-	-	-
Vietnamese	250	315	285	278

2.7.2 2009 NIST LRE

The 2009 NIST LRE [156] dataset contains 23 languages in general test set condition. The dataset contains speech utterances from both Voice of America (VOA) radio broadcasts and CTS. Most of the test segment is from VOA limited to telephone bandwidth. We consider only 12 languages which are sub-set of the NIST-2009 LRE corpora due to limited training data in remaining languages. The sub-set includes American English, Cantonese Chinese, Mandarin Russian, Farsi, Hindi, Korean, Vietnamese, Creole, Georgian, Turki, and Spanish languages. The training data is collected from CTS only, which includes NIST LRE 2007 supplementary dataset, NIST SRE (2004, 2005, 2006 and 2008) and Babel database. The experiments are performed in closed set condition on segments of 30, 10 and 3 seconds duration.

Table 2.4: Summarization of database profiles used in the AP17-OLR challenge

AP17-OL3			AP17-OL3-train/dev			AP17-OL3-test		
Code	**Description**	**Channel**	**No. of Speakers**	**Utt/Spk**	**Total Utt.**	**No. of Speakers**	**Utt/Spk**	**Total Utt.**
ka-cn	Kazakh in China	Mobile	86	50	4300	86	20	1720
ti-cn	Tibetan in China	Mobile	34	330	11220	34	50	1700
uy-id	Uyghur in China	Mobile	353	20	7060	353	5	1765
AP16-OL7			**AP16-OL7-train/dev**			**AP16-OL7-test**		
ct-cn	Cantonese in China Mainland and Hongkong	Mobile	24	320	7680	6	300	1800
zh-cn	Mandarin in China	Mobile	24	300	7680	6	300	1800
id-id	Indonesian in Indonesia	Mobile	24	320	7680	6	300	1800
ja-jp	Japanese in Japan	Mobile	24	320	7680	6	300	1800
ru-ru	Russian in Russia	Mobile	24	300	7680	6	300	1800
ko-kr	Korean in Korea	Mobile	24	300	7680	6	300	1800
vi-vn	Vietnamese in Vietnam	Mobile	24	300	7680	6	300	1800

2.7.3 AP17-OLR

The speech database for oriental language recognition (OLR) are collected from AP16-OL7 and AP17-OL3 database as per AP17-OLR Challenge [157]. The AP16-OL7 [158] database is created by Speechocean Inc. It is divided into three parts: AP16-OL7-train, AP16-OL7-dev, and AP16-OL7-test. The database includes seven languages namely Cantonese, Mandarin, Indonesian, Japanese, Russian, Korean and Vietnamese. The speech signals for Mandarin, Cantonese, Vietnamese and Indonesian were recorded in quiet environment while for Russian, Korean and Japanese, the recordings were conducted in 2 different sessions for each speaker. In the first session, the recordings were conducted in quiet environment and in the second session, the speakers were asked to record the utterances in noisy environment. However, AP17-OL3 database is created as a part of the multilingual minorlingual automatic speech recognition (M2ASR) project in China. The database involves the newly included 3 oriental languages namely Kazakh, Tibetan and Uyghur. The AP17-OL3 database is partitioned into 3 parts: AP17-OL3-train, AP17-OL3-dev and AP17-OL3-test. While creating this database, the sentences of each language were randomly selected from the original M2ASR database. The AP17-OL3 database contains much more variations in terms of recording conditions and number of speakers as compared to the AP17-OL3 database. All the speech signals were recorded in reading style using mobile phones at a sampling rate of 16 kHz and 16 bits/sample resolution. Each language consist of about 10 hours of speech data. The transcriptions of the all the training utterances and lexicons of all the 10 languages are also available to the participants. The details of the databases are described

in Table 2.4. As per the challenge, the participants can use all the mentioned database for training the LR system excluding AP17-OL3-test and required to report the results on development set. The development set includes AP16-OL7-test and AP17-OL3-dev. Note that development and training data sets are non-overlapping, i.e., there is no common speech utterances. The performances are evaluated in closed set condition on three different development data sets corresponding to 1 second, 3 seconds and full-length utterances. The AP17-OLR test set includes the data from AP17-OL3 and AP17-OL7 test sets. The AP17-OL7 database is a newly created database by SpeechOcean. It contains 7 languages as in AP16-OL7, each containing 1800 utterances. The recording conditions of AP17-OL7 are the same as as those of AP16-OL7.

2.8 Performance evaluation metric

Appropriately designed performance metric has been employed for measuring the confidence of the developed LR systems in decision making. In LD task, it is determined whether the unknown speech segment belongs to the claimed language of interest or not, thus posing a two class problem. The basic pair-wise LR performance is represented in terms of miss and false alarm probabilities. The miss probability (also known as false rejection rate (FRR)) is defined as the probability at which the genuine trials are wrongly rejected by the system, whereas the false alarm probability (or false acceptance rate (FAR)) is the percentage in which the system incorrectly accepts the imposter trials as the genuine one. These probabilities are combined into a single number that represents the cost performance of a system, and expressed as

$$C(L_T, L_N) = C_{\text{Miss}} \cdot P_{\text{Target}} \cdot P_{\text{Miss}}(L_T) + C_{\text{FA}} \cdot P_{\text{Non-Target}} \cdot P_{\text{FA}}(L_T, L_N) \tag{2.29}$$

where,

L_T and L_N are the target and non-target languages,

C_{Miss}, C_{FA} and P_{Target} are application model parameters.

For both LRE07 and LRE09, $C_{\text{Miss}} = 1$, $C_{\text{FA}} = 1$ and $P_{\text{Target}} = 0.5$. The range of values of $C(L_T, L_N)$ is 0 to 0.5. The minimum value of 0 is obtained when $P_{\text{Miss}}(L_T) = P_{\text{FA}}(L_T, L_N) = 0$ and the maximum value of 0.5 is obtained if $P_{\text{Miss}}(L_T) = 0$ and $P_{\text{FA}}(L_T, L_N) = 1$ or $P_{\text{Miss}}(L_T) = 1$ and $P_{\text{FA}}(L_T, L_N) = 0$. In addition, an average cost performance which is a measure the cost of taking bad decisions, is used to evaluate the capabilities of one-vs.-all language detection. This primarily used evaluation metric is

considered in various language recognition task [79, 105, 108]. The performance metric as defined in 2007 NIST LRE and 2009 NIST LRE plans is given as

$$C_{avg} = \frac{1}{L}\sum_{L_T}\left\{\begin{array}{r} C_{\text{Miss}} \cdot P_{\text{Target}} \cdot P_{\text{Miss}}(L_T) \\ +\sum_{L_N} C_{\text{FA}} \cdot P_{\text{Non-Target}} \cdot P_{\text{FA}}(L_T, L_N) \\ +C_{\text{FA}} \cdot P_{\text{Out-of-set}} \cdot P_{\text{FA}}(L_T, L_O) \end{array}\right\} \tag{2.30}$$

where,

L is the number of target languages in the (closed set) test,

L_O is the Out-of-set language,

$$P_{\text{Out-of-set}} = \begin{cases} 0.0, & \text{for the closed-set conditions} \\ 0.2, & \text{for the open-set conditions} \end{cases}$$

and

$P_{\text{Non-Target}} = (1 - P_{\text{Target}} - P_{\text{Out-of-set}}) \,/\, (L-1)$.

Prior to the release of 2007 NIST LRE plan, commonly used metric was the equal error rate (EER), computed for each of the target languages. The EER is defined as the point at which FAR and FRR is equal. This can be graphically shown by detection error trade-off (DET) curve. The EER value (in %) varies between 0 to 50. The lower the value of performance metric (C_{avg} or EER), greater the confidence that the segments contains the speech of target language.

On the other hand, the primary evaluation metric used for language identification task is an average identification rate (IDR), and is given by

$$IDR = \frac{1}{L}\sum_{i=1}^{L} S_i \tag{2.31}$$

where S_i is the correct score of the i-th target language.

3

Language Recognition using Sparse Representation

Contents

The sparse representation technique relies on the underlying assumption that the signals can have a compact representation as a linear combination of few columns (atoms) of an overcomplete dictionary. In an ideal condition, the selected dictionary atoms belong to the same class label as that of a query signal. The existing SRC based LR system [100] utilizes sparse coding algorithms over an exemplar dictionary to obtain the sparse coefficients (*s-vector*), which are then employed for classification purpose. The exemplar dictionary is created by simply grouping the suitable vector representation of class-specific training examples. With the increase of a number of training examples (e.g., i-vector), the size of the exemplar dictionary increases, and the computational burden of sparse coding over such dictionary becomes quite prohibitive. To address that, various sparse representation techniques over simple/discriminately learned dictionaries are explored for LR task. The SDC is used as the front-end features for the i-vector framework.

In general, the dictionary learning approaches are based on *batch* or *online* mode. At each iteration, the batch-mode based dictionary learning requires the entire set of training examples to minimize the objective function under some constraints. On the other hand, in the online mode, the dictionary is updated incrementally by taking one training example. To improve the speed of convergence of the online learning, the mini-batch (draws b-training example, $b > 1$) extension is also proposed [159]. The K-SVD [160] and the online learning (OL) [159] dictionaries are two popular examples of the batch and the online dictionary learning modes, respectively. Both approaches employ an iterative procedure that alternates between the sparse representation of the training example based on the current dictionary and the dictionary update stages at each iteration. Apart from the differences in the learning process, the two algorithms also differ in dictionary update rule. In the work reported in this chapter, both K-SVD and OL dictionaries have been explored. Though originally designed for optimizing the reconstruction fidelity, these learned dictionaries are also investigated for classification tasks [101, 161]. The dictionary designed especially for classification can also be categorized into two groups [162]. In the first group, the dictionary is forcibly made discriminative and representation error is used for the classification while in another group the sparse coefficient is made discriminative and used as the input feature to the classifiers. The Meta-face learning [163] and dictionary learning with structured incoherence (DLSI) [164] are two specific examples of the first group. The discriminative-KSVD (D-KSVD) [165], label consistent-KSVD (LC-KSVD) [166], and Fisher discrimination dictionary learning (FDDL) [167] methods belongs to the second group. The D-KSVD and LC-KSVD dictionaries

specially designed for classification purpose are also investigated in this research work. The OMP [168] based sparse coding solves l_0-norm minimization problem while modified version of LAR [169] is used to solve the l_1-norm minimization problem as well as combined l_1 and l_2 (denoted by symbol $l_1 - l_2$) norm minimization problem. Other than SRC, CDS [78], SVM [139], G-PLDA [147], GG [79] and MLR [152] based classifiers are also developed for the contrast purpose. To calibrate the language score, GB followed by MLR is used prior to the final decision. It is to be noted that the GB is essentially the same model as the GG classifier. However, its inputs are the scores from the classifiers rather than the i-vector/GMM supervector as utterance representation.

The rest of this chapter is organized as follows. Section 3.1 provides the basic concepts of l_p-norm and its usage in determining the sparse representation. The LR using a sparse representation with learned-exemplar and discriminately learned dictionaries employing various regularization techniques are also investigated in Section 3.2. The experimental setup is described in Section 3.3 while the results and discussion are presented in Section 3.4. The chapter is summarized in Section 3.5.

3.1 Role of l_p-norm in sparse representation

In this section, we present the basic concepts of l_p-norm followed by its role in determining the sparse representation of a signal.

3.1.1 l_p-norm : Basic concepts

The p-norm or l_p-norm ($1 \leq p \leq \infty$) of an n-dimensional real vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ in Euclidean space is given by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{3.1}$$

- when $p = 0$

$$\|\mathbf{x}\|_0 = \lim_{p \to 0} \|\mathbf{x}\|_p^p = \lim_{p \to 0} \left(\sum_{i=1}^{n} |x_i|^p \right) \tag{3.2}$$

or

$$\|\mathbf{x}\|_0 = \#\left(i : x_i \neq 0\right) \tag{3.3}$$

where the notation # denotes the number of non-zeros elements in vector $\mathbf{x}$.

The l_0-norm is actually not a norm, but a special case of l_p-norm for $p \to 0$, and hence also known as l_0-pseudo-norm. The l_0-norm determines the number of non-zero coefficients in a vector $\mathbf{x}$,

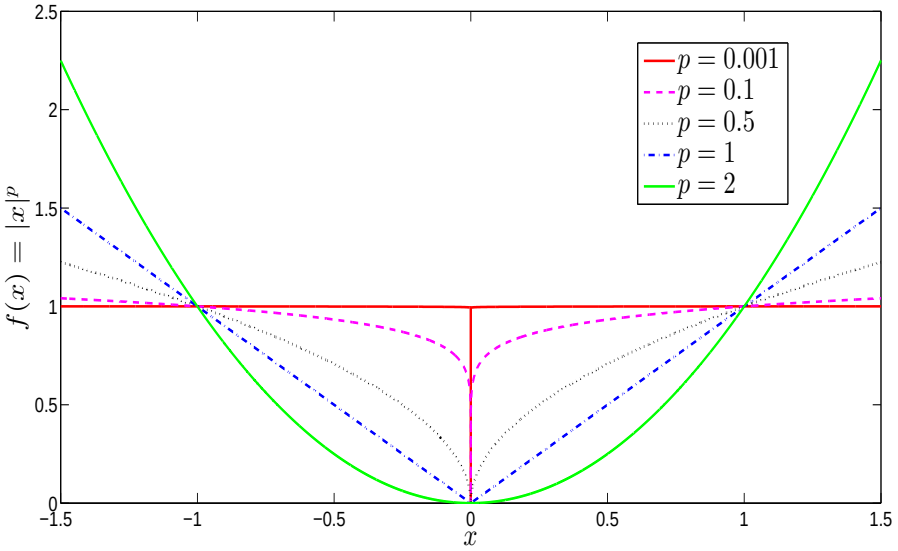

Figure 3.1: Behavior of the scalar function $f(x) = |x|^p$ – the core of l_p-norm computation for various values of p in 1-D space. As $p \to 0$, $f(x) = |x|^p$ approaches the indicator function, which is 0 for $x = 0$ and 1 elsewhere [2].

and is a measure of sparsity of a vector. If vector $\mathbf{x}$ has k non-zero coefficients, the vector is called k-sparse vector. In 1-D case, it is observed from Figure 3.1 that as $p \to 0$, $f(x) = |x|^p$ approaches the indicator function, which is 0 for $x = 0$ and 1 elsewhere. The concave, non-smooth, discontinuous, and global non-differential behavior of l_0-norm function can be seen in Figure 3.1. The geometric interpretation of the p-norm in 2-D space for the value of $p = 0.001$ is shown in Figure 3.2 which is a crisscross.

- when $0 < p < 1$

  The l_p-norm $(0 < p < 1)$ function is a concave, non-smooth, global non-differential function, and can be seen in Figure 3.1. The geometric interpretation of p-norm in 2-D space for the value of $(p = 0.5)$ is shown in Figure 3.2 which is a Astroid.

- when $p = 1$

  The l_1-norm of vector $\mathbf{x}$, i.e., $\|\mathbf{x}\|_1$ represents the sum of the absolute values of the coefficients in the vector $\mathbf{x}$. The l_1-norm function is a convex, non-smooth, and global non-differential function, which can be seen from the Figure 3.1. The geometric interpretation of p-norm in 2-D space for the value of $p = 1$ is shown in Figure 3.2 which is a square with forty-five degree rotation.

- when $p = 2$

  The l_2-norm or Euclidean norm is defined as the square root of the sum of the coefficients square.

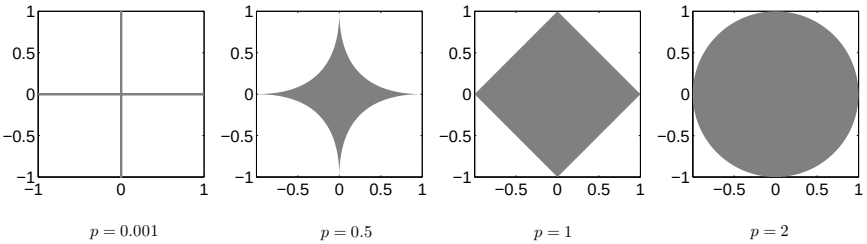

Figure 3.2: Geometric interpretations of the p-norm unit balls for different values of p in 2-D space [3]. The two axes of the above plots are x_1 and x_2.

The l_2-norm function is convex, smooth and global differential function, which can be seen in Figure 3.1. The geometric interpretation of p-norm in 2-D space for the value of $p = 2$ is shown in Figure 3.2 which is a circle.

3.1.2 Sparse solutions of a linear system of equations

Given a target signal $\mathbf{y} \in \mathbb{R}^m$ and a full rank matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m < n$, the underdetermined linear system of equation is defined as $\mathbf{y} = \mathbf{A}\mathbf{x}$. The underdetermined or overcomplete-basis system has lesser number of equations than the number of unknown variables, which results in infinitely many solutions. In order to narrow the choice to one well-defined solution, additional criteria are needed. A familiar way to do this is regularization, where a function $J(\mathbf{x})$ is introduced to govern the kind of solution(s). The general optimization problem is defined as

$$\min_{\mathbf{x}} J(\mathbf{x}) \quad \text{subject to} \quad \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{3.4}$$

The unique solution of the Equation 3.4 is guaranteed by selecting a strictly convex function $J(.)$. Considering $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$, the squared Euclidean norm, the optimization problem is given by

$$\min_{\mathbf{x}} \|\mathbf{x}\|_2^2 \quad \text{subject to} \quad \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{3.5}$$

The unique solution of the Equation 3.5 is given by

$$\hat{\mathbf{x}} = \mathbf{A}^{\dagger}\mathbf{x} = \mathbf{A}^{'}(\mathbf{A}\mathbf{A}^{'})^{-1}\mathbf{y}. \tag{3.6}$$

where $\mathbf{A}^{\dagger}$ and $\mathbf{A}^{'}$ are the pseudo-inverse and transpose of the matrix $\mathbf{A}$, respectively. It is to be

noted that the matrix $\mathbf{A}$ is assumed to be full rank, and hence the matrix $\mathbf{A}\mathbf{A}^{'}$ is positive definite and thus invertible. *The solution* $\hat{\mathbf{x}}$ *obtained using* l_2*-norm minimization is not the sparse solution.*

Considering $J(\mathbf{x}) = \|\mathbf{x}\|_0$, the optimization problem for solving the sparsest solution (sparse code) $\hat{\mathbf{x}}$ is formulated as

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{3.7}$$

This is a non-convex combinatorial optimization problem, and it has proven that obtaining an exact solution to the Equation 3.7 is, in general NP hard [170]. However, there exist a number of pursuit algorithms to find the approximate solution. One of the popular sparse coding algorithm among those is OMP [168]. The OMP is an iterative greedy algorithm and at each step it selects a dictionary atom that is most correlated with the current residual. In order to obtain the approximate solution, the optimization problem given in Equation 3.7 is reformulated as

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \leq \epsilon \tag{3.8}$$

or

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq \gamma \tag{3.9}$$

or

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda_0\|\mathbf{x}\|_0 \tag{3.10}$$

where ϵ is the error bound, γ and λ_0 are the sparsity constraint (i.e., controls the number of non-zero elements). The solution $\hat{\mathbf{x}}$ is the sparse representation of the target signal/vector $\mathbf{y}$. The sparse representation refers to the representation of a target signal using a linear combination of only few dictionary atoms.

Another well known pursuit algorithm used for computing sparse representation is basis pursuit (BP) [171]. In BP, the optimization problem is obtained by setting the function $J(\mathbf{x}) = \|\mathbf{x}\|_1$. It uses l_1-norm constraint instead of l_0-norm which makes the optimization problem convex. The optimization problem based on l_1-norm minimization is formulated as

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{3.11}$$

This problem can be solved using general purpose solvers such as simplex and interior point methods [172] in order $\mathcal{O}(n^3)$ which is slow for large scale problems. If data $\mathbf{y}$ is noisy, the optimization problem

given in Equation 3.11 is reformulated as

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \leq \epsilon \tag{3.12}$$

or

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda_1\|\mathbf{x}\|_1 \tag{3.13}$$

where λ_1 is a regularization parameter that controls the trade-off between sparsity and reconstruction fidelity. The problem mentioned in Equation 3.13 is called basis pursuit denoising (BPDN) in signal processing domain. In the statistical community, the optimization problem related to Equation 3.13 is well known as Lasso [173] and formulated as

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_1 \leq q \tag{3.14}$$

where q is the l_1-norm constraint on variables. The matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is implicitly assumed to have $m > n$, i.e. representing an overdetermined linear system. The Lasso is an efficient regularization method for estimating the sparse solution in linear models. It regularizes or shrinks a fitted model through an l_1 penalty or constraint. The problem mentioned in Equation 3.13 is equivalent to Equation 3.14 under an appropriate correspondence of parameters. If $\tilde{\mathbf{x}}$ is a solution to Equation 3.13 for some $\lambda_1 \geq 0$, it also solves Equation 3.14 for $q = \|\tilde{\mathbf{x}}\|_1$ [174]. The Lasso is not robust when predictors (dictionary atoms) are highly correlated. It will arbitrarily choose one and ignore the others. The Lasso selects not more than m variables if $n \gg m$. To overcome these limitations, there exist an ENet [175] regularized regression method which linearly combines l_1 and l_2 penalties of the Lasso and the ridge regression (RR), and is given as

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda_1\|\mathbf{x}\|_1 + \frac{\lambda_2}{2}\|\mathbf{x}\|_2^2 \tag{3.15}$$

where λ_1 and λ_2 are the positive regularization coefficients. The l_1 penalty term of Equation 3.15 does the variable selection while the l_2 part does the grouped selection and stabilizes the solution paths with respect to random sampling, thereby improving prediction [175]. In ENet, a group of highly correlated variables tend to have coefficients of same magnitude. When $n \gg m$, the ENet select more than m variables unlike the Lasso. The ENet problem simplifies to RR when $\lambda_1 = 0$ and to the Lasso when $\lambda_2 = 0$. The Equation (3.15) can be transformed into an equivalent Lasso problem as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y}^* - \mathbf{A}^*\mathbf{x}\|_2^2 + \lambda_1\|\mathbf{x}\|_1 \tag{3.16}$$

where

$$\mathbf{y}^* = \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix}_{(m+n)\times 1} \quad \text{and} \quad \mathbf{A}^* = \begin{pmatrix} \mathbf{A} \\ \sqrt{\lambda_2}\mathbf{I} \end{pmatrix}_{(m+n)\times n} \tag{3.17}$$

Note that the dimension of $\mathbf{y}^*$ is $(m+n)$ and $\mathbf{A}^*$ has a rank of n. This means that in all situations the elastic net can selects all n variables, and thus overcome the limitation of Lasso. The details of sparse coding algorithms such as OMP and a modified version of LAR referred as LARS are provided in Appendix A. The Lasso-based regularization problem can be efficiently solved using the LARS algorithm. The same algorithm can be used to solve the ENet-based regularization problem after transforming the ENet-based problem into Lasso-based one.

The success of any sparse coding algorithm depends on the choice of matrix or dictionary $\mathbf{A}$. In general, the dictionaries are either parametric (analytic) or data-driven. The examples of the analytic dictionary include Fourier [176], wavelet [177] , curvelet [178] and contourlet [179] transforms where a pre-specified set of mathematical functions is employed to represent the target data. The resulting dictionary usually leads to simple and fast algorithms which do not involve multiplication by the dictionary matrix for computing the sparse representation [176]. The model functions used in the analytic dictionaries are limited and over-simplistic compared to the complexity of many natural signals which is a major drawback of the analytic dictionary. On the other hand, data-driven dictionaries are derived from the training data (i.e., exemplar dictionary) itself or using a learning algorithm. The method of optimal directions (MOD) [180], K-SVD [160], D-KSVD [165], LC-KSVD [166] and OL dictionary [181] are few examples of dictionary learning algorithms. In the subsequent sections, we have explored the use of sparse representation for LR task, considering exemplar and various learned dictionaries like K-SVD, D-KSVD, LC-KSVD, and OL. The steps involved in learning these dictionaries are provided in Appendix B. The proposed sparse representation based LR systems are contrasted with the i-vector based LR systems employing various classifiers like CDS, SVM, MLR, G-PLDA, and GG.

3.2 Sparse representation based language recognition

In this section, we have discussed sparse representation over an exemplar, K-SVD/OL learned dictionary and class-based K-SVD/OL learned dictionary. The K-SVD/OL were originally designed for reconstruction task, and hence, in addition, the discriminatively learned dictionary which was

specially designed for classification tasks are also described. The proposed techniques are applied to two spoken utterance representations: the i-vector and the GMM mean supervector. Therefore, the training and test vectors represent either the i-vector or the GMM mean supervector.

3.2.1 Sparse coding over exemplar dictionary based LR

Assume there are L distinct languages in the training set with lth language containing n_l example utterances. Let $\mathbf{y}_{lj} \in R^m$ denote the suitable m-dimensional vector representation for the jth example of the lth language, where $l = 1, 2, \ldots, L$ and $j = 1, 2, \ldots, n_l$ denote the indices of the languages and the training examples in the lth language, respectively. It is assumed that a test vector $\mathbf{w} \in R^m$ belonging to the lth language class can be approximated as,

$$\mathbf{w} \approx a_{l1}\mathbf{y}_{l1} + a_{l2}\mathbf{y}_{l2} + \cdots + a_{ln_l}\mathbf{y}_{ln_l} \tag{3.18}$$

where $\{a_{lj}\}_{j=1}^{n_l}$ are the real scalar coefficients.

For computing the sparse representation, an exemplar dictionary $\mathbf{Y}$ is formed by stacking the sub-matrices corresponding to L languages as,

$$\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_L] \in R^{m \times n}, \;\; n = \textstyle\sum_{l=1}^{L} n_l \tag{3.19}$$

where the lth matrix $\mathbf{Y}_l = [\mathbf{y}_{l1}, \mathbf{y}_{l2}, \ldots, \mathbf{y}_{ln_l}] \in R^{m \times n_l}$ is formed by stacking all training vectors corresponding to that language.

Now the test vector $\mathbf{w}$ is approximated as linear combination of n columns of the exemplar dictionary $\mathbf{Y}$ as

$$\mathbf{w} \approx \mathbf{Y}\mathbf{s} \tag{3.20}$$

where $\mathbf{s} \in R^n$ is the vector of unknown coefficients. With the assumption made in Equation 3.18, the vector $\mathbf{s}$ is expected to be sparse. It is known as *s-vector* corresponding to a test vector $\mathbf{w}$. The sparse solution $\hat{\mathbf{s}}$ to Equation 3.20 can be obtained by solving any of the following optimization problems:

- l_0-norm minimization based sparse coding problem:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \|\mathbf{w} - \mathbf{Y}\mathbf{s}\|_2^2 \;\text{ subject to }\; \|\mathbf{s}\|_0 \leq \gamma \tag{3.21}$$

  where γ is the chosen constraint on the sparsity controlling the number of nonzero coefficients in the s-vector $\hat{\mathbf{s}}$. The OMP sparse coding algorithm is used to solve the Equation 3.21.

- BPDN or Lasso sparse coding problem:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \frac{1}{2}\|\mathbf{w} - \mathbf{Y}\mathbf{s}\|_2^2 + \lambda_1\|\mathbf{s}\|_1 \tag{3.22}$$

  where λ_1 is the positive regularization coefficient, which controls the number of nonzero coefficients in the solution $\hat{\mathbf{s}}$. The modified LAR for Lasso (LARS) is used to solve the Lasso based sparse coding problem.

- ENet sparse coding problem:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \frac{1}{2}\|\mathbf{w} - \mathbf{Y}\mathbf{s}\|_2^2 + \lambda_1\|\mathbf{s}\|_1 + \frac{\lambda_2}{2}\|\mathbf{s}\|_2^2 \tag{3.23}$$

  where λ_1 and λ_2 are the positive regularization coefficients. The l_1 penalty term of Equation 3.23 does the variable selection while the l_2 part does the grouped selection and stabilizes the solution paths with respect to random sampling, thereby improving prediction. The Equation 3.23 is transformed into an equivalent Lasso problem and solved using LARS. The Lasso sparse coding problem is a special case of Equation 3.23 with $\lambda_2 = 0$.

In the s-vector $\hat{\mathbf{s}}$, ideally all nonzero coefficients should correspond to the dictionary atoms (columns) from the language class to which the test vector $\mathbf{w}$ belongs to. This is valid based on the assumption in Equation 3.18. However, in practice, the s-vector does have some nonzero coefficients for the atoms other than those belonging to the class of $\mathbf{w}$ owing to modeling error, noise and session/channel variability.

3.2.1.1 Language identification

The LID is done by comparing the class-wise mean of the sparse representation vector as

$$\arg\max_{l} \left(\frac{1}{n_l} \sum_{j=1}^{n_l} \hat{\mathbf{s}}_{lj} \right) \tag{3.24}$$

where $\hat{\mathbf{s}}_{lj}$ is the jth sparse coefficient of lth language class. The unknown test vector is assigned the class of the maximum class-wise mean. The classification can also be done by comparing the class-wise reconstruction error as,

$$\arg\min_{l} \|\mathbf{w} - \mathbf{Y}_l\hat{\mathbf{s}}_l\|_2^2 \tag{3.25}$$

Here, the class leading to minimum reconstruction error is assigned as the class of the unknown test vector. Both of these approaches are referred to as the SRC. The performance of the LID system is

measured by the average language identification rate (IDR). The higher the value of IDR, greater the confidence in the identified language of the segment.

3.2.1.2 Language detection

The language detection using sparse representation is performed by comparing the target (the language of interest or claimed) and the non-target scores. These scores are determined by taking the mean of the sparse representation vector corresponding to the target and the non-target sparse coefficients. The maximum among these two scores is used to decide whether the claimed language is present or not. The detection may be true or false. The primary performance measure for language detections is an average detection cost C_{avg} metric as defined in 2007 NIST LRE [154]. The lower the value of C_{avg}, greater confidence that the segment contains the speech of target language.

3.2.2 Sparse coding over learned dictionary based LR

The learning of dictionary atoms (i.e., bases) from the data instead of using off-the-shelf bases (e.g., cosine and wavelet) leads to state-of-the-art results in many applications such as image denoising [182], image classification [183], speaker verification [101], etc. The learned dictionary can be designed to have fewer bases, and hence finding a sparse representation of test vectors over a reduced sized learned dictionary is computationally less expensive than that of an exemplar dictionary (language-wise concatenation of training i-vectors). The dictionary learning problem can be divided into two subproblems: a) finding the sparse solution of the data samples based on the initial dictionary or that obtained from the previous iteration, and b) update the dictionary atoms keeping the s-vector fixed. These two subproblems are solved in an iterative manner while fixing one of the solutions. The regularization term in the sparse coding stage is based on either l_0, l_1 and $l_1 - l_2$ norms.

3.2.2.1 K-SVD/OL dictionary learning with l_0 regularization

Given a set $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$ of training vectors and the sparsity constraint γ', the dictionary learning problem can be defined as:

$$\min_{\mathbf{D},\mathbf{S}} \left\{ \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{D}\mathbf{s}_i\|_2^2 \right\} \quad \text{subject to} \quad \|\mathbf{s}_i\|_0 \leq \gamma' \quad \forall i \tag{3.26}$$

or

$$\min_{\mathbf{D},\mathbf{S}} \{\|\mathbf{Y} - \mathbf{D}\mathbf{S}\|_F^2\} \quad \text{subject to} \quad \|\mathbf{s}_i\|_0 \leq \gamma' \quad \forall i \tag{3.27}$$

where $\mathbf{D}$ is the learned dictionary of size $m \times k$, where m denotes the dimensionality of training vectors and k is the number of atoms. The set of sparse vector (s-vector) corresponding to $\mathbf{Y}$ is denoted as $\mathbf{S} = \{\mathbf{s}_i\}_{i=1}^{N}$ and $\mathbf{s}_i$ being the s-vector for the ith training vector. An alternative form of Equation 3.26 is given by

$$\min_{\mathbf{D},\mathbf{S}} \left\{ \sum_{i=1}^{N} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\mathbf{s}_i\|_2^2 + \lambda_0' \|\boldsymbol{s}_i\|_0 \right\} \tag{3.28}$$

where λ_0' is the dictionary sparsity parameter which controls the number of non-zeros elements in the s-vector. The solution to Equation 3.26 or Equation 3.28 can be obtained by solving the sparse coding sub-problem and dictionary update subproblems in an iterative fashion. The OMP algorithm can be used to find the sparse solution while the dictionary update rule of K-SVD/OL can be used for updating the atoms of the dictionary.

3.2.2.2 K-SVD/OL dictionary learning with l_1 regularization

In this, the dictionary is created by considering l_1 regularization term in sparse coding stage and dictionary update rule of either K-SVD and OL dictionary . The dictionary learning problem is given as

$$\min_{\mathbf{D},\mathbf{S}} \left\{ \sum_{i=1}^{N} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\mathbf{s}_i\|_2^2 + \lambda_1' \|\boldsymbol{s}_i\|_1 \right\} \tag{3.29}$$

where λ_1' is the dictionary sparsity parameter which controls the number of non-zero elements in the s-vector.

3.2.2.3 K-SVD/OL dictionary learning with $l_1 - l_2$ regularizations

In this approach, the learned dictionary is created by $l_1 - l_2$ regularization terms in sparse coding stage and dictionary update rules of either K-SVD or OL dictionary. The dictionary learning problem is given as

$$\min_{\mathbf{D},\mathbf{S}} \left\{ \sum_{i=1}^{N} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\mathbf{s}_i\|_2^2 + \lambda_1' \|\mathbf{s}_i\|_1 + \frac{\lambda_2'}{2} \|\mathbf{s}_i\|_2^2 \right\} \tag{3.30}$$

- s-vector extraction and scoring

  Once the dictionary $\mathbf{D}$ is obtained following any of the learning approaches discussed in Equations 3.26, 3.29 and 3.30, the train vector $\mathbf{y}$ and test vector $\mathbf{w}$ can be modeled as the linear combination of dictionary atoms:

$$\mathbf{y} = \mathbf{D}\mathbf{s}_{\mathbf{y}} \tag{3.31}$$

$$\mathbf{w} = \mathbf{D}\mathbf{s}_{\mathbf{w}} \tag{3.32}$$

where $\mathbf{s_y}$ and $\mathbf{s_w}$ are the s-vectors for the train and test data, respectively. These s-vectors can be considered as the new representation of the speech utterances. The s-vector extraction uses similar type of sparse coding algorithm as used during dictionary learning. Unlike the exemplar case, the atoms of the learned dictionary could not be assigned to language classes in a straightforward manner. Thus, the s-vectors corresponding to train and test data vectors are determined with respect to the learned dictionary using Equation 3.31 and Equation 3.32, and are applied as inputs to CDS classifier to obtain the score. The mean of the language specific coefficients in the s-vector is computed to obtain the score. Finally, the scores are calibrated, and are used for language identification and detection.

3.2.3 Sparse coding over class-specific learned dictionary based LR

The learned dictionaries are more efficient than the exemplar ones, but they happen to be deficient in two aspects:

(i) During dictionary learning process, the class labels of the training data do not get retained. Thus, the class labels of the atoms of the resulting dictionary are unknown.

(ii) The number of atoms corresponding to each of the classes can not be controlled. Thus, there is no guarantee that the set of atoms belonging to different classes in the dictionary may not be balanced across the classes.

The first aspect rules out the direct application of the SRC approach, whereas the second one is more critical as it prohibits from keeping the same sparsity constraint same for all the classes. To overcome these challenges, we have also explored the class-based K-SVD/OL learned dictionary based SRC for LR task.

3.2.3.1 Class-based KSVD/OL dictionary learning with l_0 regularization

Given the training data for the lth language containing n_l numbers of m-dimensional vectors $\mathbf{Y}^l = \{\mathbf{y}_i^l\}_{i=1}^{n_l}$ and the constraint on sparsity as γ', the problem of class-based dictionary learning can be defined as,

$$\min_{\mathbf{D}^l, \mathbf{S}^l} \|\mathbf{Y}^l - \mathbf{D}^l\mathbf{S}^l\|_F^2 \text{ subject to } \|\mathbf{s}_i^l\|_0 \leq \gamma' \quad \forall i \tag{3.33}$$

where $\mathbf{D}^l$ is the language specific learned dictionary having k columns and $l = 1, \ldots, L$. The matrix $\mathbf{S}^l$ denotes a set of sparse vectors corresponding to $\mathbf{Y}^l$ with $\mathbf{s}_i^l$ being the sparse vector for the ith vector

of the lth language.

3.2.3.2 Class-based K-SVD/OL dictionary learning with l_1 regularization

The l_1 regularized (Lasso) class-based dictionary learning problem is formulated as

$$\min_{\mathbf{D}^l,\mathbf{S}^l}\left\{\sum_{i=1}^{n_l}\frac{1}{2}\|\mathbf{y}_i^l-\mathbf{D}^l\mathbf{s}_i^l\|_2^2+\lambda_1'\|\mathbf{s}_i^l\|_1\right\} \tag{3.34}$$

where $\mathbf{y}_i^l$ is the ith training data of the lth language. Equation. 3.34 can be efficiently solved by performing the sparse coding stage and dictionary update stage iteratively while keeping one of them fixed. The LARS sparse coding algorithm is used to solve the Lasso-based sparse coding problem, while the dictionary atoms are updated using K-SVD/OL dictionary update rule.

3.2.3.3 Class-based K-SVD/OL dictionary learning with l_1-l_2 regularization

The l_1-l_2 regularization (ENet) class-based dictionary learning problem is formulated as

$$\min_{\mathbf{D}^l,\mathbf{S}^l}\left\{\sum_{i=1}^{n_l}\frac{1}{2}\|\mathbf{y}_i^l-\mathbf{D}^l\mathbf{s}_i^l\|_2^2+\lambda_1'\|\mathbf{s}_i^l\|_1+\frac{\lambda_2'}{2}\|\mathbf{s}_i^l\|_2^2\right\} \tag{3.35}$$

Equation. (3.35) can be efficiently solved by performing the sparse coding stage and dictionary update stage iteratively while keeping one of them fixed. The ENet based sparse coding problem is transformed into Lasso problem and solved using LARS algorithm, while the dictionary atoms are updated using the K-SVD/OL dictionary update rule.

- s-vector extraction and scoring:

  The language specific dictionaries obtained using any of the dictionary learning approaches as mentioned in Equations 3.36, 3.34 and 3.35 are combined to form a single learned-exemplar dictionary $\mathbf{D}_{\text{LD-XR}}$ as,

  $$\mathbf{D}_{\text{LD-XR}}=[\mathbf{D}^1\mid\mathbf{D}^2\mid\ldots\mid\mathbf{D}^L] \tag{3.36}$$

  where '|' denotes a horizontal concatenation operator. For sparse representation, the test vector $\mathbf{w}$ can be represented as the linear combination of the dictionary atoms as

  $$\mathbf{w}=\mathbf{D}_{\text{LD-XR}}\mathbf{s} \tag{3.37}$$

  where $\mathbf{s}$ is the s-vector corresponding to the test vector. The computation of s-vectors are done with the similar sparse coding algorithm as used in dictionary learning. In this approach, the class-labels of the s-vector coefficients are known, and SRC can be used for LR.

3.2.4 Discriminative-KSVD dictionary based language recognition

The D-KSVD [165] is a dictionary learning approach which incorporates classification error term in addition to reconstruction error used in classical K-SVD. The idea is to simultaneously learn a dictionary $\hat{\mathbf{D}}$ and a linear classifier $\hat{\mathbf{W}}$ by solving the joint optimization dictionary learning problem. The D-KSVD dictionary learning problem employing l_0-norm regularization is formulated as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{W}}, \hat{\mathbf{S}} \rangle = \underset{\mathbf{D},\mathbf{W},\mathbf{S}}{\arg\min} \left\{ \|\mathbf{Y} - \mathbf{DS}\|_F^2 + \beta \|\mathbf{H} - \mathbf{WS}\|_F^2 \right\} \quad \text{such that} \quad \|\mathbf{s}_i\|_0 \leq \gamma' \ \ \forall i \tag{3.38}$$

where $\mathbf{Y} \in R^{m \times N}$ is the set of training data vectors, $\mathbf{D} \in R^{m \times k}$ is the learned dictionary, $\mathbf{S} \in R^{k \times N}$ is the set of sparse codes corresponding to $\mathbf{Y}$, β is the regularization parameter, the matrix $\mathbf{H} \in R^{L \times N}$ contains the class label of training data vectors, $\mathbf{W} \in R^{L \times k}$ is a linear classifier.

For solving Equation 3.38 using the classical K-SVD algorithm, we could rearrange it as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{W}}, \hat{\mathbf{S}} \rangle = \underset{\mathbf{D},\mathbf{W},\mathbf{S}}{\arg\min} \left\| \begin{pmatrix} \mathbf{Y} \\ \sqrt{\beta}\mathbf{H} \end{pmatrix} - \begin{pmatrix} \mathbf{D} \\ \sqrt{\beta}\mathbf{W} \end{pmatrix} \mathbf{S} \right\|_F^2 \quad \text{such that} \quad \|\mathbf{s}_i\|_0 \leq \gamma' \ \ \forall i, \tag{3.39}$$

Once the dictionary $\mathbf{D}$ is obtained, the OMP algorithm can be used to compute sparse coefficient vector $\mathbf{s}$ of the test vector over learned dictionary $\mathbf{D}$. The class-similarity vector $\mathbf{c_w}$ of the test vector can be obtained by applying predicted linear classifier $\hat{\mathbf{W}}$ to the sparse coefficient $\mathbf{s}$, i.e., $\mathbf{c_w} = \hat{\boldsymbol{W}}\mathbf{s}$. Finally, the class label of the test vector is decided by noting the index corresponding to the maximum value of $\mathbf{c_w}$. The D-KSVD dictionary design is further extended with l_1-norm regularization (Lasso) and $l_1 - l_2$ norm regularization (ENet).

3.2.5 Label consistent-KSVD dictionary based language recognition

In LC-KSVD [166] dictionary learning approach, discriminative sparse-code error term enforcing label consistency was incorporated in addition to the terms (i.e., reconstruction and classification error term) used in D-KSVD formulation. The LC-KSVD dictionary learning problem employing l_0-norm regularization is formulated as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{A}}, \hat{\mathbf{W}}, \hat{\mathbf{S}} \rangle = \underset{\mathbf{D},\mathbf{A},\mathbf{W},\mathbf{S}}{\arg\min} \left\{ \|\mathbf{Y} - \mathbf{DS}\|_F^2 + \alpha \|\mathbf{Q} - \mathbf{AS}\|_F^2 + \beta \|\mathbf{H} - \mathbf{WS}\|_F^2 \right\} \quad \text{s.t.} \quad \|\mathbf{s}_i\|_0 \leq \gamma' \ \ \forall i \tag{3.40}$$

where $\mathbf{Y} \in R^{m \times N}$ is the set of training data vectors, $\mathbf{D} \in R^{m \times k}$ is the learned dictionary, $\mathbf{S} \in R^{k \times N}$ is the set of sparse codes corresponding to $\mathbf{Y}$, matrix $\mathbf{H} \in R^{L \times N}$ contains the class label of training data

vectors, and $\boldsymbol{W} \in R^{L\times k}$ is a linear classifier. The matrix $\mathbf{Q} \in R^{k\times N}$ is the discriminative sparse codes matrix promoting label consistency (refer Appendix B for details), $\mathbf{A} \in R^{k\times k}$ is a linear transformation, α and β are the regularization parameters used to balance the discriminative sparse code errors and classification error to overall objective function, respectively. For solving through the classical K-SVD algorithm, Equation 3.40 can be rearranged as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{A}}, \hat{\mathbf{W}}, \hat{\mathbf{S}} \rangle = \underset{\mathbf{D},\mathbf{A},\mathbf{W},\mathbf{S}}{\arg\min} \left\| \begin{pmatrix} \mathbf{Y} \\ \sqrt{\alpha}\mathbf{Q} \\ \sqrt{\beta}\mathbf{H} \end{pmatrix} - \begin{pmatrix} \mathbf{D} \\ \sqrt{\alpha}\mathbf{A} \\ \sqrt{\beta}\mathbf{W} \end{pmatrix} \mathbf{S} \right\|_F^2 \quad \text{s.t.} \quad \|\mathbf{s}_i\|_0 \leq \gamma' \ \ \forall i \tag{3.41}$$

Once the dictionary $\mathbf{D}$ is obtained, the sparse coefficient vector $\mathbf{s}$ corresponding to the test vector can be computed by using any suitable sparse coding algorithm such as OMP. Now, on applying the predicted linear classifier $\hat{\mathbf{W}}$ to the sparse vector $\mathbf{s}$, we obtain a class-similarity vector $\mathbf{c_w}$ as $\mathbf{c_w} = \hat{\boldsymbol{W}}\mathbf{s}$. Finally, the class label of the test vector is decided based on the class label associated with the index in vector $\mathbf{c_w}$ attaining the maximum value. The LC-KSVD dictionary design is further extended with l_1-norm regularization (Lasso) and $l_1 - l_2$ norm regularization (ENet).

The special case of LC-KSVD problem with $\beta = 0$ is referred as LC-KSVD1 while with non-zeros values of regularization parameters α and β, it is referred as LC-KSVD2. The learning procedure of LC-KSVD1 is same as of LC-KSVD2, however the classifier $\mathbf{W}$ for LC-KSVD1 is trained separately after the computation of $\mathbf{D}$, $\mathbf{A}$, and $\mathbf{S}$. The estimate of linear classifier $\hat{\mathbf{W}}$ in case of LC-KSVD1 is obtained by solving the equation

$$\hat{\mathbf{W}} = \mathbf{H}\hat{\mathbf{S}}'(\hat{\mathbf{S}}\hat{\mathbf{S}}' + \lambda\mathbf{I})^{-1} \tag{3.42}$$

where λ is the l_2-norm regularization coefficient and $\mathbf{I}$ is the identity matrix. The dictionary initialization step of LC-KSVD differs from D-KSVD. In the LC-KSVD, the dictionary is initialized using a single learned-exemplar dictionary while the initial dictionary in D-KSVD uses pooled data from all classes [184]. In our implementation, a single learned-exemplar dictionary based initialization is considered in both D-KSVD and LC-KSVD based approaches. This has been done to know the actual performance gains of the algorithms.

3.3 Experimental setup

Various sparse coding-based LR systems are developed employing different variants of dictionaries created using both the GMM mean supervector and the i-vector as the utterance representation. The recognition performances of these sparse coding-based LR systems are primarily contrasted with the i-vector employing various classifiers. The dictionary learning approaches are based on popular K-SVD and OL. In the following subsections, we have provided the details of developed systems.

3.3.1 Dataset and system parameters

The experiments are performed on the 2007 NIST language recognition evaluation (LRE) database. It contains speech data from 14 languages collected from CTS. The performance evaluation of the system is done as per the NIST 2007 LRE evaluation plan [154]. In this work, we have focused on the closed set with test utterances having 30 seconds duration. The full training data is used for the system development (i.e., for building the GMM-UBM and a T-Matrix). However, a balanced training data consisting of 420 speech utterances collected from CTS are considered during classification. The distributions of the training and test utterances are described in Section 2.7.1.

The speech signals are analyzed with 20 ms Hamming window with 10 ms shift and a pre-emphasis factor of 0.97. Each frame is converted into 7-dimensional ($c_0 - c_6$) base MFCCs [9] by considering 23-channel Mel filterbank. The acoustic MFCC features are normalized using VTLN [130] and CMN [126]. Then, 49-dimensional shifted delta cepstral (SDC) [125] coefficients using 7-1-3-7 scheme are obtained and appended to 7 static MFCCs to form 56-dimensional feature vector. A energy based VAD is used to remove the frames corresponding to silence regions.

A language-independent GMM-UBM of 1024 Gaussians is employed to build the LR system by pooling features from all 14 languages. For each utterance, the corresponding acoustic feature vectors are mapped to an i-vector using T-matrix of rank 400 based on a GMM with 1024 mixture components. The dimensionality of the i-vector is chosen based on the experimentation. The GMM mean supervector is formed by concatenating the mean vectors corresponding to utterance adapted GMM-UBM. For the dimensionality of acoustic features being 56 and the number of mixtures in GMM-UBM being 1024, the resulting supervectors turn out to have the dimensionality of $56 \times 1024 = 56$ K.

The WCCN and LDA matrices are created using balanced training data and are used to compensate for the session/channel variations present in the i-vector. The size of WCCN matrix is 400×400. For

the number of target languages being 14, the dimensionality of the LDA compensated i-vector reduces to 13, i.e., one less than the number of target languages. For compensating the nuisances present in the GMM mean supervectors, the JFA is employed. The number of language factor (LF) and channel factor (CF) are chosen to be 14 and 400, respectively.

A host of simple and discriminative classifiers such as SVM, G-PLDA, MLR, GG, CDS, and SRC, are explored for computing the language scores. For score-calibration purpose, techniques like GB, MLR and GB followed by MLR (GB+MLR) implemented in multi-class FoCal toolkit [153], are reapplied. The performance of the language detection systems is primarily evaluated using average detection cost function C_{avg}. However, for language identification, an average identification rate (IDR) is used for an evaluation metric. The details are described in Section 2.8.

3.4 Results and discussion

In this section, various experimental results of the proposed sparse representation based LR systems as well as of contrast systems on NIST 2007 LRE are presented and discussed. The tuning of system parameters is done to get the best performance.

3.4.1 Contrast LR systems

In this subsection, we have investigated the i-vector based LR system with various classifiers (SVM, G-PLDA, MLR, GG, and CDS) for contrast purpose. For session/channel compensation, LDA, WCCN, and LDA followed by WCCN (LDA+WCCN) compensation techniques are used for i-vector. However, JFA is used for mitigating the session/channel effect in supervector based utterance representation. Three calibration techniques namely GB, MLR and GB+MLR are used for score calibration. The performance of i-vector based LR system employing CDS classifier with different session/channel compensation and calibration techniques are summarized in Table 3.1. It is to be noted that the performance of i-vector based LR system with WCCN compensation and GB+MLR calibration is superior compared to LR systems which use LDA or LDA+WCCN compensation with other two calibration schemes. The work is further extended with other mentioned classifiers considering the WCCN compensation and GB+MLR calibration scheme. The results are summarized in Table 3.2 and the best performance is noted with GG classifier.

Table 3.1: Performances of LDA/WCCN/LDA+WCCN compensated i-vector based LR systems employing CDS classifier with different score calibration schemes.

Calibration	LDA		WCCN		LDA+WCCN	
	LD	LID	LD	LID	LD	LID
	$100 \times C_{avg}$	IDR (%)	$100 \times C_{avg}$	IDR (%)	$100 \times C_{avg}$	IDR (%)
None	6.98	82.95	6.49	83.04	6.33	83.66
GB	3.71	88.67	3.17	89.82	3.77	88.79
MLR	4.87	85.62	5.24	86.16	5.04	85.90
GB+MLR	3.46	88.93	**3.13**	**89.84**	3.43	89.15

Table 3.2: Performance comparison between i-vector based LR with different classifiers (SVM:support vector machine, G-PLDA:Gaussian PLDA, MLR:multi-class logistic regression, GG:generative Gaussian, CDS:cosine distance scoring). The scores are calibrated using GB+MLR.

Classifier	Compensation Technique	LD	LID
		$100 \times C_{avg}$	IDR (%)
CDS	WCCN	3.13	89.84
GG		**3.07**	**90.33**
MLR		3.18	90.10
SVM		3.20	89.96
G-PLDA	None	3.28	89.14

3.4.2 Language recognition using sparse representation over an exemplar dictionary

In this subsection, we explore the SRC based LR system where an undercomplete dictionary is created by concatenating all GMM mean shifted supervectors corresponding to the training data. For contrasting the performance, the SRC with overcomplete dictionary, another system is developed using the i-vectors for the same task. The resultant dictionary in both the situation is known as *exemplar dictionary*. Three sparse coding algorithms: OMP, Lasso and ENet are used for finding the s-vectors, which are further used for classification. The class-wise mean are computed to obtain the final scores followed by score calibration. The GB+MLR is used to for score calibration.

The reason behind the undercomplete dictionary is the length of supervector which is quite large. Typically the number of Gaussian mixtures is in the order of 1 K and the acoustic features (e.g., SDC) of 56 dimensions. For developing an overcomplete exemplar dictionary in such cases, one would require much more than 56 K training examples. The collection of such a huge training dataset, especially in language processing area is not an easy task.

Figures 3.3, 3.4, and 3.5 show the performance comparison of proposed SRC based LR approach employing GMM mean shifted supervector and i-vector utterance representation with OMP, Lasso and

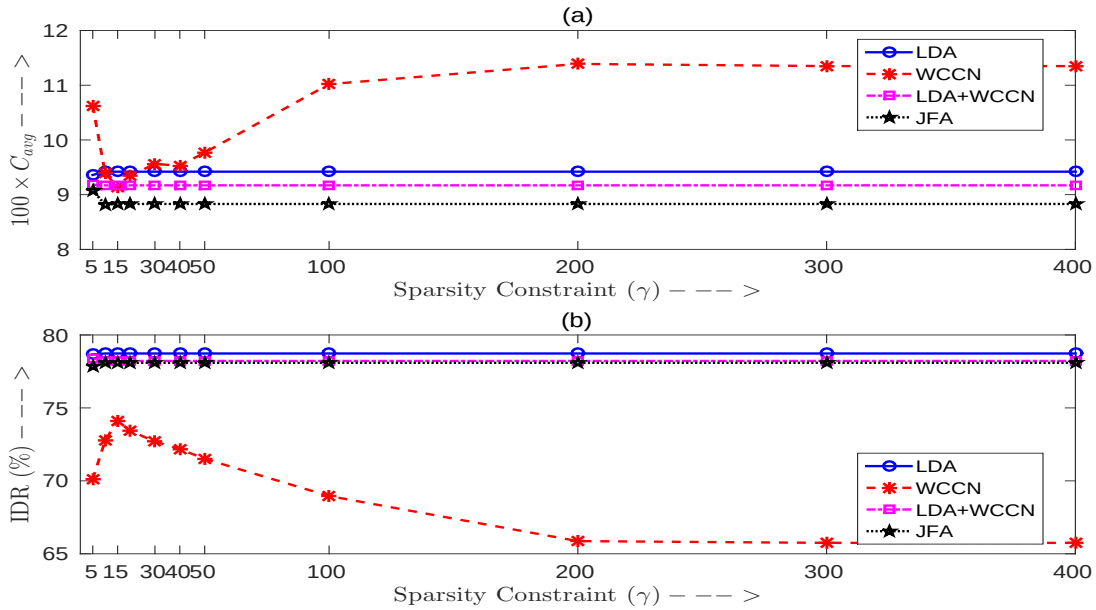

Figure 3.3: Effect of sparsity constraint (γ) in SRC based on i-vector/GMM supervector with appropriate session/channel compensation in terms of: (a) C_{avg} and (b) IDR. The sparse coding is performed using OMP algorithm.

ENet sparse coding algorithms, respectively. Two compensation techniques namely LDA and WCCN are applied to the i-vector system, while JFA has been used in GMM mean supervector system. In Figure 3.3, s-vectors corresponding to i-vectors/GMM-supervectors are computed using OMP algorithm over exemplar dictionary. It is observed that in OMP based SRC framework, the performance of LR system employing supervector utterance representation is much better compared to i-vector based representation. Furthermore, with fixed sparsity constraint $\gamma = 10$, the results are summarized in Table 3.3. We used an undercomplete dictionary in solving sparse classification problem in GMM mean shifted supervector approach unlike overcomplete dictionary in i-vector approach. But, still, the results obtained in GMM supervector approach with SRC are better than i-vector approach. A relative improvement of about 4% C_{avg} is observed with SRC based GMM mean shifted supervector with JFA than SRC based i-vector with LDA+WCCN. However, with Lasso and ENet based SRC framework, i-vector with WCCN compensation performs better than JFA-supervector, summarized in Table 3.3. Among the three sparse coding algorithms, the best performance is noted with WCCN compensated i-vector employing the ENet coding. It is also to be noted that the computational complexity requires to extract the s-vector in case of supervector is very high due to its high dimensionality. The optimal parameters of OMP, Lasso and ENet sparse coding algorithms are selected by proper tuning, as shown

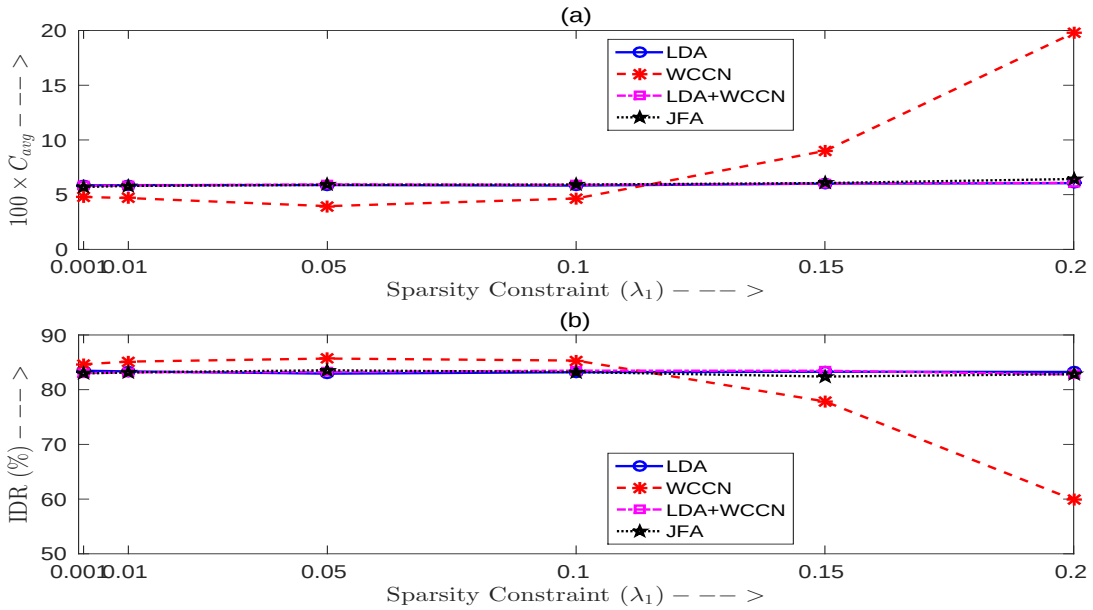

Figure 3.4: Effect of sparsity constraint (λ_1) in SRC based on i-vector/GMM supervector with appropriate session/channel compensation in terms of: (a) C_{avg} and (b) IDR. The sparse coding is performed using Lasso algorithm.

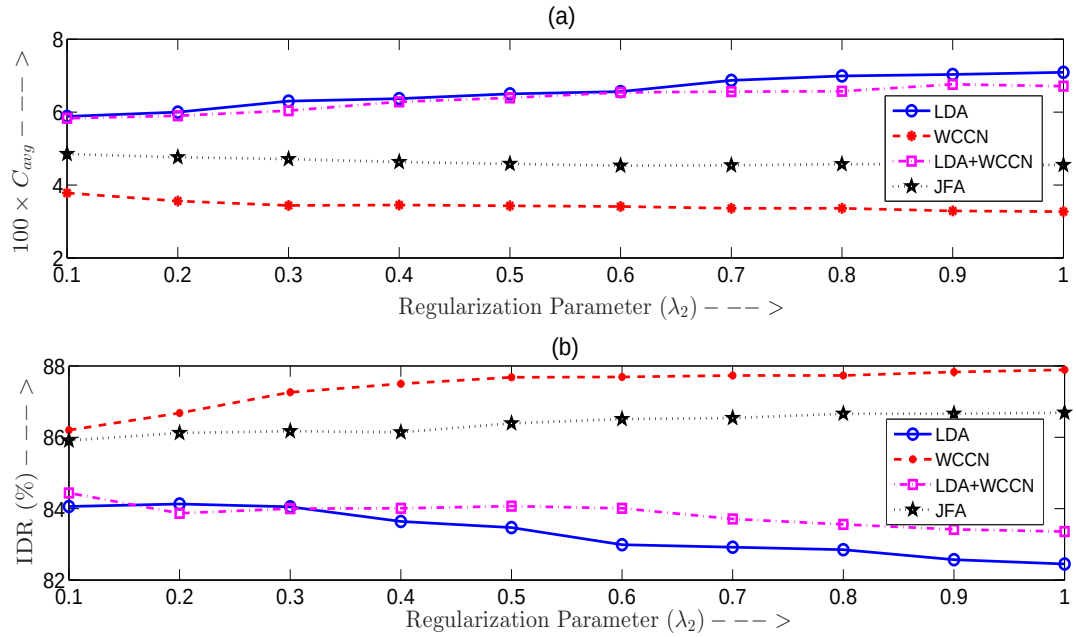

Figure 3.5: Effect of Regularization Parameter (λ_2) for fixed ($\lambda_1 = 0.05$) in SRC based on i-vector/GMM supervector with appropriate session/channel compensation in terms of: (a) C_{avg} and (b) IDR. The sparse coding is performed using ENet algorithm.

in Figures 3.3, 3.4, and 3.5, respectively.

Table 3.3: Performances of language recognition systems using exemplar dictionary with SRC classifier employing different regularization techniques (OMP, Lasso and ENet) on two utterance representations: i-vector and supervector. The regularization parameters are tuned to give best result. The sparsity constraint in OMP is chosen as $\gamma = 10$. In lasso, regularization parameter $\lambda_1 = 0.05$ is selected while regularization parameters $\lambda_1 = 0.05$ and $\lambda_2 = 0.1/1/1$ (in LDA/WCCN/JFA to obtain best result) are chosen in ENet.

Utterance Representation	Channel Compensation	Regularization	LD	LID
			$100 \times C_{avg}$	IDR (%)
ivector	LDA	OMP	9.42	78.73
		Lasso	5.88	82.93
		ENet ($\lambda_2 = 0.1$)	5.88	84.06
	WCCN	OMP	9.39	72.76
		Lasso	3.95	85.69
		ENet ($\lambda_2 = 1$)	**3.27**	**87.89**
	LDA+WCCN	OMP	9.17	78.22
		Lasso	5.94	83.23
		ENet ($\lambda_2 = 0.1$)	5.83	84.44
Supervector	JFA	OMP	8.82	78.08
		Lasso	5.89	83.54
		ENet ($\lambda_2 = 1$)	4.55	86.69

3.4.3 Language recognition using sparse representation over learned dictionary

In this subsection, LR using sparse representation over K-SVD/OL dictionary on two utterance representation: i-vector and supervector employing three sparse coding algorithms (OMP, Lasso, and ENet) are reported. For session/channel compensation, WCCN and JFA are applied on i-vector and GMM-mean supervector respectively. Both K-SVD and OL approaches use an iterative method that alternates between the sparse representation of the training example based on the current dictionary and the dictionary update stage. The sparse coding stage is used to find the sparse representation of the training data samples given the dictionary. This can be done by selecting any sparse coding algorithms like OMP, Lasso, and ENet. The dictionary update rule of the two dictionary learning algorithms differs, apart from the differences in the dictionary learning process. In the dictionary update stage of K-SVD, each atom is updated with the topmost singular vector obtained by singular value decomposition of the data samples associated with that atoms. However, in OL approach, dictionary update uses block coordinate descent with warm restarts. Once the dictionary is learned, s-vectors corresponding to training and test utterance representation is extracted over learned dictionary followed by CDS classifier. The resultant scores are class averaged and finally, scores are calibrated using GB+MLR. The size of dictionary and regularization parameters of sparse coding algorithms like OMP, Lasso and ENet is tuned to obtain the best system performance, and described in Appendix C.

Table 3.4: Performance of LR systems using sparse representation over learned dictionary (of size 28) with best tuning parameters values. In OL approach, the size of mini-batch is selected to be $b = 512$.

Utt. Representation +Channel Comp.	Dictionary/ Regularization	LD		LID	
		Parameter(s)	$100 \times C_{avg}$	Parameter(s)	IDR (%)
ivector+WCCN	K-SVD/OMP	$\gamma' = 20$, $\gamma = 15$	3.29	$\gamma' = 20$, $\gamma = 20$	**90.41**
	K-SVD/Lasso	$\lambda'_1 = 0.01$, $\lambda_1 = 0.001$	3.13	$\lambda'_1 = 0.02$, $\lambda_1 = 0.001$	89.94
	K-SVD/ENet	$\lambda'_1 = 0.01$, $\lambda'_2 = 0.4$, $\lambda_1 = 0.001$,$\lambda_2 = 0.2$	**3.07**	$\lambda'_1 = 0.01$,$\lambda'_2 = 1.2$, $\lambda_1 = 0.001$, $\lambda_2 = 0.2$	89.94
	OL/OMP	$\gamma' = 20$,$\gamma = 20$	3.52	$\gamma' = 20$,$\gamma = 20$	88.79
	OL/Lasso	$\lambda'_1 = 0.015$, $\lambda_1 = 0.001$	3.14	$\lambda'_1 = 0.005$, $\lambda_1 = 0.01$	89.84
	OL/ENet	$\lambda'_1 = 0.015$,$\lambda'_2 = 0.4$, $\lambda_1 = 0.001$,$\lambda_2 = 0.2$	**3.07**	$\lambda'_1 = 0.015$,$\lambda'_2 = 0.8$, $\lambda_1 = 0.001$, $\lambda_2 = 0.2$	**89.84**
supervector+JFA	K-SVD/OMP	$\gamma' = 20$,$\gamma = 15$	6.83	$\gamma' = 20$,$\gamma = 5$	80.86
	K-SVD/Lasso	$\lambda'_1 = 0.05$, $\lambda_1 = 0.001$	4.81	$\lambda'_1 = 0.05$, $\lambda_1 = 0.02$	85.80
	K-SVD/ENet	$\lambda'_1 = 0.05$,$\lambda'_2 = 1$, $\lambda_1 = 0.001$,$\lambda_2 = 0.6$	**3.95**	$\lambda'_1 = 0.05$,$\lambda'_2 = 2$, $\lambda_1 = 0.001$, $\lambda_2 = 0.6$	**88.31**
	OL/OMP	$\gamma' = 5$,$\gamma = 10$	6.42	$\gamma' = 5$,$\gamma = 5$	81.10
	OL/Lasso	$\lambda'_1 = 0.05$, $\lambda_1 = 0.02$	4.37	$\lambda'_1 = 0.05$, $\lambda_1 = 0.02$	85.99
	OL/ENet	$\lambda'_1 = 0.05$,$\lambda'_2 = 2$, $\lambda_1 = 0.02$,$\lambda_2 = 0.2$	**3.74**	$\lambda'_1 = 0.05$,$\lambda'_2 = 0.8$, $\lambda_1 = 0.02$, $\lambda_2 = 1.8$	**88.37**

The performance of proposed K-SVD/OL dictionary employing different regularization techniques on WCCN compensated i-vector and JFA-supervector utterance representation with best tuning parameters are summarized in Table 3.4. It is observed that the performance of i-vector based LR system is much better than the JFA-supervector. Among OMP, Lasso and ENet based regularization, the ENet based sparse coding gives satisfactory performance. The performances of two dictionary learning approaches, the K-SVD and the OL dictionary learning approaches are comparable. It is also observed that the learned dictionary based LR systems performs much better than an exemplar dictionary based LR. On comparing with i-vector employing GG classifier, the equal performance is noted.

3.4.4 Language recognition using sparse representation over class-based learned dictionary

In this subsection, the performance of LR systems using sparse representation over class-based learned dictionary are summarized in Table 3.5. Two standard dictionary learning method namely K-SVD and OL are used for creating the class-wise dictionaries from language-specific training data. Finally, sparse coding is done over a composite dictionary created by concatenation of class-wise

Table 3.5: Performance of LR systems using sparse representation over class-based learned dictionary with best tuned parameter values. The K-SVD and OL learned dictionary of size 400×28 employing different regularization techniques (OMP, Lasso, and ENet) on WCCN session/channel compensated i-vector are considered.

Dictionary/ Regularization	LD		LID	
	Parameter(s)	$100 \times C_{avg}$	Parameter(s)	IDR (%)
K-SVD/OMP	$\gamma' = 1, \gamma = 9$	3.06	$\gamma' = 1, \gamma = 13$	**90.05**
K-SVD/Lasso	$\lambda'_1 = 0.04$, $\lambda_1 = 0.005$	3.01	$\lambda'_1 = 0.04$, $\lambda_1 = 0.005$	89.84
K-SVD/ENet	$\lambda'_1 = 0.04, \lambda'_2 = 0$, $\lambda_1 = 0.005, \lambda_2 = 1$	**3.00**	$\lambda'_1 = 0.04, \lambda'_2 = 0.1$, $\lambda_1 = 0.005, \lambda_2 = 0.9$	89.83
OL/OMP	$\gamma' = 1, \gamma = 14$	**3.02**	$\gamma' = 1, \gamma = 14$	**90.04**
OL/Lasso	$\lambda'_1 = 0.06$, $\lambda_1 = 0.01$	3.16	$\lambda'_1 = 0.06$, $\lambda_1 = 0.01$	89.32
OL/ENet	$\lambda'_1 = 0.06, \lambda'_2 = 1.5$, $\lambda_1 = 0.01, \lambda_2 = 0$	3.06	$\lambda'_1 = 0.06, \lambda'_2 = 0.9$, $\lambda_1 = 0.01, \lambda_2 = 1.7$	90.02

dictionaries. The size of the composite dictionary is selected to be 24 considering 2 dictionary atoms per class. Apart from the choice of the dictionary, we have also compared the LR performances considering OMP, Lasso, and ENet based sparse coding regularization techniques. The tuning parameters are varied over a wide range and the best parameters are selected based on the best LR performance.

The dictionary sparsity γ' and decomposition sparsity γ in class-based learned dictionary employing OMP regularization is varied in the range of 1-2 and 1-28 respectively. With Lasso, the dictionary sparsity λ'_1 is varied from 0.01-0.1 with increment of 0.01 while decomposition sparsity λ_1 varies between 0.005-0.02 with increment of 0.005. Using K-SVD employing Lasso, the best performance is obtained with the value of $\lambda'_1 = 0.04$ and $\lambda_1 = 0.005$. However, using OL employing Lasso, $\lambda'_1 = 0.06$ and $\lambda_1 = 0.01$ gives the best performance compared with other values. In ENet, the dictionary and decomposition sparsity are selected based on the best performance of Lasso and kept fixed. The other two regularization coefficients λ'_2 and λ_2 are varied in the range of 0-2 with an increment of 0.1. It is observed that the performance of class-based learned dictionary approach performs significantly better than that of the learned dictionary. The improved performance is due to the discriminative nature of class-based learned dictionary.

3.4.5 D-KSVD and LC-KSVD learned dictionary based language recognition

In this subsection, we have explored two discriminative learned dictionaries, the D-KSVD, and LC-KSVD specially designed for classification purpose. The results are summarized in Table 3.6 for the

Table 3.6: Performance of LR systems using D-KSVD and LC-KSVD learned dictionary with best tuned parameter values. The K-SVD learned dictionary of size 400×28 employing different regularization techniques (OMP, Lasso, and ENet) on WCCN session/channel compensated i-vector are considered.

Dictionary/ Regularization	LD		LID	
	Parameter(s).	$100 \times C_{avg}$	Parameter(s)	IDR (%)
D-KSVD/OMP	$\gamma' = 1$, $\gamma = 22, \sqrt{\beta} = 0.3$	**2.93**	$\gamma' = 10, \gamma = 23, \sqrt{\beta} = 0.2$	**90.67**
D-KSVD/Lasso	$\lambda'_1 = 0.05$, $\lambda_1 = 0.01, \sqrt{\beta} = 0.2$	2.92	$\lambda'_1 = 0.1$, $\lambda_1 = 0.01, \sqrt{\beta} = 0.4$	90.59
D-KSVD/ENet	$\lambda'_1 = 0.05, \lambda'_2 = 0.1, \lambda_1 = 0.01,$ $\lambda_2 = 0.1, \sqrt{\beta} = 0.1$	2.99	$\lambda'_1 = 0.1, \lambda'_2 = 0.9, \lambda_1 = 0.01,$ $\lambda_2 = 0.8, \sqrt{\beta} = 0.1$	90.13
LC-KSVD1/OMP	$\gamma' = 6$, $\gamma = 22, \sqrt{\alpha} = 0.1$	**2.76**	$\gamma' = 1, \gamma = 21, \sqrt{\alpha} = 0.1$	90.78
LC-KSVD1/Lasso	$\lambda'_1 = 0.05$, $\lambda_1 = 0.001, \sqrt{\alpha} = 0.1$	2.93	$\lambda'_1 = 0.005$, $\lambda_1 = 0.01, \sqrt{\alpha} = 0.1$	90.46
LC-KSVD1/ENet	$\lambda'_1 = 0.05, \lambda'_2 = 0.2, \lambda_1 = 0.001,$ $\lambda_2 = 0.4, \sqrt{\alpha} = 0.1$	2.97	$\lambda'_1 = 0.005, \lambda'_2 = 0, \lambda_1 = 0.01,$ $\lambda_2 = 1, \sqrt{\alpha} = 0.2$	**91.01**
LC-KSVD2/OMP	$\gamma' = 6$, $\gamma = 21,$ $\sqrt{\alpha} = 0.1, \sqrt{\beta} = 0.2$	**2.71**	$\gamma' = 1$, $\gamma = 23,$ $\sqrt{\alpha} = 0.1, \sqrt{\beta} = 0.2$	90.68
LC-KSVD2/Lasso	$\lambda'_1 = 0.05, \lambda_1 = 0.001,$ $\sqrt{\alpha} = 0.1, \sqrt{\beta} = 0.1$	2.86	$\lambda'_1 = 0.03, \lambda_1 = 0.015,$ $\sqrt{\alpha} = 0.1, \sqrt{\beta} = 0.2$	90.57
LC-KSVD2/ENet	$\lambda'_1 = 0.05, \lambda'_2 = 0$, $\lambda_1 = 0.001,$ $\lambda_2 = 0, \sqrt{\alpha} = 0.1, \sqrt{\beta} = 0.1$	2.91	$\lambda'_1 = 0.03, \lambda'_2 = 0.9, \lambda_1 = 0.015,$ $\lambda_2 = 0.9, \sqrt{\alpha} = 0.2, \sqrt{\beta} = 0.1$	**91.05**

best tuning parameters. On comparing with D-KSVD and LC-KSVD, a slight improvement in the LR performance is noted in LC-KSVD. The improved results are due to the incorporation of discriminative sparse-code error term enforcing label consistency in addition to the terms (i.e., reconstruction and classification error term) used in the D-KSVD formulation. In LID, the ENet based sparse coding is found to give satisfactory result compared to OMP and Lasso. However, the OMP based LR system is slightly better than the LR system employing Lasso and ENet based regularization. It is to be noted that the performance of LC-KSVD based LR system is superior among D-KSVD, class-wise learned dictionary, simple learned dictionary, an exemplar dictionary based LR systems. A large improvement is also observed compared to the i-vector LR systems employing GG classifier.

3.5 Summary

In this chapter, we have first analyzed the effect of three sessions/channel compensations on i-vector utterance representation based LR employing CDS classifier. For score calibration, GB, MLR, and GB+MLR have been investigated. The experimental results show the superior performance with WCCN compensation and GB+MLR calibration. Further, various classifiers (SVM, G-PLDA, MLR, GG, and CDS) on i-vector have been explored. Among the classifiers, the best performance is noted

with WCCN compensated i-vector with GG classifier.

The SRC based LR system have been also explored where an exemplar dictionary has been created with i-vector/GMM-mean supervector utterance representation with appropriate compensation technique. The sparse coding is performed over an exemplar dictionary employing three regularization techniques (OMP, Lasso, and ENet). The best performance has been noted with WCCN compensated i-vector using ENet based sparse coding. On comparing with the i-vector employing GG classifier, a slightly degraded performance has been noted in SRC based LR.

With the increase of a number of training examples (e.g., i-vector and GMM-mean supervector), the size of the exemplar dictionary increases and the computational burden of sparse coding over such dictionary becomes quite prohibitive. To address that, various sparse representation techniques over simple learned dictionary have been also explored for LR task. For creating the learned dictionary, both K-SVD and OL dictionary learning approaches employing different kinds of regularization have been investigated. Once the dictionary is learned, s-vectors corresponding to the training and test utterances have been extracted, and scores have been computed with the help of CDS classifier. It is to be noted that the class-labels in the case of the simple learned dictionary are not retained and hence the SRC is not feasible. To address this problem, the class-based learned dictionary has been also explored. Here, the language-specific learned dictionaries are concatenated to form the learned-exemplar dictionary. The performance of learned-exemplar dictionary based LR is much better compared to the simple learned dictionary, and this may be due to the discriminative nature of s-vector computed using learned-exemplar.

The D-KSVD and the LC-KSVD learned discriminative dictionaries specially designed for classification purpose have been also investigated. On comparing with D-KSVD and LC-KSVD, a slight improvement in the LR performance is noted in LC-KSVD. The improved results are due to the incorporation of discriminative sparse-code error term enforcing label consistency in addition to the reconstruction and classification error terms used in the D-KSVD formulation.

On comparing with two utterance representations, the GMM mean supervector and the i-vector, it is observed that the performance of the proposed language recognition approaches are much better with the i-vector employing suitable channel compensations. The major drawback of the i-vector approach lies in its computational complexity and storage requirements. Motivated by these constraints, in the next chapter ensemble of random subspaces of supervector based LR approach is explored.

4

Low Complexity Language Recognition Exploiting Ensemble of Random Subspace

Contents

4.1 Motivation

The current approaches for spoken LR are predominantly based on GMM mean supervector or its variants as the representation of the utterances. It is assumed that the language information lies in a linear manifold of low dimensional spaces. Exploiting that low dimensional projections of the GMM mean supervectors, known as *i-vectors*, are derived using a total variability matrix. The i-vector representation followed by LDA/WCCN based session/channel compensation forms the state-of-the-art. The major drawback of the i-vector approach lies in its computational complexity and storage requirements. Let m, p, and q represents the number of Gaussian mixture components, feature dimension and the size of i-vector, respectively. Using Equation 2.13, the computational complexity and memory requirement involved in i-vector estimation for an utterance is $\mathcal{O}(mpq + mq^2 + q^3)$ and $\mathcal{O}(mpq + mq^2)$, respectively. For example, the computational complexity and memory requirement in i-vector extraction for the typical value of parameters $m = 1024$, $p = 39$, and $q = 400$ are in tune of 243,814,400 and 179,814,400 bits ($\approx$ 22.477 MB), respectively. Motivated by these constraints, in this chapter we explore ensemble of random subspaces of supervector based LR approach which includes LDA/WCCN based session/channel compensation in the subspace. The proposed approach does not require any learning for creating the subspaces and has very low storage requirements. In typical LR applications, the number of languages required to be identified is limited to 10-30. As a result of that, the complexity of the JFA based system turns out to be lower than that of the i-vector based one. Exploiting this fact, the combination of the proposed ensemble approach with JFA is also explored.

4.2 Introduction

In recent times, the JFA [75] and i-vector [78] based statistical models are found to be very effective in speaker recognition [75, 78] and language recognition [79, 80, 88] tasks. The idea of JFA is to model the speaker and session/channel spaces separately. The i-vector, an extension of the JFA approach, is motivated for addressing the loss of some relevant speaker information during modeling of speaker and session/channel subspaces [78]. In the i-vector approach, the dominant variability in high dimensional supervector are first captured into a relaxed lower dimensional space and then appropriate measures like LDA and WCCN [145] are employed for compensation of session/channel variability. Recently, some works have been reported towards addressing the issues of complexity and storage requirements in the i-vector computation [93–95]. In addition to these efforts, there also exists

an ensemble learning framework based on random sampling of PCA subspace in face recognition [185] and speaker verification [186–188]. The idea was to employ LDA or Fishervoice classifiers in the randomly sampled PCA subspace of high dimensional vector representation employed. In practice, the usage of PCA gets restricted due to the high complexity involved, whereas with random sampling the chance of losing some useful information is always there. It has also been shown that the constructed LDA classifier is often biased and unstable when the PCA subspace dimension is relatively high [185].

In this work, we propose a sequential partitioning of randomly permuted supervector to derive low dimensional subvectors (slices). The subspaces created in this manner are expected to be more balanced than the sequential partitioning of the supervector without randomization. Note that in the proposed approach the information in supervector simply gets distributed into the subvectors without any loss unlike that in the random sampling approach. Also, the proposed approach does not involve any learning for the creation of subspace unlike the PCA or i-vector approach. The LDA followed by WCCN channel compensation is then applied on low dimensional random subspace to mitigate the session/channel variability. The compensated, reduced dimensional subvectors are concatenated to form a single vector and the CDS is used to find the similarity score.

It is to note that though there are a large number of languages being spoken over the world, the current publicly available language databases cover only a small set of languages. As a result of this, the language-specific information in JFA can be captured without restricting the language space. Thus, we argue that in LR task, there will be an insignificant loss of language-specific information with JFA modeling unlike that reported for a speaker verification task. Attributed to a small number of language factors involved, the complexity of JFA based LR system happens to be quite low compared to the i-vector based one. Exploiting this fact, we have also explored the combination of the proposed ensemble approach with JFA.

4.3 Proposed ensemble of subspaces based language recognition

The main motivation of the work reported in this chapter lies in the development of an LR system having low computational complexity and storage requirements. To achieve the same, we have explored an ensemble of subvectors based LR approach. It is assumed that the GMM mean shifted supervector based representation of the utterances is available. Each of the given supervectors is then sliced into subvectors of suitable length. Prior to slicing, the supervectors can also be randomly permuted. For

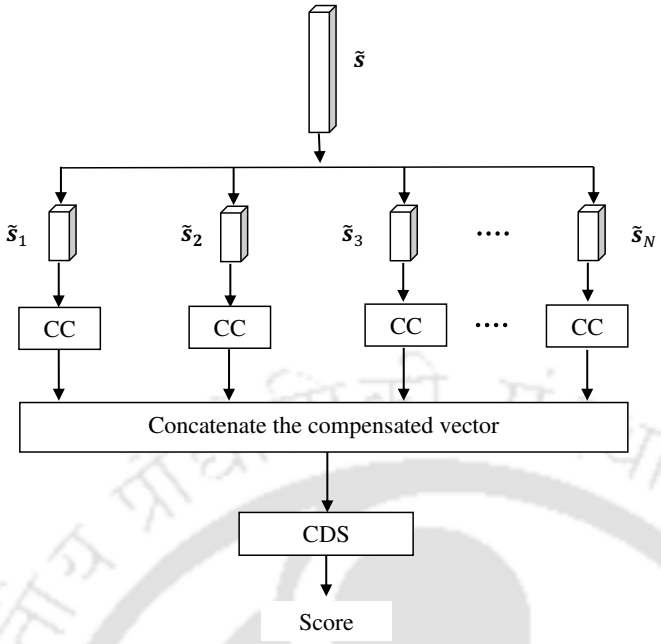

Figure 4.1: Flow diagram of the proposed ensemble of subspaces based LR system ($\tilde{\boldsymbol{s}}$: centered GMM mean supervector, $\tilde{\boldsymbol{s}}_N$: N^{th} partitioned subvector, CC: session/channel compensation using LDA/WCCN transformation, CDS: cosine distance scoring).

session/channel compensation, the linear transformations like LDA/WCCN are applied to the obtained subvectors. The compensated subvectors are then concatenated to form a single vector. The cosine kernel scoring has been used for recognition purpose. Figure 4.1 outlines the proposed ensemble of subspaces based LR system. The algorithm of the proposed LR approach and a brief description of the partitioning stage are presented in the following subsections.

4.3.1 Algorithm

The steps in the proposed ensemble based LR approach:

(i) Given the language independent GMM-UBM, generate the zeroth- and the first-order statistics for the utterance.

(ii) Using the statistics, create a GMM mean supervector and remove the bias from it by subtracting the UBM mean supervector.

(iii) Randomly permute the derived GMM mean shifted supervector $\tilde{\boldsymbol{s}}$, if required.

(iv) Partition the supervector $\tilde{\boldsymbol{s}} \in \mathbb{R}^M$ into N subvectors. The derived subvectors $\tilde{\boldsymbol{s}}_i$, $i = 1, 2, \ldots, N$ are of length K each, such that $M = K \times N$.

(v) Apply suitable session/channel compensation on each of the subvectors $\tilde{\boldsymbol{s}}_i$. Let $\tilde{\boldsymbol{s}}_{c_i} \in \mathbb{R}^P$ denote the corresponding compensated subvectors of length $P \leq K$.

(vi) Concatenate session/channel compensated subvectors $\{\tilde{\boldsymbol{s}}_{c_i}\}_{i=1}^{N}$ to form a single vector $\mathbf{x}$ of length $P \times N$.

(vii) Given the training vector $\mathbf{x}_{\text{trn}}$ and the test vector $\mathbf{x}_{\text{tst}}$, compute the cosine distance score as

$$\text{Score} = \frac{\mathbf{x}'_{\text{trn}} \cdot \mathbf{x}_{\text{tst}}}{\|\mathbf{x}_{\text{trn}}\| \ \|\mathbf{x}_{\text{tst}}\|} \tag{4.1}$$

where $\mathbf{x}'_{\text{trn}}$ is the transpose of $\mathbf{x}_{\text{trn}}$.

(viii) Repeat the step (vii) for each of the training vector keeping fixed test vector. Finally, the language specific scores are obtained by computing the mean of scores belonging to particular language class.

4.3.2 Partitioning of supervector

Consider a GMM supervector $\grave{\boldsymbol{s}}$ of length CD, where C is the number of Gaussian mixtures and D be the dimensionality of the feature vector. The mean supervector is created by stacking the mean vectors of MAP adapted GMM-UBM in row-wise fashion, such that the first D entries correspond to 1^{st} mixture and so on up till C^{th} mixture. Furthermore, in order to reduce the bias due to the UBM, the language-independent UBM mean supervector $\boldsymbol{\mu}$ is subtracted from the GMM supervector $\grave{\boldsymbol{s}}$, thus creating the centered GMM mean supervector $\tilde{\boldsymbol{s}}$.

For slicing the GMM mean shifted supervector, we can follow two schemes as described below:

- **Sequential slicing**: The given supervector is partitioned sequentially to create the slices of appropriate length. As the entries in the supervector are structured, the resulting slices would exhibit some finite structures depending on the chosen length of the slices.
- **Sequential slicing of randomized supervector**: The supervector elements are first permuted randomly prior to partitioning sequentially to create the slices. The randomization of supervector helps in reducing any definite structure embedded in the resulting slices.

Out of these schemes, the sequential slicing of randomized supervector would result in more diverse feature selection in the subvectors and hence is expected to outperform the other. These two partitioning methods are illustrated in Figure 4.2.

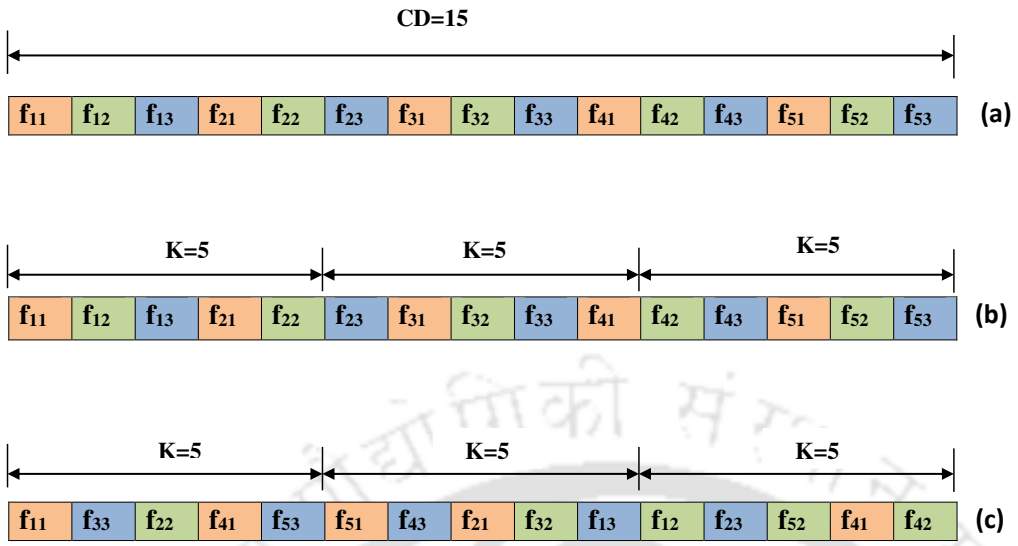

Figure 4.2: Illustration of two different ways of partitioning of the supervector. (a) Given 15-dimensional GMM mean supervector corresponding to GMM-UBM size $C = 5$ and feature vector dimension $D = 3$ which is to be partitioned into slices of lenght $K = 5$, (b) Sequential slicing, (c) Sequential slicing of randomly permuted supervector.

4.4 Experimental setup

The performance of all the developed systems is evaluated on NIST 2007 LRE datasets [154]. The evaluation dataset contains 14 different languages and consists of 7530 spoken utterances of 3 seconds, 10 seconds and 30 seconds durations including the out-of-set data. We have focused on the closed-set task and do not include any out-of-set data. Each test utterance is of 30 seconds duration. Since we did not have sufficient training as well as development data available in German, Farsi and Tamil languages, the results are reported for 1837 test utterances excluding utterances from these three languages. The development and training datasets consist of 2493 and 550 segments, respectively. These datasets are collected from conversational telephone speech in 11 languages. The development data set includes the speech data extracted from multiple corpora: OGI-multilingual, mixer (data from NIST 2004, 2005, 2006, 2008 SREs) and previous NIST 1996, 2003, 2005 LREs. The training dataset contains NIST 2007 LRE supplementary training data and some data from SRE databases which are not included in the development dataset. The development and training data sets are separately pooled language-wise in feature space after the removal of silence segments. An average duration of 10 minutes is kept for each of the segments in development and training datasets.

For the experiments, all speech data used is sampled at 8 kHz with 16 bits/sample resolution. Prior to feature computation, the speech signals are pre-emphasized using a factor of 0.97. Pre-emphasis

filter is used to enhance the higher frequencies. Its purpose is to balance the spectrum of voiced sounds that have a steep roll-off in the high frequency region. The most commonly used pre-emphasis filter is given by the following transfer function $H(z) = 1 - \alpha z^{-1}$, where the value of α controls the spectral slope and is usually set between 0.92 to 1.0.The speech data is short-time processed using a Hamming window of 20 ms duration and keeping a frame rate of 100 Hz. Each speech frame is spectrally analyzed using 23-channel triangular Mel-filter bank and converted to MFCCs [9]. The first 13 coefficients are used as a feature vector excluding zeroth coefficient value. The first and second order derivatives of MFCC feature vectors are calculated using two preceding and two successive feature frames from the current frame [125]. Thus, a total of 39-dimensional MFCC feature vectors are extracted for each frame. The CMS is applied on MFCC feature vectors to reduce the channel and environmental effects.

A language-independent GMM-UBM having 1024 Gaussian components is employed to build the LR system using 11 hours of development data created by pooling approximately 1 hour of data from each of the 11 languages being modeled. In JFA modeling, the language and channel factors are kept as $LF = 11$ and $CF = 200$, respectively. Whereas, for the i-vector extraction, the rank of the total variability matrix is kept to be 400 as suggested in the literature. The LDA and WCCN channel compensation techniques are used to mitigate the session/channel effects.

4.5 Results and discussion

The performances of the LD systems reported in this chapter are evaluated in terms of the average cost detection function (C_{avg}) as defined in NIST 2007 LRE plan [154]. To measure the LID performance, the average identification rate (IDR) is computed. Figure 4.3 shows the performances of the proposed LR system with two supervector partitioning schemes for varying subvector lengths. These performances also include LDA followed by WCCN session/channel compensation being applied in the subspaces. On comparing the two supervector slicing schemes, the randomized slicing found to give better performance in comparison to sequential slicing. On comparing with the sequential slicing, the relative improvements of 12.25% in C_{avg} and 4.11% in IDR are noted. The best performances of the proposed ensemble based LR system are achieved with a slice length of 1024.

The effect of session/channel compensation techniques with sequential slicing of supervector and sequential slicing of randomized supervector schemes are summarized in Table 4.1 and Table 4.2, respectively. The improved performances with the randomized slicing of supervector are attributed to

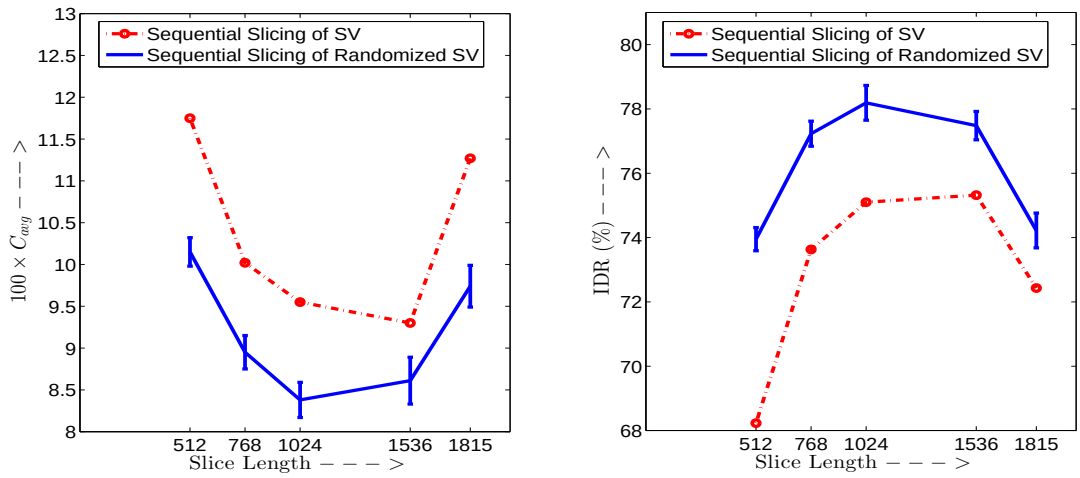

Figure 4.3: Tuning of the slice length for two schemes of partitioning of the supervector. Performances of the proposed LR system with LDA followed by WCCN based session/channel compensation are evaluated in terms of C_{avg} and IDR. In sequential slicing of randomized supervector, the standard deviation of the performances for 10 trials are represented by a vertical bar.

more diverse features being captured in the subvectors in contrast to the sequential slicing. In the following subsection, we explore the combining of both JFA and LDA/WCCN based session/channel compensation of achieving improved LR system performance with the proposed ensemble-based approach.

4.5.1 Combination of JFA with the ensemble-based approach

From Table 4.3, we note that the proposed ensemble approach based LR system results in a relative improvement of 13.34% in C_{avg} and 1.38% in IDR when compared to JFA based system, but its performances are quite inferior in contrast to that of the i-vector based system.

We wish to highlight that some degree of session/channel compensation is inherently achieved in low-rank (i-vector) modeling. This inherent session/channel compensation is achieved in higher dimensional space and is found to be complementary to those achieved by LDA/WCCN in lower dimensional space. This is the possible reason behind the i-vector approach outperforming the JFA one in many tasks. In LR tasks, the number of languages involved yet are limited, and hence the language space. The maximum number of language factors required to capture whole language space depends upon the number of language classes. In our case, a total number of language and channel factors used in JFA modeling is 211 only while the i-vector approach uses 400 total factors. Thus, the computation requirements in JFA processing turn out to be order lower than that of the i-vector computation.

Table 4.1: Slice length-wise performances of the proposed ensemble based LR systems on the GMM supervector for various session/channel compensation methods. The results are shown with sequential slicing of supervector.

Slice Length	Session/Channel Compensation Techniques	LD	LID
		$100 \times C_{avg}$	IDR (%)
39,936	No compensation	30.46	31.05
512	WCCN	11.85	69.20
	LDA	12.88	67.37
	LDA+WCCN	11.75	68.23
768	WCCN	10.20	73.63
	LDA	10.70	71.57
	LDA+WCCN	10.02	73.63
1024	WCCN	9.63	74.71
	LDA	9.80	73.93
	LDA+WCCN	9.55	75.10
1536	WCCN	10.83	71.29
	LDA	9.78	74.47
	LDA+WCCN	**9.30**	**75.32**
1815	WCCN	13.85	62.99
	LDA	11.01	72.96
	LDA+WCCN	11.27	72.43

Motivated by that, we also explored the combination of JFA based session/channel compensation with our proposed approach. For combining the two, we could simply make use of the JFA pre-processed supervectors in the proposed ensemble approach. In this way, we can exploit the session/channel compensation both in high and low dimensional spaces. Note that the JFA compensated supervectors cannot be modeled with the higher rank model than used in JFA processing for further session/channel compensation. As a result of that, we partition the JFA compensated supervector according to the size of the language factors chosen in the modeling.

The combination of JFA and the proposed ensemble-based approach considering sequential slicing of randomized supervector followed by WCCN as session/channel compensation results in a relative improvement of about 13% in C_{avg} and about 2% in IDR with respect to the i-vector based LR system. However, the combined approach with LDA as session/channel compensation is noted to give a relative degradation of 26.08% in C_{avg} and about 2% in IDR in comparison to the WCCN case. This may be due to low language factors used in JFA processing which are equal to the number of languages in the task. Therefore, in the case of JFA pre-processed supervector the language information

Table 4.2: Slice length-wise performances of the proposed ensemble based LR systems on the randomly permuted GMM supervector for various session/channel compensation methods. These results show that the slice length of 1024 has turn out to be optimal across all combination of session/channel compensation methods. The mean(μ) and standard deviation (σ) of the performances computed over 10 iterations are shown.

Slice Length	Session/Channel Compensation Techniques	LD		LID	
		$100 \times C_{avg}$		IDR (%)	
		μ	σ	μ	σ
512	WCCN	10.10	0.13	73.06	0.29
	LDA	10.54	0.18	72.57	0.30
	LDA+WCCN	10.15	0.17	73.95	0.36
768	WCCN	8.86	0.22	76.39	0.51
	LDA	9.32	0.19	76.52	0.44
	LDA+WCCN	8.95	0.20	77.23	0.39
1024	WCCN	8.54	0.26	77.20	0.84
	LDA	8.65	0.18	77.85	0.42
	LDA+WCCN	**8.38**	0.21	**78.19**	0.54
1536	WCCN	10.25	0.25	72.92	0.91
	LDA	8.66	0.26	77.39	0.45
	LDA+WCCN	8.61	0.28	77.48	0.44
1815	WCCN	12.59	0.23	66.39	1.24
	LDA	9.78	0.32	74.29	0.58
	LDA+WCCN	9.74	0.25	74.22	0.54

forced to reside in a very low subspace. Thus, the subsequent LDA processing of subvectors derived from the JFA compensated supervector does not turn out to be as effective as noted in the case of uncompensated supervectors. Even with using both LDA and WCCN in the combined approach, the resulting performances in language detection remain lower than those obtained for WCCN case. However, a marginal performance gain is observed in case of language identification.

4.5.2 Computational complexity

The complexity of the proposed ensemble based approach is obviously far less than that of the i-vector based approach since it does not involve any pre-learned projection matrix or matrix manipulations. Whereas, when the proposed ensemble approach is combined with JFA for boosting the performance, the overall complexity gets enhanced substantially. As pointed out earlier, the complexity of JFA turns out to be lower than that the i-vector extraction for limited number of languages being involved. As a result of that, the overall complexity remains on the lower side even after combining the proposed ensemble based approach with JFA. The order of multiplication complexity of various

Table 4.3: LR performances of the proposed ensemble based approach with/without JFA preprocessing (language factors, LF = 11; channel factors, CF = 200) averaged over 10 iterations along with those of the contrast systems.

LR Systems	Session/Channel Compensation Techniques	LD $100 \times C_{avg}$	LID IDR (%)
Supervector based	None	30.46	31.05
	JFA	9.67	77.12
i-vector based	None	27.96	37.81
	WCCN	6.42	82.45
	LDA	6.21	84.66
	LDA+WCCN	6.20	84.66
Proposed ensemble based (slice length=1024)	LDA+WCCN	8.38	78.19
JFA+proposed ensemble based (slice length=11)	LDA	7.01	82.70
	WCCN	**5.56**	84.07
	LDA+WCCN	6.15	**84.81**

Table 4.4: Stage-wise multiplication complexity, involved memory and runtime of the i-vector based LR system. (Size of GMM-UBM, $m = 1024$; MFCC feature dimension, $p = 39$; size of i-vector, $q = 400$; and size of LDA+WCCN projected vector $s = 10$)

Parameters	i-vector extraction	LDA+WCCN projection	CDS	Total
Multiplication complexity	$\mathcal{O}(mpq + mq^2 + q^3)$	$\mathcal{O}(qs)$	$\mathcal{O}(s)$	-
Runtime (ms) †	1551.00	0.13	-	1551.13
Memory usage (in MB)	1284.40	-	-	1284.40

† Computed using 32-bit MATLAB (R2012a) running on Intel®Core™i3-2130 CPU, 3.4 GHz with 4 GB RAM in single thread default precision.

stages in the i-vector based and the proposed ensemble based LR systems are given in Table 4.4 and Table 4.5, respectively. For the chosen parameter values in this work, the runtime and the involved memory are computed for both the systems and also listed in the corresponding tables.

In literature, there do exist some simplified i-vector extraction methods [93–95]. In these methods, by exploiting eigen-decomposition or subspaces or lookup tables, manyfold reduction in the complexity in the i-vector extraction is achieved over the default i-vector extraction method. In this work, we have used the default i-vector extraction for performance evaluation and complexity comparison. The proposed ensemble approach with JFA would certainly be found more complex if one compares it with the simplified i-vector based approach. But the techniques employed for reducing the i-vector extraction complexity can be extended for JFA case too. Thus even with the use of simplified factor analysis, the relative complexity advantage of the proposed approach over the i-vector based approach would remain intact.

Table 4.5: Stage-wise multiplication complexity, involved memory and runtime of the proposed ensemble based LR system with JFA preprocessing. (Size of GMM-UBM, $m = 1024$; MFCC feature dimension, $p = 39$; size of total loading factors, $k = 211$; size of language factors, $LF = 11$; size of subvectors, $b = 11$; number of subvectors, $u = \lfloor \frac{mp}{b} \rfloor = 3630$; and size of LDA+WCCN projected vector $s = 10$)

Parameters	Language factor extraction	Language factor to supervector	LDA+WCCN on subvectors	CDS	Total
Multiplication complexity	$\mathcal{O}(mpk+ mk^2 + k^3)$	$\mathcal{O}(mp(LF))$	$\mathcal{O}(bsu)$	$\mathcal{O}(su)$	-
Runtime (ms) †	868.00	0.32	-	-	868.32
Memory (in MB)	359.17	0.31	0.37	-	359.85

† Computed using 32-bit MATLAB (R2012a) running on Intel®Core™i3-2130 CPU, 3.4 GHz with 4 GB RAM in single thread default precision.

4.6 Summary

In this work, a low complexity LR approach has been proposed in the context of GMM mean supervector based representation of the spoken utterances. The proposed approach employs the ensemble of subspaces for LR and these subspaces are created using partitioning of the randomized supervector. The proposed approach, with LDA/WCCN session/channel compensation in subspaces, is attributed to have very low complexity and memory requirements in contrast to the concurrent LR systems based on JFA or i-vector. On NIST 2007 LRE datasets, the proposed ensemble based LR system is noted to results in relative improvements of 13.34% and 1.38% in C_{avg} and IDR respectively, over the JFA based LR system. However, when compared with the i-vector based system the proposed approach is found to be quite degraded.

Typically, the number of languages involved in LR tasks are limited, and hence the number of language factors. Thus, the complexity involved in extracting the MAP point estimate corresponding to the eigenvoice matrix in JFA processing turns out to be much smaller than that in the i-vector extraction. Using this fact, we also explored the combination of JFA with the proposed ensemble-based approach and the resulting system is noted to outperform the i-vector based system. This noted enhancement is attributed to more effective addressal of session/channel variability both in high and low dimensional spaces with the use of JFA and LDA/WCCN. It has been shown that the overall complexity and runtime of the combined system remains lower than that of the i-vector based system. Motivated by the reduced complexity, we have also used the JFA-latent vectors for creating ensemble dictionaries in Chapter 5.

5

Ensemble of Discriminative Learned Dictionaries for Language Recognition

Contents

5.1 Motivation

Computing sparse representation over an exemplar dictionary is time-consuming and computationally expensive for large dictionary size. In order to reduce the latency and finding efficient sparse representation, a single learned and discriminatively learned dictionaries are presented in Chapter 3. An alternative approach to do that is based on an ensemble of small-sized exemplar dictionaries. We hypothesize that such an ensemble of dictionaries is expected to provide the diversity gain. The different orthogonal subsets of the training data are being created by taking a fixed number of utterances from the data pool for each of the languages. Recently, an ensemble of exemplar dictionary based LR approach [100] has been reported. Unlike that, the approach presented in this chapter doesn't require any random sampling and hence there is no chance of getting same feature vector in different dictionaries (i.e., the atoms across the dictionaries are unique). The number of such exemplar dictionaries would become very high if the size of the training dataset is large. On account that many sparse codings of the test feature vector, the computational burden of the proposed approach becomes prohibitive. In order to mitigate this problem, we further explore the ensemble of learned-exemplar discriminative dictionary based LR system. The score obtained from each of the dictionaries in the ensemble is first calibrated with Gaussian back-end and then all calibrated scores are fused together using the multiple-class logistic regression for the final decision. Obviously, the proposed ensemble of the learned-exemplar dictionary (concatenation of class-based learned dictionaries) based LR approach can handle large sized training data efficiently along with providing the diversity gain. The performances are reported on two low dimensional utterance representations: the i-vectors and the JFA latent vector. *All the proposed LR systems are explained by considering the i-vector as the spoken utterance representation. Similar steps can be used when the JFA latent vector is chosen as the utterance representation.*

5.2 Proposed ensemble exemplar dictionary based LR system

The computational burden of sparse coding over an exemplar dictionary grows up with the increase of training examples. Towards addressing this issue, in [100], a single compact exemplar dictionary was created using two different approaches: random sampling and k-means clustering. In the random sampling approach, the language-specific sub-dictionary was created by randomly selecting the training examples from the particular language. Then, sub-dictionaries were concatenated to form a single

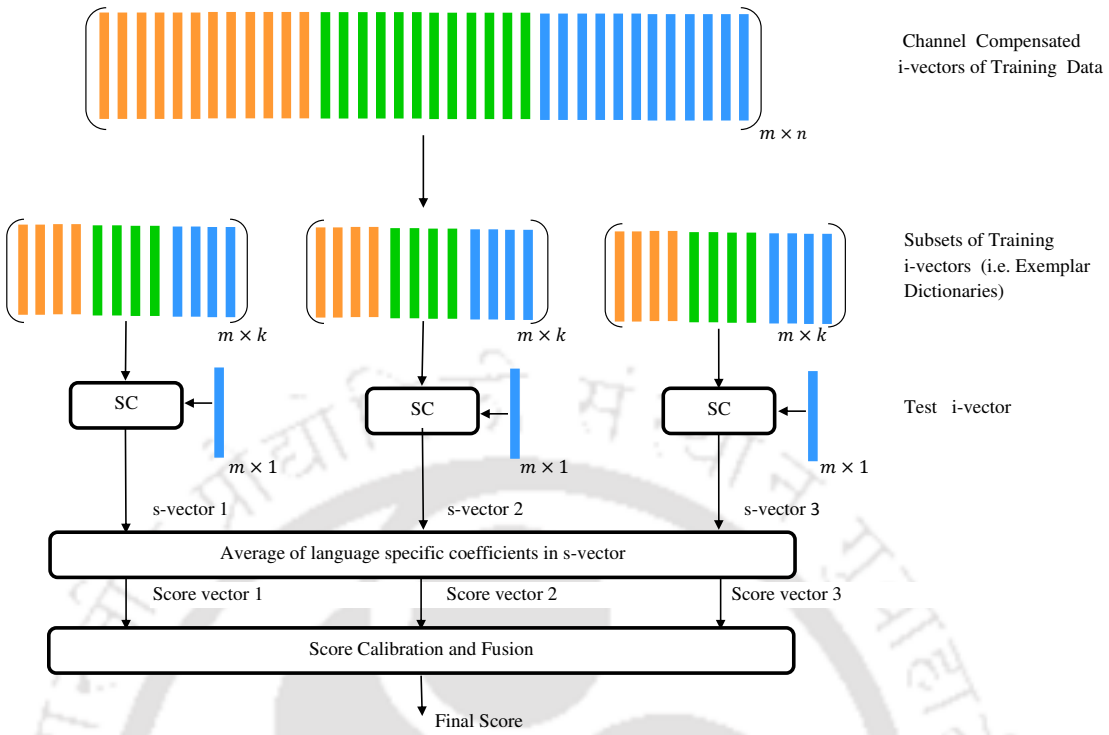

Figure 5.1: Flow diagram of the proposed ensemble exemplar dictionary based LR approach. For ease of illustration, only three language classes are considered and the corresponding session/channel compensated i-vectors are depicted in orange, green and blue colors. Similarly, the ensemble shows only three dictionaries for clarity. The given test i-vector is represented over the ensemble of exemplar dictionaries and the resulting sparse codes are denoted by s-vector 1, s-vector 2, s-vector 3, each of size $k \times 1$. The language-specific coefficients in each of the s-vectors are averages to produce the Score vector 1, Score vector 2, and Score vector3, respectively. Each of the scores undergoes through GB calibration. Finally, all the calibrated scores are fused using MLR for the language recognition (SC: Sparse Coding Algorithm).

compact exemplar dictionary. In this process, there is no guarantee that all training examples get utilized. For the fixed database, the number of un-utilized training examples increases with a decrease in the size of created sub-dictionaries. Towards addressing this problem to some extent, in [100], the authors made use of random subspace approach [189]. It involved the creation of multiple exemplar dictionaries through random sampling and the scores obtained from those dictionaries were averaged to get the final decision. In the random sampling process, the same training example may appear in more than one dictionary and the probability of the same is high for limited training data sets. Apart from the above-mentioned issues, the work in [100] did not optimize the size and the number of sub-dictionaries created.

In order to investigate these issues, we propose ensemble exemplar dictionary approach, which not only reduces the computational cost in sparse coding but may enhance the LR performance due

to diversity in the sparse coding of the target. The proposed approach doesn't require any random sampling and hence there is no chance of getting the same feature vector in different dictionaries (i.e., no repetition of selecting the same feature vector).

Figure 5.1 shows the flow diagram of the proposed ensemble exemplar dictionary based LR system. The salient steps involved in the proposed system are summarized below:

(i) Given a pool of m-dimensional WCCN session/channel compensated training i-vectors corresponding to each of the L languages in the task.

(ii) Create an exemplar dictionary by selecting k unique i-vectors from each of L languages. Let G be the number of such exemplar dictionaries get created.

(iii) For each of G exemplar dictionaries

  (a) Find the sparse code (i.e., the s-vector) of the test i-vectors.

  (b) Compute the average of language-specific coefficients of the s-vector to get the score vector.

  (c) Each of the scores undergoes through GB for calibration.

(iv) For final decision, fuse the resulting G scores using MLR.

5.3 Proposed ensemble learned-exemplar dictionary based LR system

To exploit the diversity, an ensemble of smaller sized exemplar dictionaries is created by partitioning the language-specific learned dictionaries. Each of the language-specific learned dictionaries is designed to have a same number of columns. Thus, unlike the simple exemplar dictionary, the size of the derived learned-exemplar dictionary can be kept fixed even when more and more training data is made available. Further for the best LR performance, the multiple learned-exemplar dictionaries in the ensemble approach are required to have only two atoms per language. The flow diagram of the proposed ensemble of learned-exemplar dictionary based LR approach is illustrated in Figure 5.2.

5.4 Experiments with 2007 NIST LRE data

The initial evaluations of the proposed ensemble of exemplar and learned-exemplar based LR approaches have been done using the 2007 NIST LRE evaluation data set comprising of 30 seconds duration utterances in closed test-set conditions. The database used, feature extraction and language

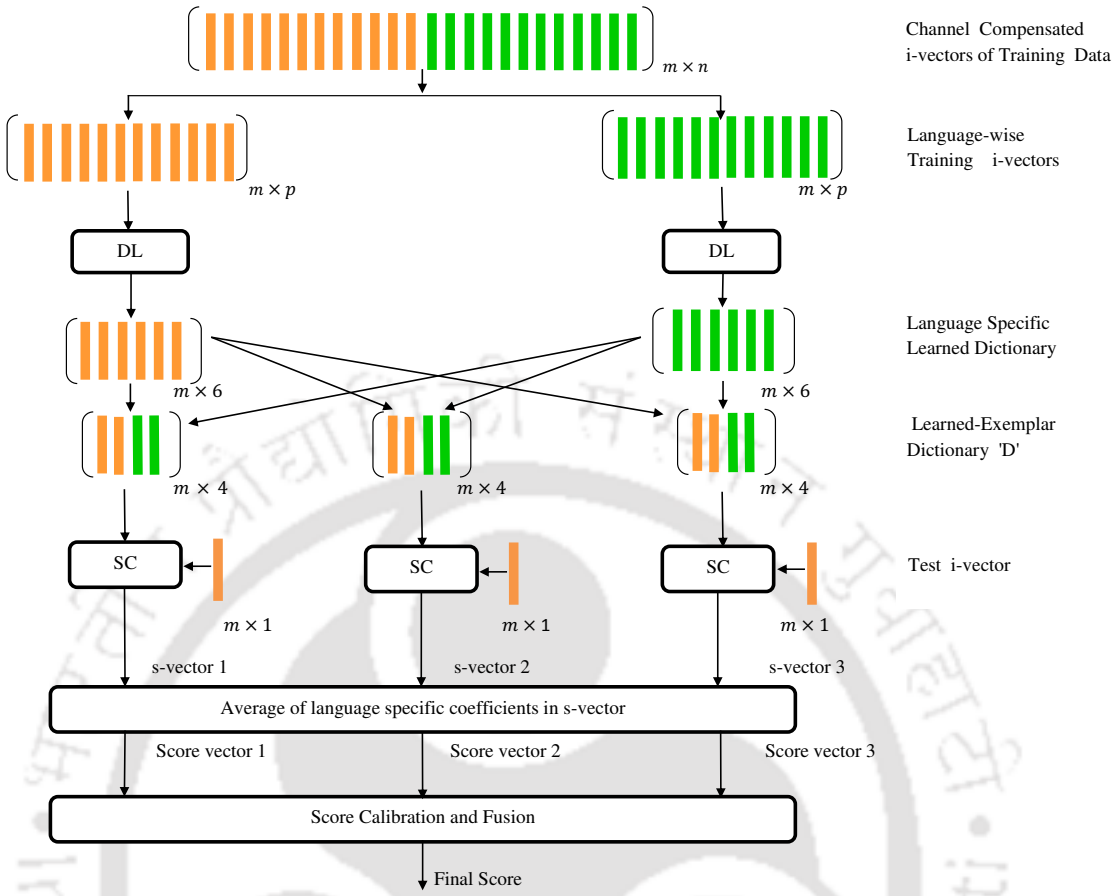

Figure 5.2: Flow diagram of the proposed ensemble learned-exemplar dictionary based LR approach. For ease of illustration, only two language classes are considered and the corresponding session/channel compensated i-vectors are depicted by orange and green colors. Also, only three dictionaries are shown in the ensemble. Given the training data, first, the language-specific learned dictionaries are created. The ensemble exemplar dictionaries are then derived by selecting two column from each of the language-specific learned dictionaries. The remaining flow of the system is identical to that of shown in Figure 5.1 (SC:Sparse Coding, DL:Dictionary Learning).

modeling have been considered the same as discussed in Section in 3.3.1. In addition to i-vector representation, in this work, the JFA-latent vector is also considered as the utterance representation. Among all languages in the training data, German has a minimum of 424 utterances. So, for a balanced single exemplar dictionary, we have chosen 420 training data per language (D/L). In an ensemble of exemplar dictionary learning, the choice of a number of exemplar dictionaries for a balanced training data may be crucial for the proper results. This number is directly related to the dictionary size (or a number of training data from each of the languages). In order to find the proper size of the individual dictionary, the number of training D/L is varied, and the performances in terms of C_{avg} and IDR corresponding to LD and LID are shown in Figure 5.3. The result shows that 2 numbers of training D/L perform

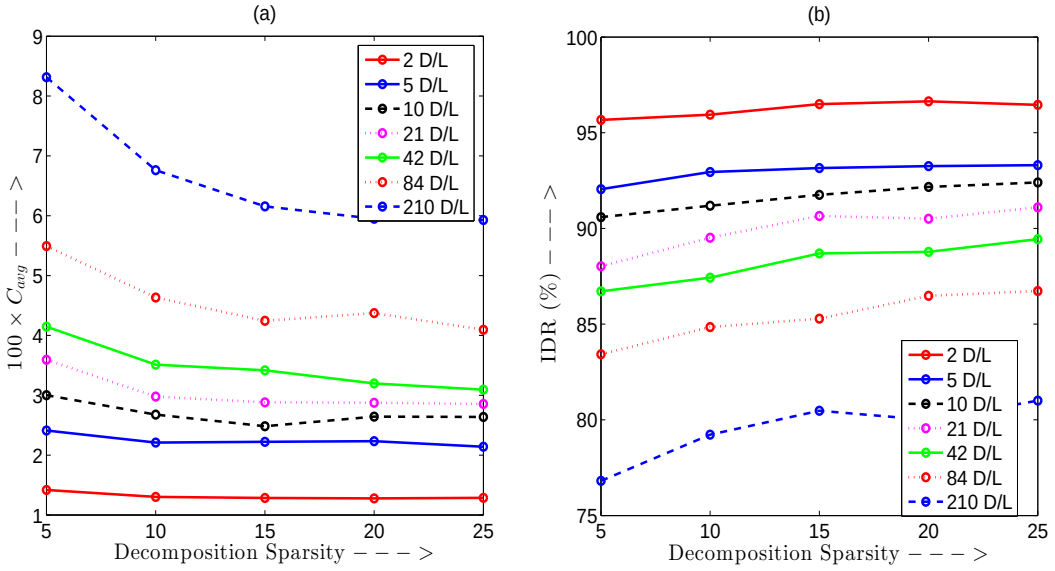

Figure 5.3: The effect of dictionary size on the ensemble of exemplar dictionary based LR systems on the i-vector representation with different decomposition sparsity. The equal number of examples per language is considered, thus the size of each dictionary is equal to the number of data per language (D/L) times the number of languages. The 2007 NIST evaluation data in close test-set condition with 30 seconds duration is used for evaluation. The results are shown for two different LR variants: (a) C_{avg} metric for LD task (b) IDR metric for LID task.

much better than the others. So, the size of each of the exemplar dictionary must be chosen to be 28. This also tells us that a total of 210 dictionaries given 420 training D/L are required for getting the maximum performance using an ensemble of the exemplar dictionary. In the SRC framework, the extraction of s-vectors of the target signals over multiple dictionaries becomes a computational burden, as it is directly proportional to the amount of training data involved. To address this problem, the idea of creating multiple sub-dictionaries are extended to learned-exemplar dictionary case. In an ensemble of learned-exemplar dictionary approach based LR, each of the learned-exemplar dictionaries is created with the help of well known K-SVD dictionary learning approach employing OMP, Lasso, and ENet regularization based sparse coding. In Section C.1, it was observed that the learned dictionary having only 28 dictionary atoms outperforms comparing with other dictionary sizes. So, the same size is considered while designing the individual learned-exemplar dictionary in the ensemble of learned-exemplar dictionary based LR approach. It is to be noted that although the dictionary size is the same, the number of the learned-exemplar dictionaries can be limited. For contrast purpose, a single learned-exemplar dictionary having 28 atoms (considering 2 atoms per language) is also developed. Apart from the dictionary size, the tuning parameters related to OMP, Lasso and ENet based

regularization are also carried out.

Considering the exemplar approach with OMP/Lasso, only one regularization coefficient, the decomposition sparsity γ/λ_1 needs to be varied. The value of regularization coefficient γ is varied in the range of 5 to 25 in steps of 5, whereas λ_1 is varied for the values of 0.001, 0.005, and in the range of 0.01 to 0.1 in steps of 0.01. In ENet, the value of λ_1 is selected based on the best performance obtained in Lasso, while another regularization coefficient λ_2 is varied in the range of 0.1 to 1 in steps of 0.1. The dictionary learning approach employing OMP/Lasso-based regularization requires tuning of dictionary sparsity (γ'/λ_1') as well as decomposition sparsity (γ/λ_1). The K-SVD employing OMP uses γ' and γ, which are varied in the range of 5 to 25 in steps of 5. However, with Lasso, the parameters λ_1' and λ_1 are varied for the values of 0.001, 0.005, and in the range of 0.01 to 0.1 in steps of 0.01. In ENet based dictionary learning, the values of λ_1' and λ_1 are selected based on the best performance of Lasso-based approach. The other two regularization coefficients λ_2' and λ_2 are varied in the range of 0 to 1 and 0 to 0.5, respectively in steps of 0.1. The best tuning parameters are selected to obtain the best score. The class-specific scores are averaged to get the language specific scores. In an ensemble approach, the language-specific scores from each of the parallel unit are calibrated using GB separately, and then fused the scores using MLR employing the FoCal toolkit [153] prior to final decision.

5.4.1 Results and discussion

The performances are shown for two different variants of LR, i.e., LD and LID. The performance of the LD systems is primarily evaluated using average detection cost function C_{avg} as per 2007 NIST LRE plan [154]. However, for LID, an evaluation metric considered is average identification rate. The results are reported on 30 seconds evaluation data in closed test-set condition. The detail of the performance metrics is described in 2.8. Separate systems are developed on the i-vector and the JFA latent vector as the low dimensional representation of the spoken language. In the following, we present the performance evaluation of the proposed ensemble exemplar and ensemble learned-exemplar dictionaries based LR systems.

5.4.1.1 Ensemble exemplar and learned dictionary based LR systems on the i-vectors

Table 5.1 shows the performances of a proposed ensemble of exemplar and learned-exemplar dictionary based LR approaches on the i-vector representation of utterances with best tuning parameters. On comparing with a single exemplar dictionary employing OMP, Lasso and ENet based regularization,

the relative improvements (in C_{avg}) of 86%, 68%, and 60%, respectively, in the case of an ensemble of 210 exemplar dictionaries. While in terms of IDR metric, the relative improvements of about 33%, 13%, and 10% are observed, respectively. The ensemble exemplar dictionary based LR system is noted to outperform not only the simple exemplar dictionary-based system but also the single learned-exemplar dictionary-based system. These improvements are attributed to the diversity in the scores for the given target obtained over multiple dictionaries. The idea of creating multiple sub-dictionaries are extended to learned-exemplar dictionary case. It can be observed that for the same number of dictionaries, the ensemble learned-exemplar employing ENet regularization based LR system outperforms the ensemble exemplar-based one. The further improvement noted is due to the efficient sparse representation over a learned-exemplar dictionary.

5.4.1.2 Ensemble exemplar and learned dictionary based LR systems on the JFA latent vector

In JFA, the language and session/channel variabilities are modeled separately whereas in the i-vector approach both are captured simultaneously. Therefore one needs to apply session/channel compensation to the i-vectors before creating the dictionaries. On the other hand, the JFA latent vector denoting the language space can be directly used for creating the dictionaries. In order to avoid any leftover nuisance in JFA latent vector, it is preferable to apply for suitable session/channel compensation on JFA latent vector too. As the LR task comprises 14 languages, the size of the latent vector in the JFA is also taken as 14. On employing these small size representations in place of 400-dimensional i-vectors, the computational cost of the proposed LR systems can be reduced hugely. This motivated us to explore the proposed ensemble exemplar and ensemble learned-exemplar dictionaries based LR approaches on the JFA latent vector as the utterance representation. The results are summarized in Table 5.2. It can be observed that the proposed ensemble of learned-exemplar dictionary based LR performs better compared to the single dictionary approach as well as the proposed ensemble of the exemplar dictionary. It is to be noted that the performance gain in the proposed ensemble of learned-exemplar dictionary employing i-vector representation based LR approach is relatively high compared to JFA-latent vector based one.

Table 5.1: Performances of ensemble of exemplar and learned-exemplar dictionary based LR approaches employing WCCN session/chaneel compensated *i-vector* with SRC. A single exemplar and a single learned-exemplar based LR systems are also considered for contrast purpose. The dictionary learning is performed using K-SVD employing OMP, Lasso and ENet regularization. For classification, SRC is used. The performances are shown on 30 seconds evaluation data of LRE07 for the best tuned parameter values. Bold values show the best performances obtained. The dictionary and decomposition sparsity of ENet are kept same as that of Lasso. The terms exemplar and learned-exemplar are denoted by XR, and LD-XR, respectively.

LR Sytems	Dictionary type/number/size	LD		LID	
		Paramter(s)	$100 \times C_{avg}$	Paramter(s)	IDR (%)
Contrast (using OMP)	XR/sngl/5880	$\gamma = 10$	9.39	$\gamma = 10$	72.76
	LD-XR/sngl/28	$\gamma' = 1, \gamma = 9$	3.06	$\gamma' = 1, \gamma = 13$	90.05
Contrast (using Lasso)	XR/sngl/5880	$\lambda_1 = 0.05$	3.95	$\lambda_1 = 0.05$	85.69
	LD-XR/sngl/28	$\lambda'_1 = 0.04, \lambda_1 = 0.005$	3.01	$\lambda'_1 = 0.04, \lambda_1 = 0.005$	89.84
Contrast (using ENet)	XR/sngl/5880	$\lambda_2 = 2$	3.20	$\lambda_2 = 2$	87.75
	LD-XR/sngl/28	$\lambda'_2 = 0, \lambda_2 = 1$	3.00	$\lambda'_2 = 0.1, \lambda_2 = 0.9$	89.83
Proposed (using OMP)	XR/Ensmbl(50)/28	$\gamma = 20$	2.51	$\gamma = 25$	92.92
	XR/Ensmbl(100)/28	$\gamma = 20$	1.76	$\gamma = 20$	94.26
	XR/Ensmbl(150)/28	$\gamma = 20$	1.47	$\gamma = 20$	95.26
	XR/Ensmbl(210)/28	$\gamma = 20$	1.28	$\gamma = 20$	96.64
	LD-XR/Ensmbl(10)/28	$\gamma' = 25, \gamma = 25$	2.38	$\gamma' = 10, \gamma = 25$	92.40
	LD-XR/Ensmbl(25)/28	$\gamma' = 25, \gamma = 15$	2.14	$\gamma' = 10, \gamma = 25$	93.33
	LD-XR/Ensmbl(50)/28	$\gamma' = 25, \gamma = 20$	1.84	$\gamma' = 15, \gamma = 15$	94.50
	LD-XR/Ensmbl(100)/28	$\gamma' = 5, \gamma = 10$	**1.26**	$\gamma' = 5, \gamma = 15$	**95.91**
Proposed (using Lasso)	XR/Ensmbl(50)/28	$\lambda_1 = 0.03$	2.51	$\lambda_1 = 0.005$	92.93
	XR/Ensmbl(100)/28	$\lambda_1 = 0.01$	1.73	$\lambda_1 = 0.005$	94.02
	XR/Ensmbl(150)/28	$\lambda_1 = 0.01$	1.51	$\lambda_1 = 0.02$	94.96
	XR/Ensmbl(210)/28	$\lambda_1 = 0.01$	1.26	$\lambda_1 = 0.005$	96.61
	LD-XR/Ensmbl(10)/28	$\lambda'_1 = 0.04, \lambda_1 = 0.001$	2.26	$\lambda'_1 = 0.04, \lambda_1 = 0.05$	92.71
	LD-XR/Ensemble(25)/28	$\lambda'_1 = 0.01, \lambda_1 = 0.001$	1.53	$\lambda 1' = 0.02, \lambda 1 = 0.01$	94.64
	LD-XR/Ensmbl(50)/28	$\lambda'_1 = 0.02, \lambda_1 = 0.001$	1.10	$\lambda'_1 = 0.01, \lambda_1 = 0.005$	96.59
	LD-XR/Ensmbl(100)/28	$\lambda'_1 = 0.02, \lambda_1 = 0.04$	**0.71**	$\lambda'_1 = 0.02, \lambda_1 = 0.06$	**97.76**
Proposed (using ENet)	XR/Ensmbl(50)/28	$\lambda_1 = 0.03, \lambda_2 = 0.1$	2.50	$\lambda_1 = 0.005, \lambda_2 = 0.1$	92.93
	XR/Ensmbl(100)/28	$\lambda_1 = 0.01, \lambda_2 = 0.1$	1.74	$\lambda_1 = 0.005, \lambda_2 = 0.1$	93.81
	XR/Ensmbl(150)/28	$\lambda_1 = 0.01, \lambda_2 = 0.1$	1.51	$\lambda_1 = 0.02, \lambda_2 = 0.3$	94.88
	XR/Ensmbl(210)/28	$\lambda_1 = 0.01, \lambda_2 = 0.1$	1.25	$\lambda_1 = 0.005, \lambda_2 = 0.1$	96.66
	LD-XR/Ensmbl(10)/28	$\lambda'_2 = 0.2, \lambda_2 = 0.4$	2.40	$\lambda'_2 = 0.1, \lambda_2 = 0.5$	90.35
	LD-XR/Ensmbl(25)/28	$\lambda'_2 = 0.7, \lambda_2 = 0.2$	1.41	$\lambda'_2 = 0.7, \lambda_2 = 0.5$	95.21
	LD-XR/Ensmbl(50)/28	$\lambda'_2 = 0.7, \lambda_2 = 0.5$	0.99	$\lambda'_2 = 0.2, \lambda_2 = 0.1$	96.94
	LD-XR/Ensmbl(100)/28	$\lambda'_2 = 0.5, \lambda_2 = 0.4$	**0.60**	$\lambda'_2 = 0.5, \lambda_2 = 0.1$	**98.13**

5.5 Experiments with 2009 NIST LRE data

5.5.1 Database

We consider only 12 languages which are sub-set of the 2009 NIST LRE [156] corpora due to limited training data in remaining languages. The sub-set includes American English, Cantonese Chinese, Mandarin and Russian, Farsi, Hindi, Korean, Vietnamese, Creole, Georgian, Turki, and Spanish languages. The training data is collected from the conversational telephone speech (CTS) only, which includes NIST SRE (2004, 2005, 2006 and 2008), 2007 NIST LRE supplementary and

Table 5.2: Performances of ensemble of exemplar and learned-exemplar dictionary based LR approaches employing WCCN compensation on *JFA latent vector* with SRC. A single exemplar and a single learned-exemplar based LR systems are also considered for contrast purpose. The dictionary learning is performed using K-SVD employing OMP, Lasso and ENet based regularization. Bold values show the best performances obtained for the proposed system. The terms exemplar and learned-exemplar are denoted by XR, and LD-XR, respectively.

LR Sytems	Dictionary type/number/size	LD		LID	
		Parameters(s)	$100 \times C_{avg}$	Parameters(s)	IDR (%)
Contrast (using OMP)	XR/sngl/5880	$\gamma = 5$	10.29	$\gamma = 10$	76.50
	LD-XR/sngl/28	$\gamma' = 1, \gamma = 12$	4.40	$\gamma' = 1, \gamma = 10$	86.26
Contrast (using Lasso)	XR/sngl/5880	$\lambda_1 = 0.04$	5.87	$\lambda_1 = 0.005$	83.44
	LD-XR/sngl/28	$\lambda'_1 = 0.001, \lambda_1 = 0.005$	4.55	$\lambda'_1 = 0.001, \lambda_1 = 0.001$	86.57
Contrast (using ENet)	XR/sngl/5880	$\lambda_1 = 0.04, \lambda_2 = 0.1$	6.84	$\lambda_1 = 0.005, \lambda_2 = 0.1$	82.87
	LD-XR/sngl/28	$\lambda'_2 = 0.4, \lambda_2 = 0.1$	4.30	$\lambda'_2 = 0.9, \lambda_2 = 0$	86.99
Proposed (using OMP)	XR/Ensmbl(50)/28	$\gamma = 20$	3.74	$\gamma = 15$	88.41
	XR/Ensmbl(100)/28	$\gamma = 15$	3.45	$\gamma = 10$	88.66
	XR/Ensmbl(210)/28	$\gamma = 10$	3.13	$\gamma = 15$	90.13
	LD-XR/Ensmbl(50)/28	$\gamma' = 15, \gamma = 15$	3.65	$\gamma' = 15, \gamma = 10$	88.31
	LD-XR/Ensmbl(100)/28	$\gamma' = 25, \gamma = 15$	3.43	$\gamma' = 25, \gamma = 25$	88.97
	LD-XR/Ensmbl(200)/28	$\gamma' = 20, \gamma = 15$	**3.14**	$\gamma' = 15, \gamma = 15$	**90.03**
Proposed (using Lasso)	XR/Ensmbl(50)/28	$\lambda_1 = 0.005$	3.73	$\lambda_1 = 0.001$	87.99
	XR/Ensmbl(100)/28	$\lambda_1 = 0.02$	3.58	$\lambda_1 = 0.001$	89.03
	XR/Ensmbl(210)/28	$\lambda_1 = 0.001$	3.37	$\lambda_1 = 0.005$	90.16
	LD-XR/Ensmbl(50)/28	$\lambda'_1 = 0.09, \lambda_1 = 0.005$	3.67	$\lambda'_1 = 0.04, \lambda_1 = 0.005$	88.26
	LD-XR/Ensmbl(100)/28	$\lambda'_1 = 0.02, \lambda_1 = 0.001$	3.34	$\lambda'_1 = 0.1, \lambda_1 = 0.001$	89.26
	LD-XR/Ensmbl(200)/28	$\lambda'_1 = 0.1, \lambda_1 = 0.005$	**3.04**	$\lambda'_1 = 0.09, \lambda_1 = 0.001$	**90.28**
Proposed (using ENet)	XR/Ensmbl(50)/28	$\lambda_1 = 0.005, \lambda_2 = 0.1$	4.19	$\lambda_1 = 0.001, \lambda_2 = 0.1$	86.90
	XR/Ensmbl(100)/28	$\lambda_1 = 0.02, \lambda_2 = 0.1$	3.97	$\lambda_1 = 0.001, \lambda_2 = 0.1$	87.95
	XR/Ensmbl(210)/28	$\lambda_1 = 0.001, \lambda_2 = 0.1$	3.28	$\lambda_1 = 0.005, \lambda_2 = 0.1$	89.57
	LD-XR/Ensmbl(50)/28	$\lambda'_2 = 0.6, \lambda_2 = 0$	3.68	$\lambda'_2 = 0.8, \lambda_2 = 0$	88.59
	LD-XR/Ensmbl(100)/28	$\lambda'_2 = 0.9, \lambda_2 = 0$	3.24	$\lambda'_2 = 0.1, \lambda_2 = 0$	89.38
	LD-XR/Ensmbl(200)/28	$\lambda'_2 = 0.6, \lambda_2 = 0$	**2.89**	$\lambda'_2 = 0.6, \lambda_2 = 0$	**90.41**

Babel [155] data-sets. The training dataset contains only 250 speech utterances from each of the language. The test data includes both VoA and CTS data from these 12 languages. The experiments are performed in closed set condition on 3, 10 and 30 seconds duration segments. The details of data distribution are discussed in 2.7.2.

5.5.2 Shifted delta cepstral feature

Speech signal is processed in frames of 20 ms duration at 10 ms frame rate. For each frame, standard 7 MFCCs are computed using 23-channel Mel filter bank. The RASTA filtering [190] followed by CMN is applied on MFCCs after silence removal. Then, 49-dimensional shifted delta cepstral (SDC) [125] coefficients using 7-1-3-7 scheme are obtained and concatenated to 7 static MFCCs to form 56-dimensional feature vector.

5.5.3 Language modeling, classifiers, and dictionary learning

A language-independent GMM-UBM of 1024 Gaussians is employed to build the LR system by pooling approximately 1 hour of data from all 12 languages. A total variability matrix of rank 400 is trained on training data and is used for computation of the i-vectors as the utterance representation. The dimension of i-vector is selected to be 400 based on experimentation. The other utterance representation used in this work is based on JFA latent-vector which is derived during JFA processing. In JFA, only 12 LFs and 200 CFs are considered. Thus, the length of the JFA latent vector is chosen to be 12. We have investigated SRC classifier in all the dictionary based LR approaches, which includes single exemplar dictionary, single learned dictionary, an ensemble of exemplar dictionary and ensemble of the learned-exemplar dictionary. For dictionary design, K-SVD based dictionary employing OMP, Lasso, and ENet regularization is considered. The LARS sparse coding algorithm is used to solve the Lasso/ENet based sparse coding problem while OMP is used for solving l_0-norm minimization problem. The classifiers namely SVM, G-PLDA, MLR, GG, CDS are also used for contrast purpose. In ensemble dictionary based LR approach, scores from each of the parallel units are calibrated using Gaussian backend. Further, the calibrated scores are fused using MLR. The FoCal toolkit [153] is used for this purpose. The different regularization parameters corresponding to each of the sparse coding problems are tuned to get the best performances.

5.5.4 Results and discussion

The performance of the proposed ensemble of exemplar and learned-exemplar dictionary employing SRC for LR task are summarized in Table 5.3. The performances of LD and LID are measured in terms of C_{avg} and IDR metric, respectively. It is observed that the i-vector based LR system employing CDS classifier is found to be better than SVM, G-PLDA, MLR, GG, and SRC over single exemplar dictionary (created by concatenating 3000 training data from 12 languages) based LR system. However, there is a slight degradation in the CDS based LR system on comparing with single learned dictionary consisting of only 24 dictionary atoms (2 atoms per language). It is also observed that ensemble of exemplar dictionary-based approach outperforms a single exemplar dictionary. The multiple exemplar dictionaries are created by partitioning whole training data (i.e., 3000 utterances) into 125 exemplar dictionaries, each with 24 training data (2 data per language). Thus, the improvement achieved with the ensemble of exemplar dictionary approach is attributed to the diversity achieved in the sparse

Table 5.3: Performance comparison between standard WCCN compensated i-vector with different classifiers (SVM:support vector machine, G-PLDA:Gaussian PLDA, MLR:multi-class logistic regression, GG:generative Gaussian, CDS:cosine distance scoring, SRC:sparse representation based classification), the single learned-exemplar dictionary based LR and ensemble of exemplar and learned-exemplar dictionary based LR. The 2009 NIST evaluation data-set in closed set condition on 3, 10 and 30 sec segments are used. Bold values show the best performances obtained in different kind of LR systems.

Dictionary type/number/size	Regularization	Classifier	LD			LID		
			$100 \times C_{avg}$			IDR (%)		
			30 sec	10 sec	3 sec	30 sec	10 sec	3 sec
–	–	SVM	5.30	12.71	25.89	83.65	64.44	37.83
		G-PLDA	4.69	11.78	24.66	85.54	66.75	39.86
		MLR	4.52	11.57	24.24	86.00	67.29	39.85
		GG	4.50	**11.35**	24.30	86.41	67.65	40.15
		CDS	**4.48**	11.43	**24.02**	**86.52**	**67.80**	**40.47**
XR/sngl/3000	OMP	SRC	6.06	13.96	26.49	81.93	62.40	37.18
XR/sngl/3000	Lasso		4.79	12.31	25.44	85.05	66.15	38.98
XR/sngl/3000	ENet		**4.53**	**11.77**	**24.84**	**86.40**	**67.23**	**39.85**
LD-XR/sngl/24	OMP	SRC	4.55	11.42	24.33	86.41	68.00	40.44
LD-XR/sngl/24	Lasso		**4.32**	11.40	24.22	86.39	67.98	40.34
LD-XR/sngl/24	ENet		4.39	**11.18**	**24.04**	**86.50**	**68.52**	**40.72**
XR/Ensmbl(125)/24	OMP	SRC	3.69	10.31	22.78	**89.80**	**71.03**	**42.61**
XR/Ensmbl(125)/24	Lasso		3.68	10.25	22.79	89.54	70.96	42.71
XR/Ensmbl(125)/24	ENet		**3.68**	**10.14**	**22.66**	89.50	70.78	42.52
LD-XR/Ensmbl(100)/24	OMP	SRC	3.41	10.04	22.40	89.19	70.74	42.25
LD-XR/Ensmbl(100)/24	Lasso		3.05	9.20	21.80	91.05	74.05	45.93
LD-XR/Ensmbl(100)/24	ENet		**2.21**	**7.40**	**18.47**	**93.45**	**79.10**	52.70

coding of the target over multiple dictionaries. This approach may not be feasible if the size of the training data is large because it requires many sparse coding of the test feature vector. To address this problem, we further explore ensemble of the learned-exemplar dictionary and found to outperforms ensemble of the exemplar dictionary with reduced numbers of the dictionary. The multiple small-sized learned-exemplar dictionaries are created by concatenating 2 dictionary atoms from each of the language-specific learned dictionaries. Thus, the number of the small-sized learned-exemplar dictionary is restricted by the size of the language-specific dictionary. The improved performances of the proposed ensemble dictionary-based approach are noted in both i-vector and JFA-latent vector as utterance representations. It is to be noted that the performance of JFA-latent vector is slightly degraded in comparison with i-vector.

5.6 Summary

In this chapter, we have explored the ensemble of exemplar and learned-exemplar dictionaries based sparse representation on the i-vector and the JFA-latent vector utterance representations for LR task. For contrast purpose, a single exemplar and learned-exemplar dictionary based LR systems have

been also developed. In a dictionary-based approach, the sparse representation of target signals over single/multiple learned-exemplar dictionaries is computed. For dictionary learning, K-SVD dictionary employing OMP, Lasso, and ENet regularization has been used. The class-wise mean of the s-vector has been computed the obtain the language-specific scores. The calibration was done using GB. Finally, the scores from each of the parallel unit have been fused using MLR for the final decision. The effectiveness of the proposed LR approaches have been evaluated on two speech corpus: the 2007 and 2009 NIST evaluation data sets. From the experimental results, it was observed that the ensemble of exemplar dictionary based LR approach performs much better than the single exemplar dictionary based LR approach. This indicates that the improvement achieved with the ensemble of the dictionary approach is attributed to the diversity achieved in the sparse coding of the target over multiple dictionaries. However, the number of such exemplar dictionaries would become very high if the size of the training dataset is large. On account that many sparse codings of the test feature vector, the computational burden of the proposed approach becomes prohibitive. This problem is tackled by the use of a reduced number of multiple learned dictionaries. In comparison, the proposed ensemble of learned-exemplar dictionary based LR has been found to outperform the ensemble of exemplar dictionary based LR approach, even in the reduced number of learned dictionaries. This further improvement may be due to the extraction of efficient sparse representation over a learned dictionary. The similar performance trends have been observed in JFA-latent vector based LR system.

Motivated by the significant performance with dictionary learning based LR approach, the recently proposed deep neural network based bottleneck features for i-vector representation in the sparse coding framework will be proposed in the next chapter.

6

DNN Conditioned Sparse Coding based Language Recognition

Contents

Following the seminal paper by Hinton *et al.* [191], the research on the deep neural networks (DNN) has grown by leaps and bounds. The DNN have pervaded in almost all pattern recognition tasks. The recent works have demonstrated that the DNN based approaches not only outperform the existing approaches but also augment them successfully. In the context of language recognition, the DNN has been used in following different ways:

- Direct classifiers,
- Extractor of novel (bottleneck) features,
- Extractor of more robust statistics,
- Capturing of high-level (phonetic sequence) information.

In the following sections, we briefly review the salient works in each of the above-mentioned categories.

6.1 DNN based language classifier

Broadly there are two kinds of multilayer neural networks: feed-forward and recurrent. The feed-forward networks perform mapping task well, while the recurrent networks having memory excel in the modeling of the sequence of events. In the following sub-section, we described two dominant DNN classifiers belonging to each of the kinds namely time delay neural network (TDNN) and long short-term memory (LSTM). These classifiers have been used for developing the contrast LR systems later in this chapter.

6.1.1 Time delay neural network

The TDNN [192] is a multilayer feedforward neural network, and must have two desirable properties. First, the network should have the ability to represent relationships between acoustic events in time. Second, the features learned by the network should be shift invariant. In many neural networks, the basic unit computes the weighted sum of its inputs and then passes this sum through an activation function (e.g., sigmoid function, hyperbolic tangent, etc.). In TDNN, this basic unit is altered by introducing delays D_1 through D_N as shown in Figure 6.1. Here, each of the inputs u_j, $j = 1, 2, \ldots, J$ are delayed from D_1 through D_N, and the weighted sum of the undelayed and delayed inputs are computed and passed through an activation function.

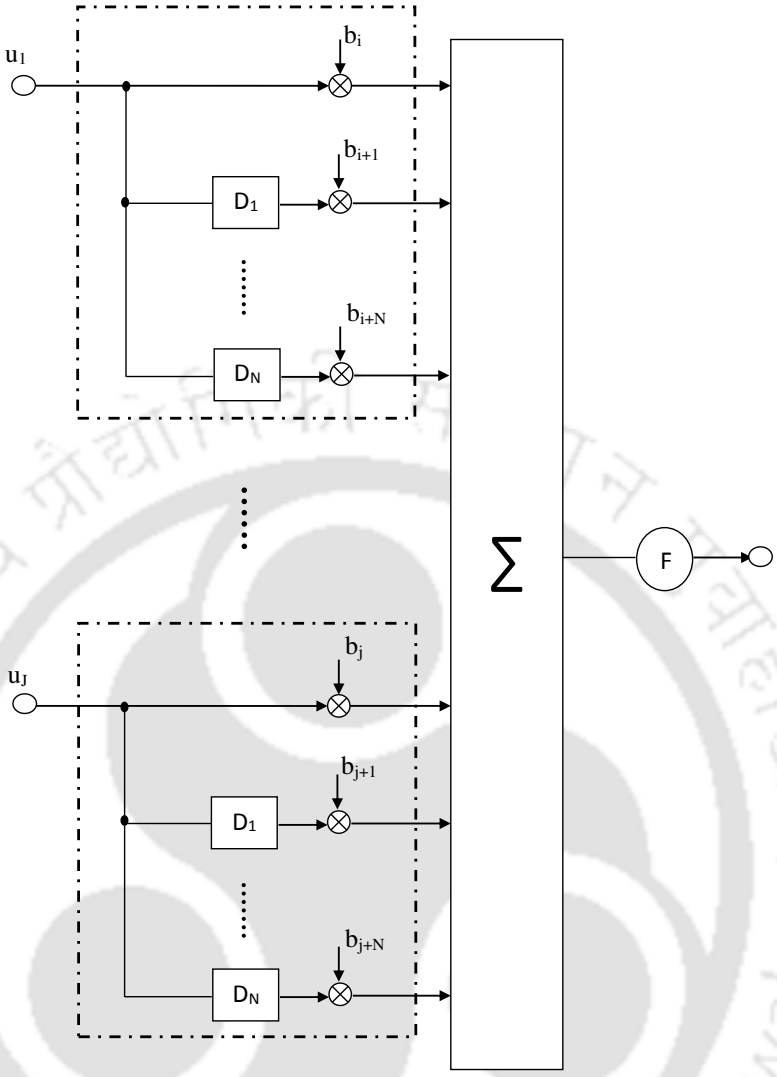

Figure 6.1: Structure of the time delay neural network basic unit.

One of the most popular learning technique used to train the neural networks is the backpropagation algorithm [193]. In this, the mean-squared error as a function of the weights is optimized using a gradient descent method. The learning procedure involves two passes through the network, i.e., forward pass and backward pass. In the forward pass, an input training is applied to the network with keeping the weights fixed. Starting from the input layer and working forward to the output layer, the outputs of all the units are calculated at each level. Then, the error between the desired and estimated output is calculated. During the backward pass, the derivative of this error is backpropagated through the network, and weights are adjusted so as to reduce the error. This procedure is iteratively done for all the training patterns until the network converges to the desired output.

Recently, Peddinti *et al.* [114] has proposed a sub-sampling technique in TDNN architecture, where hidden activations are computed at only a few time steps. This has been done to model the long temporal context efficiently, keeping training times comparable to standard feed-forward DNN.

6.1.2 Long short-term memory

The LSTM networks are a special kind of recurrent neural network (RNN) capable of learning long-term dependencies. The LSTM resolves the vanishing gradient problem (and also sometimes exploding gradient problem) of basic RNN. It was originally proposed by the German researchers Sepp Hochreiter and Jürgen Schmidhuber in the mid 1990s [115]. The most commonly used LSTM setup in the literature was originally described by Graves and Schmidhuber [194]. In this, full backpropagation through time training for the LSTM networks has been presented incorporating the changes suggested by Gers *et al.* [195, 196] into the original LSTM. A detailed analysis of several LSTM variants can found in [116].

Fig. 6.2 shows a LSTM cell block which is the core idea behind the LSTM architecture. It maintains its state over time, and non-linear gating units which regulate the information flow into and out of the cell.

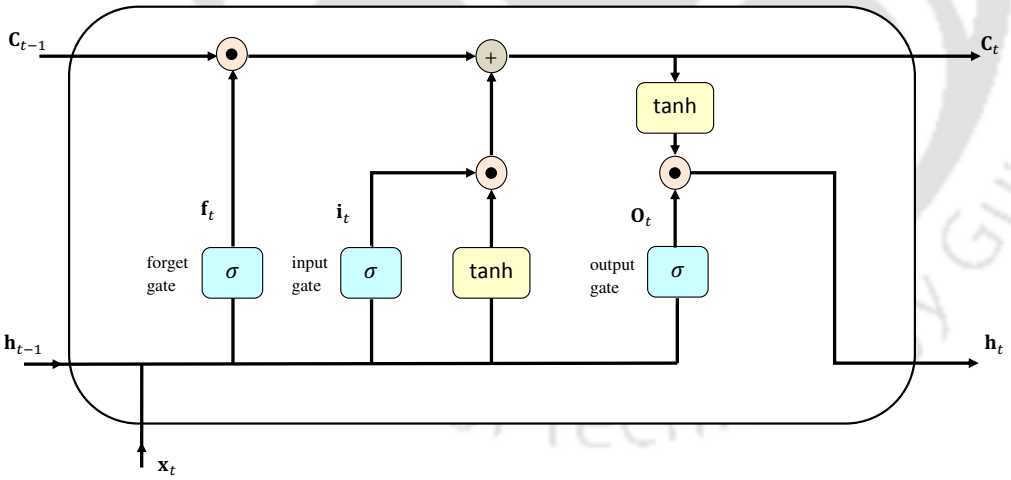

Figure 6.2: Schematic of LSTM cell block

The LSTM cell block has three gates (input, output and forget) and activation functions. The output of previous block is recurrently given as input to the current block as well as the gates. Let $\mathbf{x}_t$ be the input vector at time t, $\mathbf{h}_{t-1}$ be the block output vector at time $t-1$, N is the number of LSTM blocks, and M is the number of inputs. Then, we get the following weights for an layer:

- Input weights: $\mathbf{W}^z,\ \mathbf{W}^i,\ \mathbf{W}^f,\ \mathbf{W}^o\ \in\ \mathbb{R}^{N\times M}$
- Recurrent weights: $\mathbf{R}^z,\ \mathbf{R}^i,\ \mathbf{R}^f,\ \mathbf{R}^o\ \in\ \mathbb{R}^{N\times N}$
- Bias weights: $\mathbf{b}^z,\ \mathbf{b}^i,\ \mathbf{b}^f,\ \mathbf{b}^o\ \in\ \mathbb{R}^{N}$

During forward pass, the vector formulas for a LSTM layer are given as:

a) Block input denoted by $\mathbf{z}_t$ at time t is given as

$$\mathbf{z}_t = \tanh(\mathbf{W}^z\mathbf{x}_t + \mathbf{R}^z\mathbf{h}_{t-1} + b^z) \tag{6.1}$$

b) Input gate denoted by $\mathbf{i}_t$ at time t is given as

$$\mathbf{i}_t = \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{R}^i\mathbf{h}_{t-1} + \mathbf{b}^i) \tag{6.2}$$

It decides whether input state enters internal state or not.

c) Forget gate denoted by $\mathbf{f}_t$ at time t is given as

$$\mathbf{f}_t = \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{R}^f\mathbf{h}_{t-1} + \mathbf{b}^f) \tag{6.3}$$

It decides whether internal state forgets the previous internal state or not.

d) Output gate denoted by $\mathbf{f}_t$ at time t.

$$\mathbf{o}_t = \sigma(\mathbf{W}^o\mathbf{x}_t + \mathbf{R}^o\mathbf{h}_{t-1} + \mathbf{b}^o) \tag{6.4}$$

It decides whether internal state passes its value to output or not.

e) Memory/cell denoted by $\mathbf{c}_t$ at time t is given as

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i}_t \odot \mathbf{z}_t \tag{6.5}$$

f) Block output denoted by $\mathbf{h}_t$ at time t.

$$\mathbf{h}_t = \tanh(c_t) \odot \mathbf{o}_t \tag{6.6}$$

where $\odot$ denotes element-wise product. The *logistic sigmoid* and *hyperbolic tangent* activation functions are denoted by σ and tanh, respectively. To learn the precise timing of the output, one can use the peephole connections [196]. During backward pass, full backpropagation through time training for LSTM networks has been used.

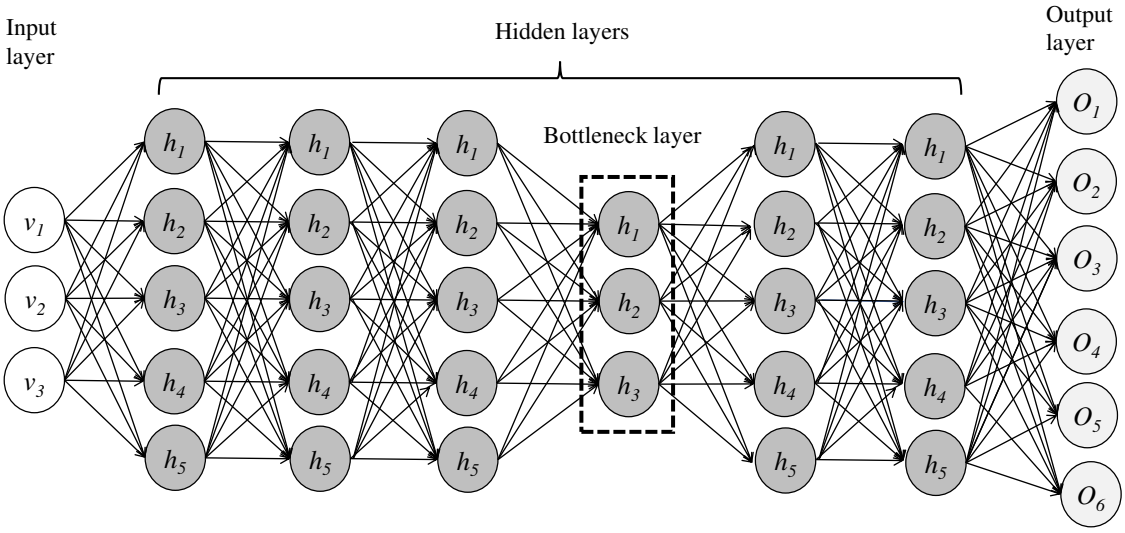

Figure 6.3: Broad structure of deep neural network employed for extracting the bottleneck features.

6.2 Bottleneck feature extraction

In this section, we first present the structure of DNN employed for extracting the features. The frame level outputs extracted from the bottleneck layer corresponding to input acoustic features are referred to as the bottleneck features (BNF).

The BNF can be generated from one of the hidden layers which are having less number of units or nodes as compared to the hidden layers on either side. The bottleneck layer (narrowest hidden layer) produces the output which is a compact version of the original inputs as it forces to represent information in low dimension. Figure 6.3 shows the broad structure of a deep network with one input layer, multiple hidden layers, and an output layer. Among the hidden layers, the layer having less number of nodes with respect to its neighboring layers is known as the bottleneck layer. In this work, $\pm N$ frames (left and right context = N) of spectral features (eg., MFCC) are spliced together, and the dimensionality is reduced by LDA using the acoustic states as classes. The reduced dimensional features are further spliced together with $\pm M$ frames (left and right context = M) and applied to the input layer of DNN. The BNF is extracted from a DNN trained for ASR using Kaldi [197] framework. In ASR, the pronunciations of all words are represented by a sequence of Senones (e.g., the tied-triphone states), trained on top of LDA followed by maximum likelihood linear transform features using the transcribed speech corpus.

6.3 Extraction of robust statistics

In recent times, DNN posteriors from the output layer of DNN trained for ASR [118] are used for extracting statistics required for the i-vector computation instead of GMM-posterior from the standard GMM-UBM. Next, we briefly describe how GMM/DNN based posteriors are computed and then used in the i-vector computation.

6.3.1 The i-vector extraction using GMM posteriors

The GMM-posterior is the per-mixture posterior probability of each feature vector computed using UBM, which is essentially a speaker/language-independent GMM. The GMM posteriors along with utterance specific feature vectors are used to accumulate the zeroth and the first-order sufficient statistics. Finally, the i-vector extraction is done using these accumulated statistics and the details are already described in Section 2.3.4.

6.3.2 The i-vector extraction using DNN posteriors

In [107], an alternate approach to extracting statistics based on DNN trained for ASR has been proposed. The DNN replaces the GMM to compute the posterior of the frames with respect to each of the classes in the model. In DNN, the classes are Senones obtained using a standard decision tree for ASR while in the case of GMM, the classes correspond to the individual Gaussians of the mixture model. In order to obtain the UBM parameters from DNN trained for ASR, let us define the following terms:

- K be the number of classes as the Senones defined by the ASR decision tree,
- $\mathbf{x}_t^i$ is the t-th speech frames from i-th speech segment,
- $\gamma_{kt}^i \approx p(k|\mathbf{x}_t^i)$ is the posterior probability of the k-th class for $\mathbf{x}_t^i$,
- η_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the weight, mean and covariance of the k-th class, respectively.

The UBM parameters using DNN trained ASR are given by

$$\eta_k = \sum_{i,t} \gamma_{kt}^i \tag{6.7}$$

$$\boldsymbol{\mu}_k = \frac{1}{\eta_k} \sum_{i,t} \gamma_{kt}^i \mathbf{x}_t^i \tag{6.8}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\eta_k} \sum_{i,t} \gamma_{kt}^i \mathbf{x}_t^i (\mathbf{x}_t^i)' - \boldsymbol{\mu}_k \boldsymbol{\mu}_k' \tag{6.9}$$

where the ASR system is used to compute the posteriors for each class k.

The sufficient statistics required for the i-vector estimation is computed by the following two equations:

$$0^{th} \text{ order statistics} : N_k^i = \sum_t \gamma_{kt}^i \tag{6.10}$$

$$\text{centralized } 1^{st} \text{ order statistics} : \tilde{\boldsymbol{F}}_k^i = \sum_{t=1} \gamma_{kt}^i (\mathbf{x}_t^i - \boldsymbol{\mu}_k) \tag{6.11}$$

Once the statistics are computed, the i-vector extraction procedure is done in the same way as it is done in the case of GMM based statistics.

6.4 Capturing of phonetic sequence information

The existing neural network based LR systems are developed using acoustic features like Mel filter banks (Fbanks) or MFCC [9]. In those systems, the information about the phonetic sequence is largely overlooked which are demonstrated to possess more discriminative high language level information than the acoustic features. In phonetic temporal network (PTN) approach [4,157], an attempt is made to improve the performance of the existing acoustic-domain LR systems using phonetic information. This methodology makes use of a phone-discriminative model to extract frame-level phonetic features and then feed them as input to an LSTM based LR system as shown in Figure 6.4. Originally the LSTM-based LR systems were built with raw acoustic features [198–200]. Later, the PTN based approach rediscovers the value of phonetic features to enhance the LR systems which have been largely overlooked due to the popularity of the i-vector based LR approach. All the recently proposed DNN-based works mentioned in the above sections are developed for the contrast purposes. On surveying the recent works on DNN for LR, we have not found any work which discusses the popular sparse coding approach. In Chapter 3, we have presented the sparse coding of i-vector utterance representation over different variants of the learned dictionary and noted to yield competitive language recognition performances when compared with the i-vector based LR employing classifiers like GG, CDS, SVM, MLR, and G-PLDA. In that approach, SDC features were used as the front-end features for i-vector extraction. Due to the superior performances of sparse representation based LR, we have investigated the sparse representation framework on DNN derived bottleneck features based i-vector representation.

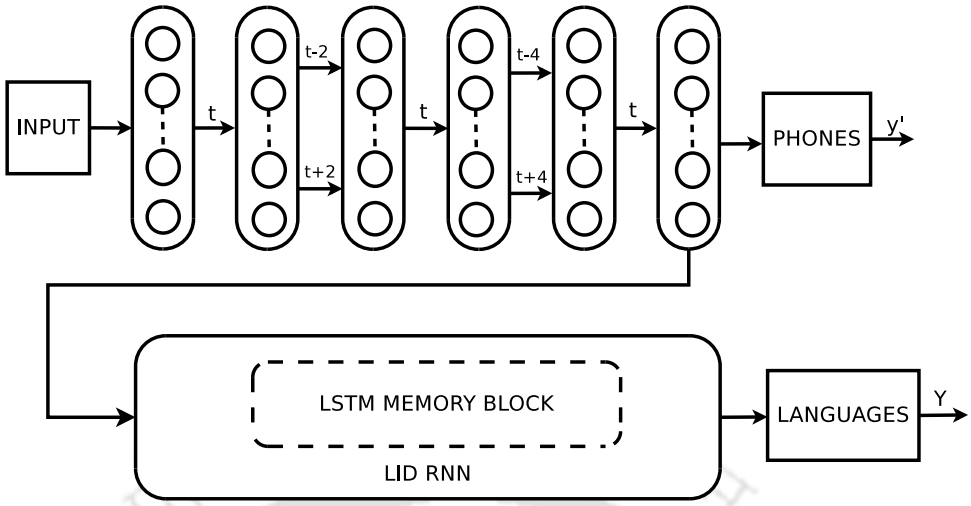

Figure 6.4: Block diagram of the phonetic temporal network (PTN) LID system [4]

6.5 DNN conditioning of sparse coding-based LR system

In this chapter, we have explored the sparse coding of the bottleneck features based i-vectors for LR task. For i-vector representation, DNN-based bottleneck front-end features are used, unlike the conventional MFCC based features. Both GMM/DNN based posterior computation techniques are incorporated in the i-vector computation and are used for contrast purposes. In literature, the performance of bottleneck features based i-vector employing GMM posteriors is superior compared to DNN posteriors based one. Thus, the sparse coding framework is only explored on the bottleneck features based i-vector employing GMM posteriors. For s-vector extraction, three discriminative dictionaries namely learned-exemplar, D-KSVD [165] and LC-KSVD [166] based approaches have been explored. The details of the above mentioned learned dictionary based LR can be found in Sections 3.4.4 and 3.4.5.

6.6 Experimental setup

6.6.1 Database

In this work, THCHS30 Chinese transcribed speech database [201] is used for training the ASR based DNN system. The training and evaluation data-sets are considered as per AP17-OLR Challenge [157]. The performances of the LR systems are measured in closed set condition on three different evaluation data sets corresponding to 1 second, 3 seconds and full-length utterances.

6.6.2 Feature extraction

The work presented in this chapter uses MFCC/BNF features for the i-vector representation while the FBank features are used for developing the PTN based LR system. Speech signals are analyzed with 20 ms Hamming window keeping frame shift of 10 ms and the pre-emphasis factor of 0.97.

6.6.2.1 MFCC features

As per the AP-17 OLR challenge protocol [157], the i-vector system is developed using 20-dimensional MFCC features including the log energy which is augmented with first and second order derivatives, thus resulting in 60-dimensional feature vectors. We have used 23-channel Mel filterbank to extract the 20-dimensional MFCC features. For feature level channel compensation, the cepstral mean normalization has also been applied to the computed acoustic feature vector. An energy-based VAD is used to remove the frames corresponding to silence regions.

6.6.2.2 Bottleneck features

For extracting the BNF, a DNN with 5 hidden layers and 1 bottleneck layer has been trained using THCHS30 Chinese speech corpus. The hidden layers consist of 1024 nodes, while the bottleneck layer has 60 nodes. The network uses the *tanh* as the non-linearity function. The bottleneck layer is placed after the third hidden layer. The DNN is trained with keeping the number of epochs as 20 and the mini-batch size as 128. A 91-dimensional feature vector is obtained by concatenating 13 dimensional MFCCs spliced across ± 3 frames. The dimensionality of the resulting feature vectors is reduced to 40 by applying LDA. These 40-dimensional feature vectors are further spliced across ± 10 frames and used for training the DNN. Once the DNN is trained, the speech data for which the BNF are to be extracted undergo the similar feature processing and then passed through this trained network. So obtained 60 dimensional features at the output of the bottleneck layer are referred to as the BNF.

6.6.2.3 FBanks features

The TDNN, LSTM, and PTN based LR systems use 40-dimensional Fbanks as the raw spectral features, extracted from each of the speech frames. For the Fbank feature computation, a 40-channel mel filterbank is used. In TDNN, the features are sliced across ± 4 frames while the symmetric 2-frame window is used to splice the neighboring frames in the case of LSTM.

6.6.3 PTN sytsem

The phonetic DNN model is a 6 hidden layers TDNN with *p-norm* activation function, trained using THCHS30 database [201]. The 40-dimensional Fbank features as raw input features are applied to the phonetic DNN. These input features are appended with splice neighboring features with symmetric 4-frame window for the TDNN and symmetric 2-frame window for the LSTM. Each TDNN layer has 2048 units. The number of cells of the LSTM is taken to be 1024. The Kaldi toolkit is used for the development of PTN based LR system.

6.6.4 i-vector modeling, classifiers and calibration

In the i-vector modeling, the sufficient statistics corresponding to each of the training and test segments are computed using the GMM posteriors from conventional GMM-UBM as well using the recently proposed DNN posteriors. The language-independent UBM consisting of 2048 Gaussian mixtures on a set of MFCC/BNF features pooled from each of 10 languages in AP17-OLR database is built. The DNN posteriors are computed from the output layer of DNN having bottleneck layer as shown in Figure 6.3. For computing the i-vectors, a total variability matrix $\boldsymbol{T}$ of 400 columns is trained on the training data following the procedure as described in [202].

The CDS classifier is used for the i-vector based LR system employing front-end MFCC/BNF features. In addition, SRC is also explored. In SRC, the coefficients of non-zero locations are used to determine the true class of the test sample. In this chapter, s-vectors corresponding to bottleneck features based i-vector representation employing GMM posteriors are investigated. For s-vectors, three discriminatively learned dictionaries namely learned-exemplar, D-KSVD, and LC-KSVD dictionary are explored. In D-KSVD and LC-KSVD based LR, s-vectors are projected on a linear classifier to obtain the class-similarity vector $\mathbf{c_w}$ and used to decide the language being spoken in the test utterances. The details are discussed in Sections 3.2.4 and 3.2.5, respectively. The score calibration is performed using Gaussian backend followed by MLR prior to the final decision.

6.7 Results and discussion

The performances of the proposed and contrast systems are reported on an AP17-OLR dataset in closed set condition for three different test durations. The LR is broadly divided into two tasks, the language detection and language identification, and their performances are measured in terms of the

Table 6.1: Performance comparison between TDDN, LSTM and PTN based LR systems.

LR system	Calibration	full-length		3 seconds		1 second	
		LD	LID	LD	LID	LD	LID
		$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR
TDNN	None	13.64	72.37	14.15	71.35	14.99	69.70
LSTM		14.36	72.59	15.09	71.06	15.65	68.65
PTN		**7.24**	**82.07**	**7.50**	**82.14**	**11.90**	**71.80**
PTN	GB+MLR	**1.76**	**94.62**	**3.21**	**90.59**	**8.41**	**77.70**

average cost detection function (C_{avg}) and the identification rate (% IDR), respectively. Table 6.1 shows the performance comparison between TDNN, LSTM and PTN based LR. It is observed that the PTN performs much better than the TDNN and LSTM. The performances of BNF based i-vector system employing GMM/DNN posteriors with three session/channel compensation techniques are shown in Table 6.2 and Table 6.3, respectively. Among the session/channel compensation methods, the WCCN is found to give satisfactory performance compared to the LDA and LDA followed by WCCN. In Table 6.4, the LR performances of GMM/DNN posteriors based i-vector representation on conventional MFCCs and bottleneck features are reported. It is observed that using bottleneck features, the performance of GMM posteriors based i-vector is found to give satisfactory performance compared to DNN posteriors based one. However, opposite trend is noted using MFCC features. Among the contrast LR systems, the best performance is achieved with bottleneck features based i-vector employing GMM posteriors, and referred as baseline LR.

In addition to contrast LR, the performances of the proposed learned dictionary based LR approaches on i-vector utterance representation using BNF and GMM posteriors are also shown in Table 6.4. On comparing with the baseline method, the performances of the proposed learned dictionary based LR approaches are found to be superior except a slight degradation in learned-exemplar case with full length test segments. In dictionary learning approaches, the parameters are tuned and the best values are selected. The dictionary sparsity λ_1' is varied for values from 0.005, 0.01, 0.015, 0.02 to 0.1 with increment of 0.01; decomposition sparsity λ_1 is varied for the values of 0.001, 0.005, 0.01, 0.015, and 0.02; $\sqrt{\alpha}$ and $\sqrt{\beta}$ vary between 0.1 to 0.5 with increment of 0.1. The best performances with LC-KSVD2 on full-length segments are noted with following tuning parameters: dictionary sparsity $\lambda_1' = 0.05$, decomposition sparsity $\lambda_1' = 0.01$, $\sqrt{\alpha} = 0.4$ and $\sqrt{\beta} = 0.4$. Similarly, for the test segment duration of 3 seconds, the tuned parameters ($\lambda_1' = 0.08$, $\lambda_1' = 0.01$, $\sqrt{\alpha} = 0.5$ and $\sqrt{\beta} = 0.4$) are selected to achieve better performances. On 1 second test segments, the best tuning parameters noted,

Table 6.2: Performances of the i-vector based LR employing bottleneck features and GMM posteriors. The scores are obtained using CDS classifier and are calibrated using Gaussian backend followed by MLR.

Session/	full-length		3 seconds		1 second	
	LD	LID	LD	LID	LD	LID
Channel Comp.	$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR
LDA	1.07	**96.84**	1.86	94.91	7.27	80.88
WCCN	**1.03**	96.80	**1.77**	**94.92**	**7.06**	**81.09**
LDA+WCCN	1.10	96.80	1.88	94.81	7.41	80.84

Table 6.3: Performances of the i-vector based LR employing bottleneck features and DNN posteriors. The scores are obtained using CDS classifier and are calibrated using Gaussian backend followed by MLR.

Session/	full-length		3 seconds		1 second	
	LD	LID	LD	LID	LD	LID
Channel Comp.	$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR
LDA	1.38	96.12	2.28	93.76	8.52	77.73
WCCN	**1.33**	**96.26**	**2.17**	**94.12**	**8.30**	**78.11**
LDA+WCCN	1.38	96.08	2.31	93.58	8.58	77.22

are $\lambda_1' = 0.005$, $\lambda_1' = 0.001$, $\sqrt{\alpha} = 0.1$ and $\sqrt{\beta} = 0.1$. The performances of LC-KSVD2 and D-KSVD based LR approaches are found comparatively.

6.8 Summary

In this chapter, we have explored sparse coding of bottleneck features based i-vector representation over three discriminatively learned dictionaries (learned-exemplar, D-KSVD and LC-KSVD) for LR systems. The performances of the proposed learned dictionaries based LR are found superior (or comparable in some cases) to the baseline system, except a slight degradation in case of learned-exemplar dictionary on full length segment. The performances of the D-KSVD and LC-KSVD have been found comparable, considering the same dictionary initialization procedure. The sufficient statistics required for i-vector estimation is computed using GMM/DNN posteriors. It is noted that MFCC based i-vector using DNN posteriors performs much better compared with GMM posteriors. However, in case of bottleneck features, GMM posteriors based i-vector representation gives satisfactory results. The effectiveness of PTN based LR system is also reported compared to TDNN and LSTM based LR systems. On comparison, the performance of bottleneck based LR system is found much superior than PTN based LR.

Table 6.4: Performances of the proposed LR systems employing i-vector based utterance representation derived from the bottleneck features. The sparse coding employing l_1-norm regularization with three different learned dictionaries: learned-exemplar (LD-XR), D-KSVD and LC-KSVD are considered. The scores are calibrated using Gaussian backend followed by MLR.

Pipeline used for i-vector extraction (features, statistics)	Dictionary	Classifier	full-length		3 seconds		1 second	
			LD	LID	LD	LID	LD	LID
			$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR	$100 \times C_{avg}$	% IDR
Contrast LR								
MFCC, GMM posteriors	–	CDS	3.08	90.80	4.74	86.45	12.19	67.74
MFCC, DNN posteriors			1.84	94.87	2.97	91.58	10.65	72.52
BNF, GMM posteriors			**1.03**	**96.80**	**1.77**	**94.92**	**7.06**	**81.09**
BNF, DNN Posteriors			1.33	96.26	2.17	94.12	8.30	78.11
Proposed LR								
BNF, GMM posteriors	LD-XR	SRC	1.25	96.29	1.77	94.95	7.04	80.67
	D-KSVD	$\mathbf{c_w = Ws}$	**0.96**	97.08	**1.72**	94.99	7.02	81.06
	LC-KSVD1		1.03	96.78	1.73	95.00	**6.99**	81.06
	LC-KSVD2		**0.96**	**97.16**	**1.72**	**95.14**	7.03	81.00

7

Conclusions

Contents

7.1 Summary of thesis

The objective of the work presented in this thesis is to explore the sparse representation techniques in LR task. The thesis begins with the review of existing LR approaches. It is followed by the description of the recently proposed sparse representation techniques over an exemplar dictionary which is created by concatenating the i-vector based utterance representation from all the language classes. The resultant dictionary happens to be an overcomplete dictionary as the number of training examples is more than i-vector dimensionality. In the proposed approach, the i-vector representation of the given test utterance is sparse coded over the created dictionary. The resulting s-vector (sparse vector) is obtained by solving the Lasso algorithm. On employing the SRC criterion on the s-vectors, the language detection/identification is performed. We first compare the performance of i-vector SRC based LR employing OMP and ENet regularization. The work is further extended with GMM mean supervector as utterance representation. It is noted that the exemplar dictionary created using GMM mean supervector happens to be undercomplete, as it is practically impossible to get the number of examples in the order of supervector dimensionality. The OMP, Lasso and ENet regularization based sparse coding problems are solved for the over-determined system. To mitigate the nuisances present in the utterance representations, suitable session/channel compensations techniques are applied. The LDA, WCCN, and LDA followed by WCCN compensation techniques are applied on the i-vector while JFA pre-processing is done on the GMM mean supervector. Using Lasso and ENet based regularization, the improved performances of WCCN compensated i-vector based LR is noted compared to JFA processed GMM mean supervector. However, the opposite trend is noted considering OMP based regularization. For contrast purposes, the state-of-the-art i-vector representation employing various classifiers like CDS, GG, SVM, MLR and G-PLDA are developed. The proposed SRC based approach using an exemplar dictionary is found slightly degraded compared to best performing i-vector employing GG classifier.

With the increasing size of the exemplar dictionary, the computational burden of sparse coding over such dictionary becomes quite prohibitive. To reduce the computational burden and to achieve effective sparse representation, we propose learned dictionary based sparse coding methods for LR task. We explore two types of dictionary learning approaches: the K-SVD and the online learning based dictionary. The K-SVD dictionary design requires the complete training data for processing, and hence results in slow convergence speed, especially in GMM mean supervector based utterance representation

with large training examples. To hasten the learning process, OL based dictionary is used which processes the training examples in batch mode. In this work, the sparse representation of training and test i-vector/GMM mean supervector is determined, which is used as a representation of speech utterances. The popular CDS classifier is investigated on the extracted s-vector. The experimental results show the comparable performance of K-SVD and OL dictionary based LR systems on both the i-vector and GMM mean supervector utterance representations. The performance of i-vector based learned dictionary is found superior to GMM mean supervector. It is also observed that the performance of learned dictionary based LR is better than the exemplar-based approach and comparable to i-vector employing GG classifier.

The limitation of the learned dictionary-based approach is that the class label of the learned dictionary atoms is no longer preserved. Hence the location of non-zero elements in the s-vector does not infer about the true language class of the test i-vector like done in case of the exemplar dictionary, i.e. SRC is not feasible. To address this issue, a class-based learned dictionary is created. For this purpose, the language-specific training i-vectors are used to create the language-specific learned dictionary. These language-specific dictionaries are then concatenated to form a learned-exemplar dictionary. Thus, applying sparse coding over class-based learned dictionary helps to determine the true class label of the test i-vector by knowing the non-zero locations in the s-vector. This method further reduces the computational complexity as it doesn't require any classifiers on s-vector. Furthermore, the sparse representation of i-vector over K-SVD and LC-KSVD discriminative dictionaries is also explored. The experimental results show the superior performance of LC-KSVD based discriminative dictionary based LR.

Despite the success of the i-vector based LR approach, the major drawback of the i-vector approach lies in its computational complexity and storage requirements. Motivated by these constraints, an ensemble of random subspaces of GMM mean supervector based LR approach which includes LDA/WCCN session/channel compensation in subspace is explored. The proposed approach does not require any learning for creating the subspaces and has very low storage requirements. On account of a limited number of languages involved, the complexity of the JFA based system is expected to be lower than that of i-vector based one. Exploiting this fact, the combination of the proposed ensemble approach with JFA is also explored. The results show that the extraction of JFA latent vector requires less runtime complexity than that of i-vector.

The ensemble of exemplar and learned-exemplar dictionary considering i-vector and JFA latent vector is also proposed. The improved performance of an ensemble of learned-exemplar with a reduced number of dictionaries is observed comparing ensemble of the exemplar dictionary. The reason for the improved performance is due to the extraction of efficient sparse representation over learned-exemplar and diversity gain obtained with multiple dictionaries. It is noted that the performance of an ensemble of learned-exemplar with low dimensional JFA-latent vector is better compared with contrast LR system; the i-vector employing GG classifier.

The recently proposed BNF based i-vector representation employing GMM posteriors is also explored in the sparse representation framework. In addition, the use of DNN posteriors is also made in the extraction of the i-vector. The sparse coding based LR system employing such i-vector representations is noted to outperform the existing LR approaches employing TDNN, LSTM and PTN modeling.

7.2 Salient contributions

The important contributions of the research work reported in this thesis are summarized below:

(i) Proposition of learned dictionary based sparse coding paradigm for LR task and its experimental validation on NIST LRE databases.

(ii) Exploitation of random subspace projection approach towards reducing the computational complexity of the front-end feature extraction in LR task.

(iii) Proposition of sparse coding-based LR approach employing an ensemble of learned dictionaries for further improving the classification performance through diversity gain.

(iv) Exploitation of recently proposed bottleneck feature based i-vector representations in sparse coding-based LR paradigm.

7.3 Directions for future work

In this section, we will provide some suggestions for further research.

(i) The proposed ensemble of the learned-exemplar dictionary may be extended to state-of-the-art bottleneck feature based i-vector representation.

(ii) The proposed SRC based approaches employing various regularization techniques may be investigated with prosodic features and can be fused with acoustic features to further improve the performance.

(iii) The block-structured learned dictionary is noted to enhance the reconstruction ability [203] as well as the classification ability [204]. Recently, correlation and class-based block formation for improved structured dictionary learning are explored [205] and found to give significant improvement in speaker verification task. Therefore, it would be nice to explore the block-structured learned dictionary employing bottleneck feature based i-vector for LR task.

A

Sparse coding algorithms: OMP and LARS

Contents

A.1 Orthogonal matching pursuit

The orthogonal matching pursuit (OMP) is an iterative greedy algorithm and at each step it selects a dictionary atom that is most correlated with the current residual. The approximate sparse solution $\hat{\mathbf{x}}$ corresponding to the target signal/vector $\mathbf{y}$ over an dictionary $\mathbf{D}$ can be obtained by OMP algorithm by solving the sparse coding problem given in Equation A.1. The sparse representation refers to the representation of a target signal using a linear combination of only few dictionary atoms. The sparse coding problem is formulated as

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq \gamma \tag{A.1}$$

where γ is the sparsity constraint (i.e., controls the number of non-zeros elements). The steps involved in the OMP algorithm are given in the Algorithm 1.

Algorithm 1 Orthogonal Matching Pursuit

Inputs: Dictionary $\mathbf{D} \equiv \{\mathbf{d}_i\}_{i=1}^{p}, \mathbf{d}_i \in \mathbb{R}^m$, the target vector $\mathbf{y} \in \mathbb{R}^m$ and γ is the number of non-zero elements in sparse solution.
Initialization: Initialize the iteration variable $k = 0$, and set
- the initial solution $\mathbf{x}^0 = \mathbf{0}$
- the initial residual $\mathbf{r}^0 = \mathbf{y} - \mathbf{D}\mathbf{x}^0 = \mathbf{y}$
- the initial solution support $\mathcal{S}^0 = Support\{\mathbf{x}^0\} = \emptyset$, the null set.

Main iteration:
for $k = 1, 2, 3, \ldots \gamma$, do

(i) **Element Selection:** Determine the dictionary column $i^\star$ that maximizes the correlation with the residual.

$$i^\star = \arg\max_{i=1,\ldots,p} \left\{ \| < \mathbf{r}^{k-1}, \mathbf{d}_i > \| \right\} \tag{A.2}$$

(ii) **Update Support:** $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{i^\star\}$

(iii) **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2$ subject to $Support\{\mathbf{x}\} = S^k$.

(iv) **Update Residual:**

$$\mathbf{r}^k = \mathbf{y} - \mathbf{D}\mathbf{x}^k \tag{A.3}$$

end for
Output: Solution $\hat{\mathbf{x}}$ is a vector having γ non-zero coefficients at the indices given by the support $\mathcal{S}^\gamma$.

A.2 Least angle regression

Consider a linear model $\mathbf{y} = \mathbf{D}\mathbf{x} + \epsilon$, where $\mathbf{y} \in \mathbb{R}^m$ is the observed response variable, $\mathbf{D} \equiv \{\mathbf{d}_i\}_{i=1}^p$ is the dictionary whose columns represents the *ith* predictor variable (or covariates) $\mathbf{d}_i \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^m$ is the set of model coefficients, and ϵ are the residual errors. In order to represent the response variable using a linear combination of only few potential covariates, well known least angle regression (LAR) [169] algorithm can be used. This can be achieved by estimating the model coefficients, the sparse vectors.

The steps involved in the LAR algorithm is given in the Algorithm 2.

Algorithm 2 Least Angle Regression

Inputs: Dictionary $\mathbf{D} \equiv \{\mathbf{d}_i\}_{i=1}^p, \mathbf{d}_i \in \mathbb{R}^m$, and the observed response variable $\mathbf{y} \in \mathbb{R}^m$. It is assumed that the vector $\mathbf{y}$ is centered and $\mathbf{D}$ is centered and normalized such that each variable has zero mean and unit Euclidean length.

Initialization:

- Initialize the coefficient vector $\mathbf{x}^{(0)} = \mathbf{0}$ and the fitted vector $\hat{\mathbf{y}}^{(0)} = \mathbf{0}$,
- Initialize the active set $\mathcal{A} = \emptyset$ and the inactive set $\mathcal{I} = \{1, 2, \ldots, p\}$

Main iteration:

for $k = 0$ to $p - 2$, **do**

(i) Update the residual $\epsilon = \mathbf{y} - \hat{\mathbf{y}}^{(k)}$

(ii) Find the maximal correlation $c = \max_{i \in \mathcal{I}} \mid \mathbf{d}_i' \epsilon \mid$

(iii) Move variable corresponding to c from $\mathcal{I}$ to $\mathcal{A}$

(iv) Calculate the least squares solution $\mathbf{x}_{\text{OLS}}^{(k+1)} = (\mathbf{D}_{\mathcal{A}}' \mathbf{D}_{\mathcal{A}})^{-1} \mathbf{D}_{\mathcal{A}}' \mathbf{y}$

(v) Calculate the current direction $\mathbf{d} = \mathbf{D}_{\mathcal{A}} \mathbf{x}_{\text{OLS}}^{(k+1)} - \hat{\mathbf{y}}^{(k)}$

(vi) Calculate the step length $\delta = \min_{i \in \mathcal{I}}{}^{+} \{\frac{\mathbf{d}_i' \epsilon - c}{\mathbf{d}_i' \mathbf{d} - c}, \frac{\mathbf{d}_i' \epsilon + c}{\mathbf{d}_i' \mathbf{d} + c}\}, \ 0 < \delta \leq 1$

(vii) Update regression coefficients $\mathbf{x}^{(k+1)} = (1 - \delta)\mathbf{x}^{(k)} + \delta \mathbf{x}_{\text{OLS}}^{(k+1)}$

(viii) Update the fitted vector $\hat{\mathbf{y}}^{(k+1)} = \hat{\mathbf{y}}^{(k)} + \delta \mathbf{d}$

end for

Compute $\mathbf{x}^{(p)}$, the full least squares solution given as $\mathbf{x}^{(p)} = (\mathbf{D}'\mathbf{D})^{-1}\mathbf{D}'\mathbf{y}$

Outputs:

Output the series of coefficients $X = [\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(p)}]$

A simple modification to least angle regression (LAR) solves the l_1-norm minimization based sparse

coding problem, also popularly known as the Lasso problem. The Lasso problem is formulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda_1\|\mathbf{x}\|_1 \tag{A.4}$$

where λ_1 is the positive regularization coefficient, which controls the number of nonzero coefficients in the solution $\hat{\mathbf{x}}$.

The following minor modification is added in order to determine the entire regularization path of optimal solutions to the Lasso problem: If a nonzero coefficient becomes zero, the corresponding variable is removed from the active set, and the current joint least-squares direction for LAR is recomputed over the remaining variables. The modified LAR for solving the Lasso problem is referred as LARS.

B

Dictionary learning algorithms: K-SVD, D-KSVD, LC-KSVD and Online

Contents

B.1 K-SVD dictionary learning

Given a set $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$ of training vectors and the sparsity constraint γ', the classical K-SVD dictionary learning problem can be defined as:

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{X}} \rangle = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \quad \text{such that} \quad \|\mathbf{x}_i\|_0 \leq \gamma' \quad \forall i \tag{B.1}$$

where $\mathbf{D}$ is the learned dictionary of K atoms, and $\mathbf{X} \equiv \{\mathbf{x}_i\}_{i=1}^N$ is the sparse matrix corresponding to $\mathbf{Y}$. The steps involved in the K-SVD algorithm are given in the Algorithm 3.

Algorithm 3 K-SVD Dictionary Learning

Inputs: Data Matrix $\mathbf{Y} \equiv \{\mathbf{y}_i\}_{i=1}^N$, Number of non-zero elements required in the sparse solutions (sparsity constraint), γ'.
Initialization: Initialize the dictionary, $\mathbf{D}^0 \in \mathbb{R}^{m \times K}$ with l_2 normalized columns. Set the iteration variable $J = 0$.
Main iteration: Increment J by 1 and perform the following steps

(i) **Sparse coding:** Solve the optimization problem given in Equation B.2 using any sparse coding algorithm (e.g., OMP) to determine the sparse codes $\mathbf{x}_i$ corresponding to each data vector $\mathbf{y}_i$, with a sparsity constraint γ'.

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 \text{ such that } \|\mathbf{x}_i\|_0 \leq \gamma' \quad \forall i \tag{B.2}$$

(ii) **Dictionary update:** For each column $k = 1, 2 \ldots, K$ in $\boldsymbol{D}^{(j-1)}$, update it by

- Define the group of examples that use this atom, $\omega_k = \{i | 1 \leq i \leq N, \mathbf{x}_T^k(i) \neq 0\}$.
- Compute the overall representation error matrix $\mathbf{E}_k$ by

$$\mathbf{E}_k = \mathbf{Y} - \sum_{j \neq k} \mathbf{d}_j \mathbf{x}_T^j \tag{B.3}$$

- Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k, and obtain $\mathbf{E}_k^R$.
- Apply SVD decomposition $\mathbf{E}_k^R = \mathbf{U}\boldsymbol{\Delta}\mathbf{V}^T$. Choose the updated dictionary column $\mathbf{d}_k$ to be the first column of $\mathbf{U}$. Update the coefficient vector $\mathbf{x}_R^k$ to be the first column of $\mathbf{V}$ multiplied by $\Delta(1,1)$

(iii) Set $J = J + 1$

The K-SVD [160] is an iterative algorithm having three stages: dictionary initialization, the sparse coding, and the dictionary update stages. In the dictionary initialization stage, initial dictionary is either created by randomly selecting training data vectors or random matrix. In the sparse coding stage, the data vectors are represented as linear combination of only few dictionary atoms. Usually the OMP algorithm is used for performing the sparse coding and the given sparsity constraint of error

threshold is used for stopping the OMP algorithm. In the dictionary update stage, the atoms of the dictionary are modified such that the overall representation error of the given set of data vectors is minimized.

B.2 D-KSVD dictionary learning

The discriminative-K-SVD (D-KSVD) [165] is a dictionary learning approach which incorporates classification error term in addition to reconstruction error used in classical K-SVD. The idea is to simultaneously learn a single dictionary $\hat{\mathbf{D}}$ and a linear classifier $\hat{\mathbf{W}}$ by solving the joint optimization problem which is formulated as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{W}}, \hat{\mathbf{X}} \rangle = \underset{\mathbf{D},\mathbf{W},\mathbf{X}}{\arg\min} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \beta \|\mathbf{H} - \mathbf{W}\mathbf{X}\|_F^2 \right\} \quad \text{such that} \quad \|\mathbf{x}_i\|_0 \leq \gamma' \ \ \forall i \tag{B.4}$$

where $\mathbf{Y}$ is the set of training data vectors, $\mathbf{D}$ is the learned dictionary, $\mathbf{X}$ is the set of sparse codes corresponding to $\mathbf{Y}$. β is the regularization parameter, the matrix $\mathbf{H}$ contains the class label of training data vectors, and $\mathbf{W}$ is a linear classifier. The steps involved in D-KSVD dictionary learning algorithm are given in Algorithm 4.

Algorithm 4 D-KSVD Dictionary Learning

Inputs: Data vectors $\mathbf{Y} \in \mathbb{R}^{m \times N}$, class-label matrix $\mathbf{H} \in \mathbb{R}^{L \times N}$, regularization parameter α and sparsity constraint γ'

Outputs: Dictionary $\mathbf{D} \in \mathbb{R}^{m \times K}$, linear classifier $\mathbf{W} \in \mathbb{R}^{L \times K}$ and sparse codes $\mathbf{X} \in \mathbb{R}^{K \times N}$

(i) **Initialize:**

- Compute $\mathbf{D}^{(0)}$ by concatenating language specific K-SVD learned dictionaries.
- Compute sparse codes $\mathbf{X}^{(0)}$ corresponding to train vector $\mathbf{Y}$ over initial learned dictionary $\mathbf{D}^{(0)}$ using sparse coding algorithm.
- Compute $\mathbf{W}^{(0)} = \mathbf{H}\mathbf{X}^{(0)'}(\mathbf{X}^{(0)}\mathbf{X}^{(0)'} + \mathbf{I})^{-1}$

(ii) **K-SVD:** For solving through classical K-SVD, the D-KSVD dictionary learning problem given in Equation B.4 can be rearranged as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{W}}, \hat{\mathbf{X}} \rangle = \underset{\mathbf{D},\mathbf{W},\mathbf{X}}{\arg\min} \left\| \begin{pmatrix} \mathbf{Y} \\ \sqrt{\beta}\mathbf{H} \end{pmatrix} - \begin{pmatrix} \mathbf{D} \\ \sqrt{\beta}\mathbf{W} \end{pmatrix} \mathbf{X} \right\|_F^2 \quad \text{such that} \quad \|\mathbf{x}_i\|_0 \leq \gamma' \ \ \forall i \tag{B.5}$$

Use initial dictionary $\begin{pmatrix} \mathbf{D}^{(0)} \\ \sqrt{\beta}\mathbf{W}^{(0)} \end{pmatrix}$.

(iii) **Normalize:** Normalize the columns of dictionary $\mathbf{D}$ and linear classifier $\mathbf{W}$ as

$\mathbf{D} \longleftarrow \left\{ \frac{\mathbf{d}_1}{\|\mathbf{d}_1\|_2}, \frac{\mathbf{d}_2}{\|\mathbf{d}_2\|_2}, \dots \frac{\mathbf{d}_K}{\|\mathbf{d}_K\|_2} \right\}$

$\mathbf{W} \longleftarrow \left\{ \frac{\mathbf{w}_1}{\|\mathbf{d}_1\|_2}, \frac{\mathbf{w}_2}{\|\mathbf{d}_2\|_2}, \dots \frac{\mathbf{w}_K}{\|\mathbf{d}_K\|_2} \right\}$

B.3 Label consistent-KSVD dictionary learning

In LC-KSVD [166] dictionary learning approach, discriminative sparse-code error term enforcing label consistency was incorporated in addition to the terms (i.e., reconstruction and classification error term) used in D-KSVD formulation. The LC-KSVD dictionary learning problem employing l_0 norm regularization is formulated as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{A}}, \hat{\mathbf{W}}, \hat{\mathbf{X}} \rangle = \underset{\mathbf{D},\mathbf{A},\mathbf{W},\mathbf{X}}{\arg\min} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \alpha \|\mathbf{Q} - \mathbf{A}\mathbf{X}\|_F^2 + \beta \|\mathbf{H} - \mathbf{W}\mathbf{X}\|_F^2 \right\} \quad \text{s.t.} \quad \|\mathbf{x}_i\|_0 \leq \gamma' \ \ \forall i \tag{B.6}$$

where $\mathbf{Y} \in \mathbb{R}^{m \times N}$ is the set of training data vectors, $\mathbf{D} \in \mathbb{R}^{m \times K}$ is the learned dictionary, $\mathbf{X} \in \mathbb{R}^{K \times N}$ is the set of sparse codes corresponding to $\mathbf{Y}$, matrix $\mathbf{H} \in \mathbb{R}^{L \times N}$ contains the class label of training data vectors, and $\mathbf{W} \in \mathbb{R}^{L \times K}$ is a linear classifier. The matrix $\mathbf{Q} \in \mathbb{R}^{K \times N}$ is the discriminative sparse codes matrix promoting label consistency, $\mathbf{A} \in \mathbb{R}^{K \times K}$ is a linear transformation, α and β are the regularization parameters used to balance the discriminative sparse code errors and classification error to overall objective function, respectively. The matrix $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N] \in R^{K \times N}$ is the discriminative sparse codes of input signals $\mathbf{Y}$ for classification.

We say that $\mathbf{q}_i = [\mathbf{q}_i^1, \mathbf{q}_i^2, \ldots, \mathbf{q}_i^K]^T = [0, 0, \ldots, 1, 1, \ldots, 0, 0]^T \in R^K$ is a discriminative sparse code corresponding to an input signal $\mathbf{y}_i$ if the nonzero values of $\mathbf{q}_i$ occur at those indices where the input signal $\mathbf{y}_i$ and the dictionary atom $\mathbf{d}_k$ share the same label. For example, assuming $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_6]$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_6]$ where $\mathbf{y}_1, \mathbf{y}_2, \mathbf{d}_1, \mathbf{d}_2$ are from class 1; $\mathbf{y}_3, \mathbf{y}_4, \mathbf{d}_3, \mathbf{d}_4$ are from class 2; and $\mathbf{y}_5, \mathbf{y}_6, \mathbf{d}_5, \mathbf{d}_6$ are from class 3. The matrix Q can be defined as

$$\mathbf{Q} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

where each column corresponds to a discriminative sparse code for an input signal. The term $\|\mathbf{Q} - \mathbf{A}\mathbf{S}\|_F^2$ represents the discriminative sparse code error, which enforces that the transformed sparse codes $\mathbf{AS}$ approximate the discriminative sparse codes $\mathbf{Q}$. It forces the signals from the same class to have very similar sparse representations (i.e., promoting label consistency in the resulting sparse codes).

The special case of LC-KSVD problem with $\beta = 0$ is referred as LC-KSVD1 while with non-zeros values of regularization parameters α and β, it is referred as LC-KSVD2. The learning procedure of LC-KSVD1 is same as of LC-KSVD2, however the classifier $\mathbf{W}$ for LC-KSVD1 is trained separately after the computation of $\mathbf{D}$, $\mathbf{A}$, and $\mathbf{X}$. The estimate of linear classifier $\hat{\mathbf{W}}$ in case of LC-KSVD1 is obtained by solving the equation

$$\hat{\mathbf{W}} = \mathbf{H}\hat{\mathbf{X}}^{'}(\hat{\mathbf{X}}\hat{\mathbf{X}}^{'} + \lambda\mathbf{I})^{-1} \tag{B.7}$$

where λ is the l_2-norm regularization coefficient and $\mathbf{I}$ is the identity matrix. The steps involved in LC-KSVD dictionary learning are given in Algorithm 5.

Algorithm 5 Label Consistent K-SVD Dictionary Learning

Inputs: Data vectors $\mathbf{Y} \in \mathbb{R}^{m\times N}$, class-label matrix $\mathbf{H} \in \mathbb{R}^{p\times N}$, discriminative sparse code matrix $\mathbf{Q} \in \mathbb{R}^{K\times N}$, regularization parameters α and β, sparsity constraint γ'
Outputs: Dictionary $\mathbf{D} \in \mathbb{R}^{m\times K}$, linear transformation matrix $\mathbf{A}$, linear classifier $\mathbf{W} \in \mathbb{R}^{p\times K}$ and sparse codes $\mathbf{X} \in \mathbb{R}^{K\times N}$

(i) **Initialize:**

- Compute $\mathbf{D}^{(0)}$ by concatenating language specific K-SVD learned dictionaries.
- Compute sparse codes $\mathbf{X}^{(0)}$ corresponding to train vector $\mathbf{Y}$ over initial learned dictionary $\mathbf{D}^{(0)}$ using sparse coding algorithm.
- Compute $\mathbf{A}^{(0)} = \mathbf{Q}\mathbf{X}^{(0)^{'}}(\mathbf{X}^{(0)}\mathbf{X}^{(0)^{'}} + \mathbf{I})^{-1}$
- Compute $\mathbf{W}^{(0)} = \mathbf{H}\mathbf{X}^{(0)^{'}}(\mathbf{X}^{(0)}\mathbf{X}^{(0)^{'}} + \mathbf{I})^{-1}$

(ii) **K-SVD:** For solving through classical K-SVD, the LC-KSVD dictionary learning problem can be rearranged as

$$\langle \hat{\mathbf{D}}, \hat{\mathbf{A}}, \hat{\mathbf{W}}, \hat{\mathbf{X}} \rangle = \underset{\mathbf{D},\mathbf{A},\mathbf{W},\mathbf{X}}{\arg\min} \left\| \begin{pmatrix} \mathbf{Y} \\ \sqrt{\alpha}\mathbf{Q} \\ \sqrt{\beta}\mathbf{H} \end{pmatrix} - \begin{pmatrix} \mathbf{D} \\ \sqrt{\alpha}\mathbf{A} \\ \sqrt{\beta}\mathbf{W} \end{pmatrix} \mathbf{X} \right\|_F^2 \quad \text{s.t.} \quad \|\mathbf{x}_i\|_0 \leq \gamma' \ \ \forall i \tag{B.8}$$

Use initial dictionary $\begin{pmatrix} \mathbf{D}^{(0)} \\ \sqrt{\alpha}\mathbf{A}^{(0)} \\ \sqrt{\beta}\mathbf{W}^{(0)} \end{pmatrix}$.

(iii) **Normalize:** Normalize the columns of dictionary $\mathbf{D}$, linear transformation matrix $\mathbf{A}$, and linear classifier $\mathbf{W}$ as

$\mathbf{D} \longleftarrow \left\{ \frac{\mathbf{d}_1}{\|\mathbf{d}_1\|_2}, \frac{\mathbf{d}_2}{\|\mathbf{d}_2\|_2}, \ldots \frac{\mathbf{d}_K}{\|\mathbf{d}_K\|_2} \right\}$

$\mathbf{A} \longleftarrow \left\{ \frac{\mathbf{a}_1}{\|\mathbf{d}_1\|_2}, \frac{\mathbf{a}_2}{\|\mathbf{d}_2\|_2}, \ldots \frac{\mathbf{a}_K}{\|\mathbf{d}_K\|_2} \right\}$

$\mathbf{W} \longleftarrow \left\{ \frac{\mathbf{w}_1}{\|\mathbf{d}_1\|_2}, \frac{\mathbf{w}_2}{\|\mathbf{d}_2\|_2}, \ldots \frac{\mathbf{w}_K}{\|\mathbf{d}_K\|_2} \right\}$

B.4 Online dictionary learning

The online dictionary learning [159] approach requires one element (or a small subset) of the training set at a time. On the other hand, batch mode based learning approach (e.g., K-SVD) access whole training sets at each iteration. Thus, online dictionary learning approach can efficiently handle large training set or dynamic training data changing over time.

The steps involved in online dictionary learning using one element of the training set are summarized in Algorithm 6. The dictionary update stage of the online learning procedure is described in Algorithm 7.

Algorithm 6 Online dictionary learning

Inputs: $\mathbf{y} \in \mathbb{R}^m \sim p(y)$ (Assuming the training set composed of i.i.d samples of a distribution $p(y)$), initial dictionary $\mathbf{D}_0 \in \mathbb{R}^{m \times K}$, Number of iterations T, Regularization parameter $\lambda_1 \in \mathbb{R}$.

(i) Matrices $\mathbf{A}_0 = \mathbf{0}$ and $\mathbf{B}_0 = \mathbf{0}$ (reset the past information)

(ii) **for** $t = 1$ to T, **do**

(iii) Draw $\mathbf{y}_t$

(iv) Sparse coding (Compute using LARS):

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y}_t - \mathbf{D}_{t-1}\mathbf{x}\|_2^2 + \lambda_1\|\mathbf{x}\|_1 \tag{B.9}$$

(v) $\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{x}_t\mathbf{x}_t'$

(vi) $\mathbf{B}_t = \mathbf{B}_{t-1} + \mathbf{y}_t\mathbf{x}_t'$

(vii) Dictionary update (Use Algorithm 7, the block coordinate descent with $\mathbf{D}_{t-1}$ as warm restart):

$$\mathbf{D}_t = \arg\min_{\mathbf{D}} \frac{1}{t}\sum_{i=1}^{t} \frac{1}{2}\|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 + \lambda_1\|\mathbf{x}_i\|_1 \tag{B.10}$$

(viii) **end for**

(ix) **Output:** learned dictionary $\mathbf{D}_T$.

In practice, the convergence speed of the online algorithm can be increased by drawing more than one training examples (i.e., using mini-batch approach) at each iteration. Let us denote $\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,P}$,

Algorithm 7 Dictionary Update

Inputs: Input Dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k] \in \mathbb{R}^{m \times K}$,
$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_k] \in \mathbb{R}^{K \times K} = \sum_{i=1}^{t} \mathbf{x}_i \mathbf{x}_i'$
$\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k] \in \mathbb{R}^{m \times K} = \sum_{i=1}^{t} \mathbf{y}_i \mathbf{x}_i'$

(i) **repeat**

(ii) **for** $j = 1$ to K, **do**

(iii) update the j-th column to optimize the Equation B.10

$$\mathbf{u}_j = \frac{1}{\mathbf{A}_{jj}}(\mathbf{b}_j - \mathbf{D}\mathbf{a}_j) + \mathbf{d}_j \tag{B.11}$$

$$\mathbf{d}_j = \frac{1}{\max(\|\mathbf{u}_j\|, 1)}\mathbf{u}_j \tag{B.12}$$

(iv) **end for**

(v) **until convergence**

(vi) **Output:** Updated dictionary $\mathbf{D}$.

the training examples drawn at iteration t. Then, replace the lines (v) and (vi) of Algorithm 6 by

$$\mathbf{A}_t = \beta \mathbf{A}_{t-1} + \sum_{i=1}^{P} \mathbf{x}_{t,i} \mathbf{x}_{t,i}' \tag{B.13}$$

$$\mathbf{B}_t = \beta \mathbf{B}_{t-1} + \sum_{i=1}^{P} \mathbf{y}_{t,i} \mathbf{x}_{t,i}' \tag{B.14}$$

where $\beta = \frac{\theta+1-\eta}{\theta+1}$. If $t < P$ then $\theta = tP$ elseif $t \geq P$ then $\theta = P^2 + t - P$.

C

Parameters Tuning in K-SVD/OL Learned Dictionary based LR

Contents

C.1 Parameters tuning in learned dictionary based LR with OMP and Lasso

The optimal performance of LR using OMP/Lasso-based sparse representation over K-SVD/OL learned dictionary mainly depends upon the three parameters: (i) size of the dictionary (ii) the number of atoms selected while learning the dictionary (dictionary sparsity) (iii) the number of atoms selected while representing the target data (decomposition sparsity). The parameters are tuned to obtain the best results in terms of C_{avg} and IDR. Figure C.1 shows the performance of LR using OMP based sparse coding of WCCN compensated i-vector over K-SVD learned dictionary with tuning parameters: dictionary size, dictionary sparsity (γ') and decomposition sparsity (γ). The CDS classifier is applied on the training and test s-vectors. The score calibration is done by GB+MLR technique employing multi-class FoCal toolkit [153] prior to final decision. It is observed that the best performance is achieved with a dictionary size of 28, $\gamma' = 20$ and $\gamma = 15$ for language detection while in language identification, the optimal parameters noted are $\gamma' = 20$ and $\gamma = 20$. The performance of the LR system using Lasso-based sparse coding of WCCN compensated i-vector over K-SVD learned dictionary is shown in Figure C.2. With fixed dictionary size of 28, the best performance in terms of C_{avg} for language detection is noted with dictionary sparsity $\lambda_1' = 0.01$ and decomposition sparsity $\lambda_1 = 0.001$ while in language identification, the optimal parameters noted are $\lambda_1' = 0.02$ and $\lambda_1 = 0.001$.

In addition to i-vector utterance representation, sparse representation of JFA compensated GMM mean supervector over K-SVD learned dictionary employing OMP/Lasso sparse coding algorithms are investigated and results are shown in the Figures C.3 and C.4. It can be observed that using OMP, the best performance is achieved with a dictionary size of 28, dictionary sparsity $\gamma' = 20$, decomposition sparsity $\gamma = 15$ while using Lasso, the optimum performance is achieved with dictionary sparsity $\lambda_1' = 0.05$, decomposition sparsity $\lambda_1 = 0.001$ and dictionary size of 28. On comparing with OMP, a Lasso-based sparse coding algorithm provides slightly better performance.

The work is further extended with OL dictionary employing OMP and Lasso based sparse coding considering both WCCN compensated i-vector and JFA-supervector. The results are reported in Figures C.5, C.6, C.7 and C.8. Using OL employing OMP and the i-vector representation, the best language detection performance is achieved with dictionary sparsity $\gamma' = 15$ and decomposition sparsity $\gamma = 20$ shown in Figure C.5 (a) while for language identification task, the sparsity constraints $\gamma' = 20$ and $\gamma = 20$ gives best result as shown in Figure C.5 (b). However, using OL dictionary employing Lasso

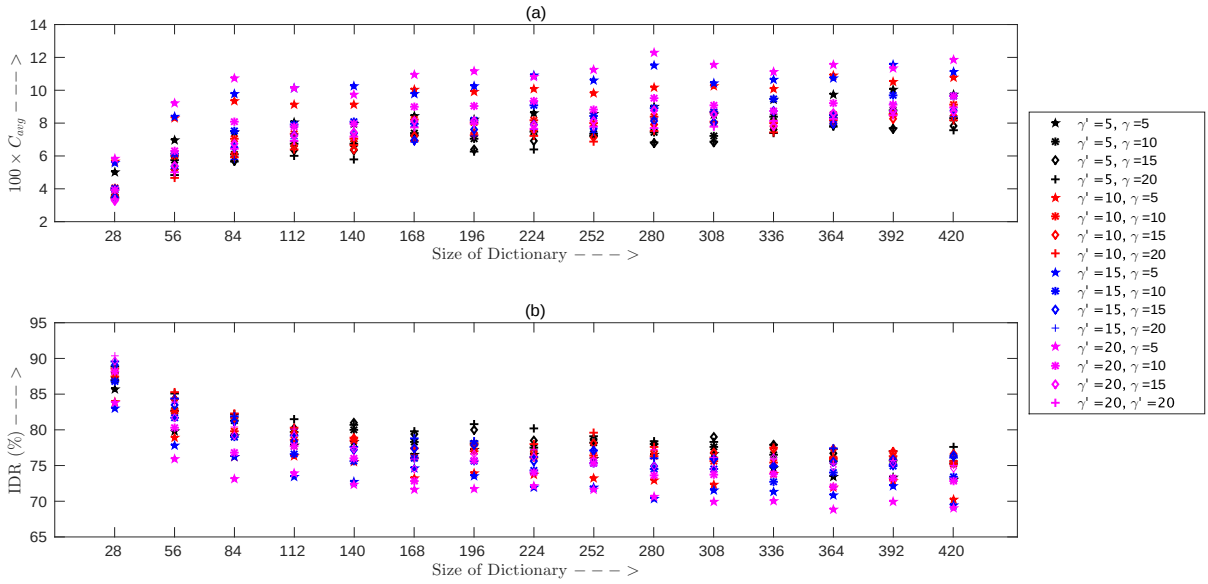

Figure C.1: Performance of the LR system using the sparse representation of WCCN compensated i-vector over K-SVD learned dictionary with the tuning of dictionary size, dictionary sparsity (γ') and decomposition sparsity (γ).The sparse coding is performed using the OMP algorithm. (a) C_{avg} (b) IDR

.

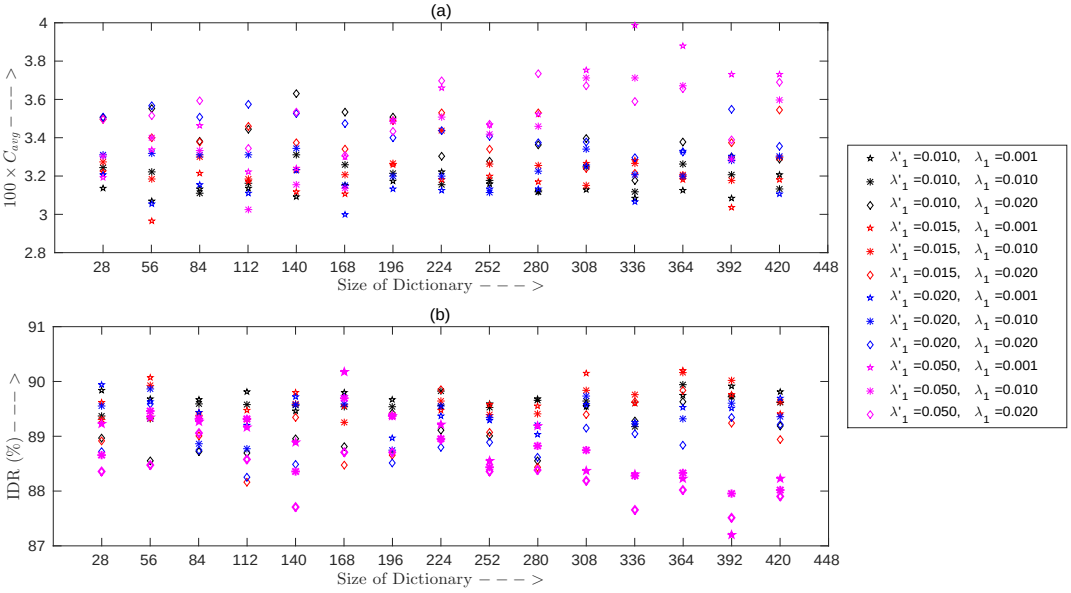

Figure C.2: Performance of the LR system using the sparse representation of WCCN compensated i-vector over K-SVD learned dictionary with tuning of dictionary size, dictionary sparsity (λ'_1) and decomposition sparsity (λ_1). The sparse coding is performed using Lasso algorithm. (a) C_{avg} (b) IDR

.

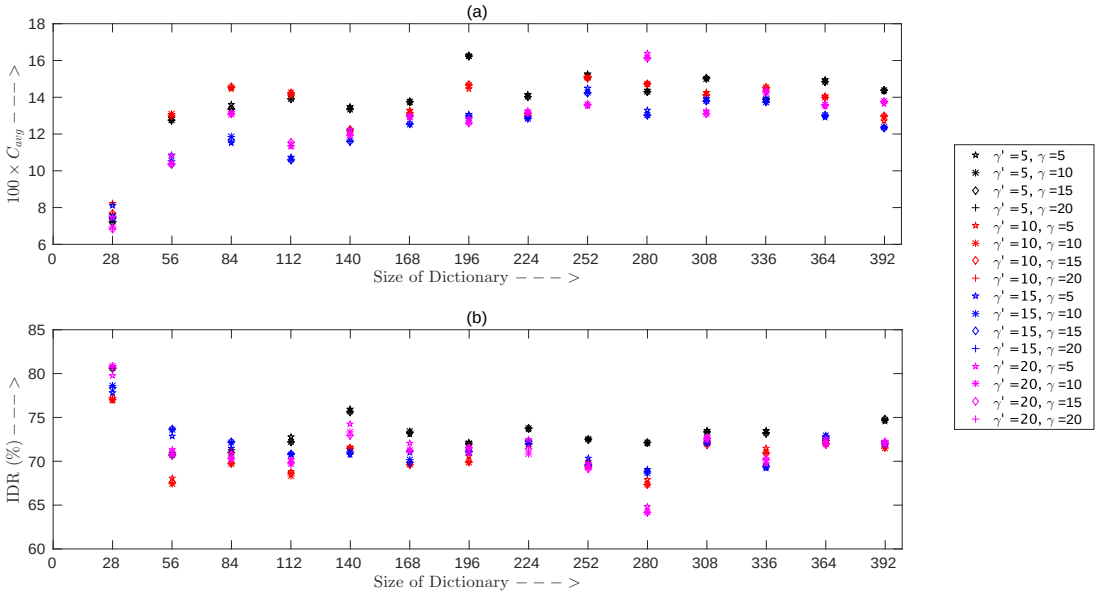

Figure C.3: Performance of the LR system using the sparse representation of JFA-supervector over K-SVD learned dictionary with tuning of dictionary size, dictionary sparsity (γ') and decomposition sparsity (γ). The sparse coding is performed using OMP algorithm. (a) C_{avg} (b) IDR

.

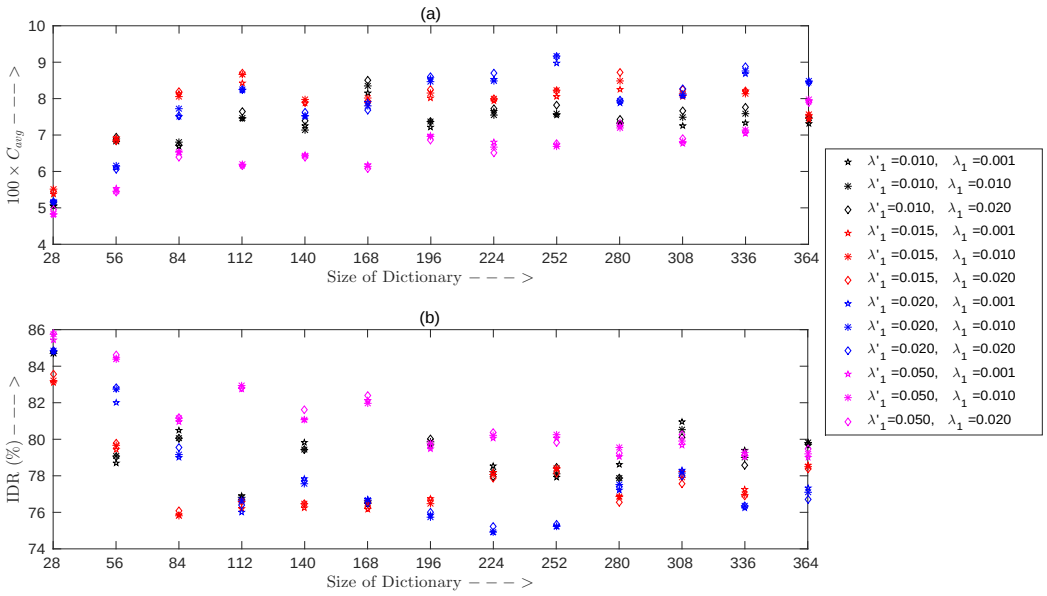

Figure C.4: Performance of the LR system using the sparse representation of JFA-supervector over K-SVD learned dictionary with tuning of dictionary size, dictionary sparsity (λ'_1) and decomposition sparsity (λ_1). The Lasso based sparse coding is performed. (a) C_{avg} (b) IDR

.

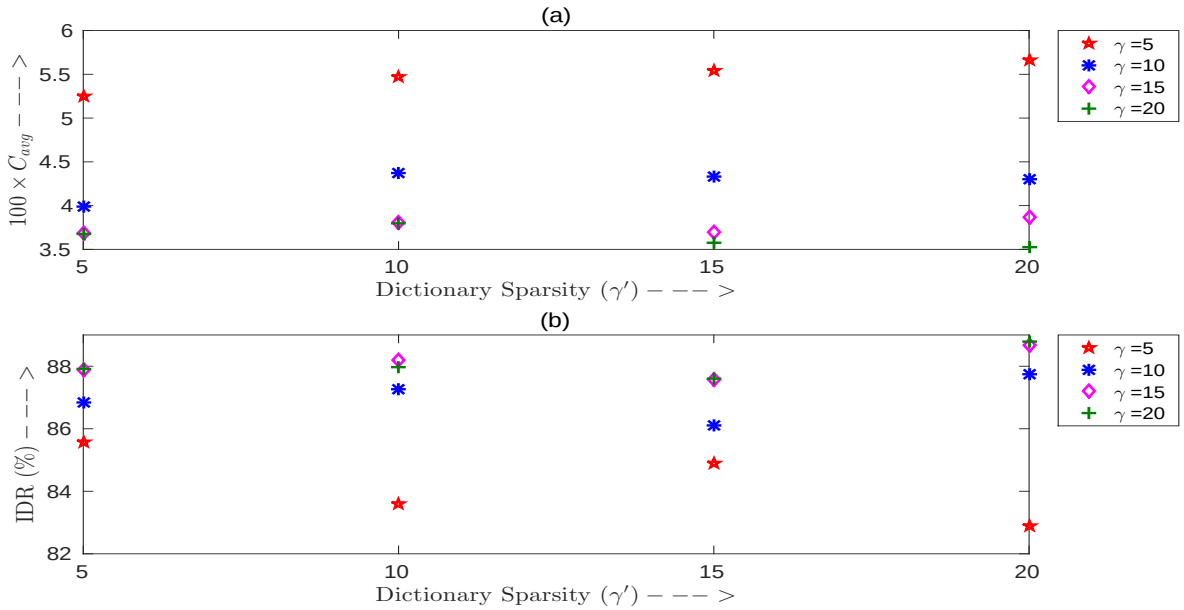

Figure C.5: Performance of the LR system using the sparse representation of WCCN compensated i-vector over OL learned dictionary (28 dictionary atoms) with tuning of dictionary sparsity (γ') and decomposition sparsity (γ). The sparse coding is performed using OMP algorithm. (a) C_{avg} (b) IDR

.

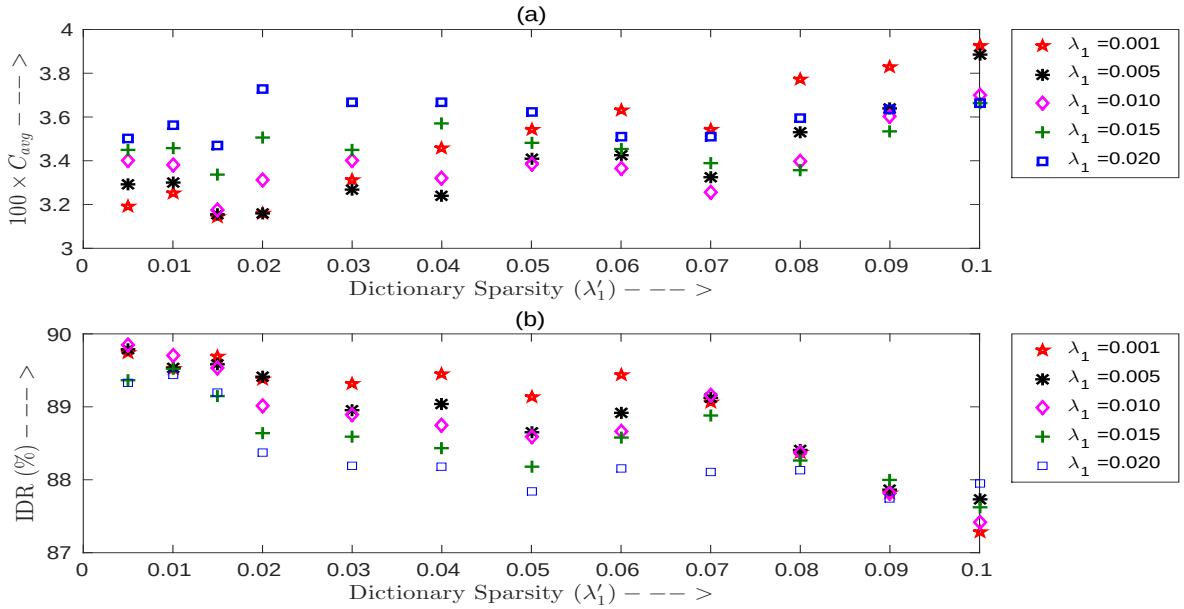

Figure C.6: Performance of the LR system using the sparse representation of WCCN compensated i-vector over OL learned dictionary (28 dictionary atoms) with tuning of dictionary sparsity (λ_1') and decomposition sparsity coefficient (λ_1). The Lasso based sparse coding is performed. (a) C_{avg} (b) IDR

.

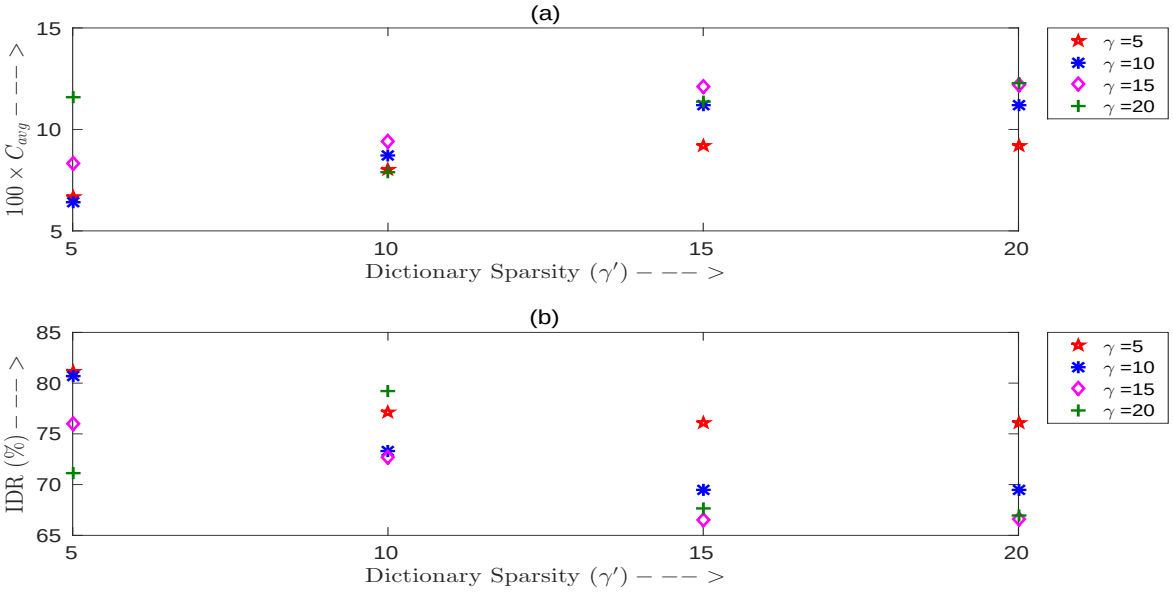

Figure C.7: Performance of the LR system using the sparse representation of JFA-supervector over OL learned dictionary (28 dictionary atoms) with tuning of dictionary sparsity (γ') and decomposition sparsity (γ). The sparse coding is performed using OMP algorithm. (a) C_{avg} (b) IDR

.

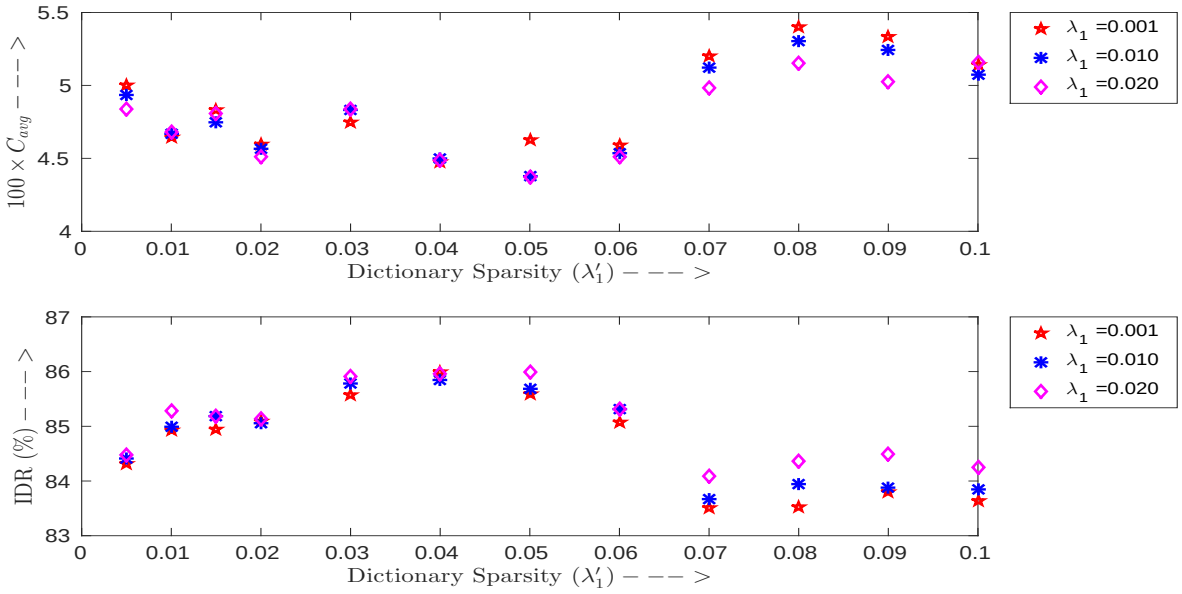

Figure C.8: Performance of the LR system using the sparse representation of JFA-supervector over OL learned dictionary (28 dictionary atoms) with tuning of dictionary sparsity (λ_1') and decomposition sparsity (λ_1). The Lasso based sparse coding is performed. (a) C_{avg} (b) IDR

.

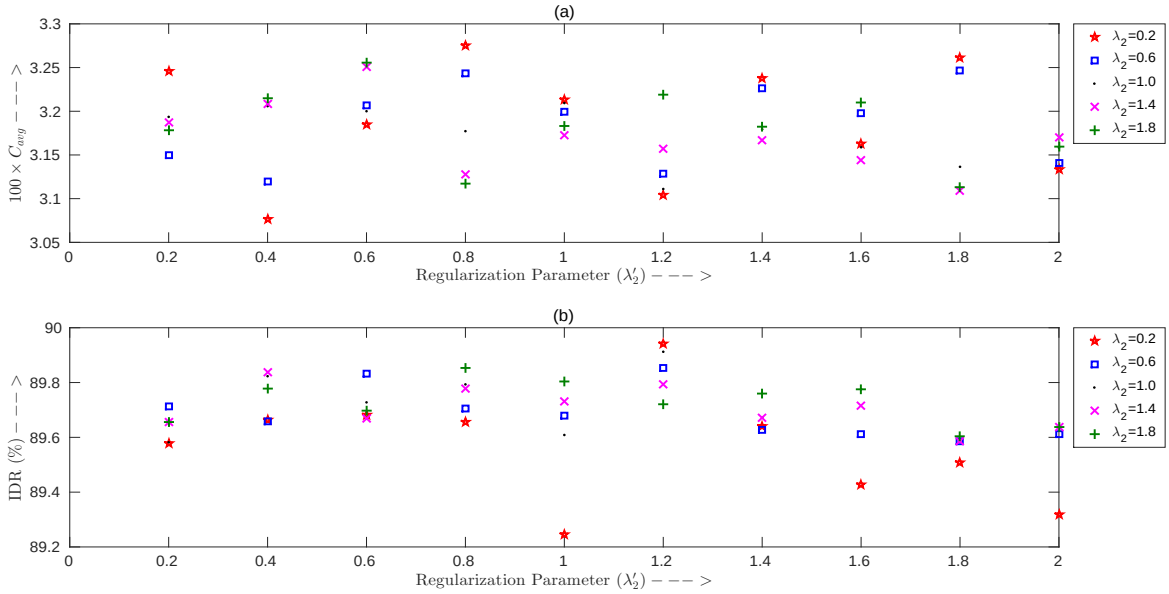

Figure C.9: Performance of the LR system using the sparse representation of WCCN compensated i-vector over K-SVD learned dictionary (28 dictionary atoms) with tuning of regularization coefficient (λ_2') in dictionary learning and regularization coefficient (λ_2) in s-vector extraction for fixed $\lambda_1' = 0.01, \lambda_1 = 0.001$. The ENet based sparse coding is performed. (a) C_{avg} (b) IDR

.

and the i-vector representation, the best language detection performance is achieved with dictionary sparsity $\lambda_1' = 0.015$ and decomposition sparsity $\lambda_1 = 0.001$ as shown in Figure C.6 (a) while for language identification task, the sparsity constraints $\lambda_1' = 0.005$ and $\lambda_1 = 0.01$ gives best result as shown in Figure C.6 (b).

Figures C.7 and C.8 show the performances of LR using OL employing OMP and Lasso, respectively. The JFA-supervector is used as the utterance representation. Using OMP, the value of dictionary sparsity $\gamma' = 5$ and decomposition sparsity $\gamma = 5$ gives best performance both in language detection and language identification as shown in Figure C.7. In Lasso based sparse coding, the best performances in language detection as well language identification are noted with dictionary sparsity $\lambda_1' = 0.05$ and decomposition sparsity $\lambda_1 = 0.02$ as shown in Figure C.8.

C.2 Parameters tuning in learned dictionary based LR with ENet

The sparse representation over KSVD/OL learned dictionary based LR with ENet sparse coding algorithm requires tuning of five parameters:(i) size of dictionary (ii) regularization coefficients λ_1' and λ_2' in dictionary learning (iii) regularization coefficients λ_1 and λ_2 in extracting sparse representation

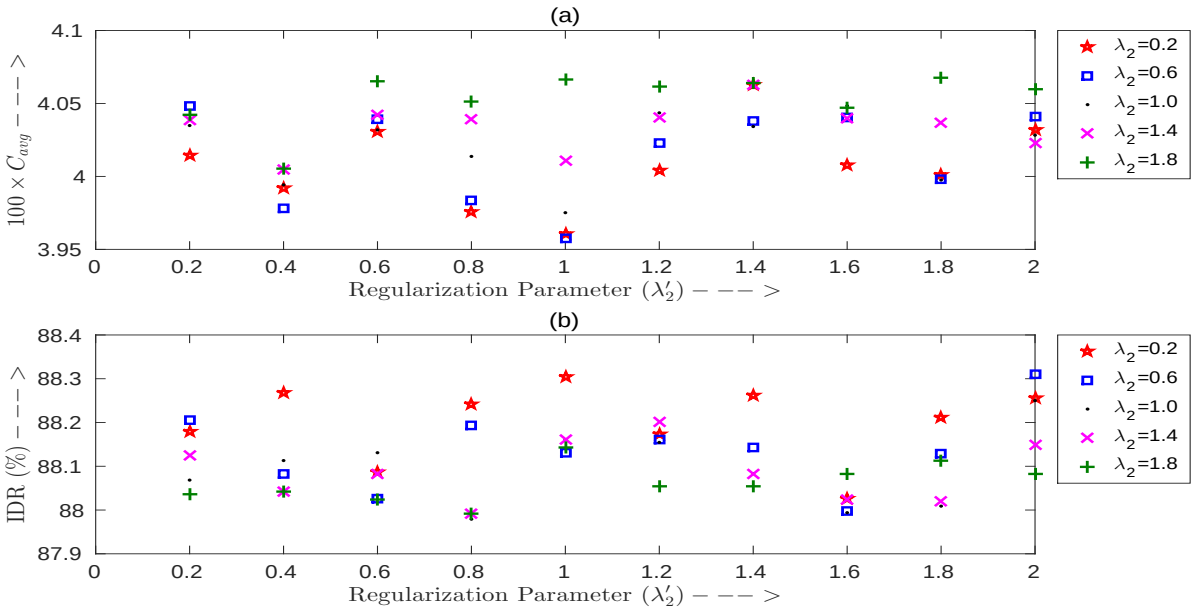

Figure C.10: Performance of the LR system using the sparse representation of JFA-supervector over K-SVD learned dictionary (28 dictionary atoms) with tuning of regularization coefficient (λ_2') in dictionary learning and regularization coefficient (λ_2) in s-vector extraction for fixed $\lambda_1' = 0.05, \lambda_1 = 0.001$. The ENet based sparse coding is performed. (a) C_{avg} (b) IDR

over learned dictionary. The regularization coefficients λ_1' and λ_1 are the dictionary sparsity and decomposition sparsity, respectively. The coefficients λ_2' and λ_2 encourages the grouped selection and stabilizes the solution paths with respect to random sampling in dictionary learning and sparse representation over the learned dictionary, respectively. In ENet sparse coding, the values of dictionary sparsity and decomposition sparsity are kept same as that of Lasso to maintain the same sparsity and selected based on the best performance achieved by the Lasso. In the previous subsection, it was also noted that only 28 dictionary atoms are sufficient to obtain the good performance. Thus, tuning of regularization coefficients λ_2' and λ_2 are done keeping fixed dictionary size, λ_1' and λ_1 to obtain the best performance using learned dictionary employing ENet.

Figures C.9 and C.10 show the performance of KSVD learned dictionary based LR with two utterance representation: WCCN compensated i-vector and JFA-supervector, respectively, for various tuning parameters. In Figure C.9 (a), the best language detection performance is achieved with $\lambda_2' = 0.4$ and $\lambda_2 = 0.2$ while for language identification task, the parameters $\lambda_2' = 1.2$ and $\lambda_2 = 0.2$ gives best result as shown in Figure C.9 (b). The tuning of parameters are done by keeping fixed $\lambda_1' = 0.01$ and $\lambda_1 = 0.001$. However, in Figure C.10 (a), the best language detection performance is

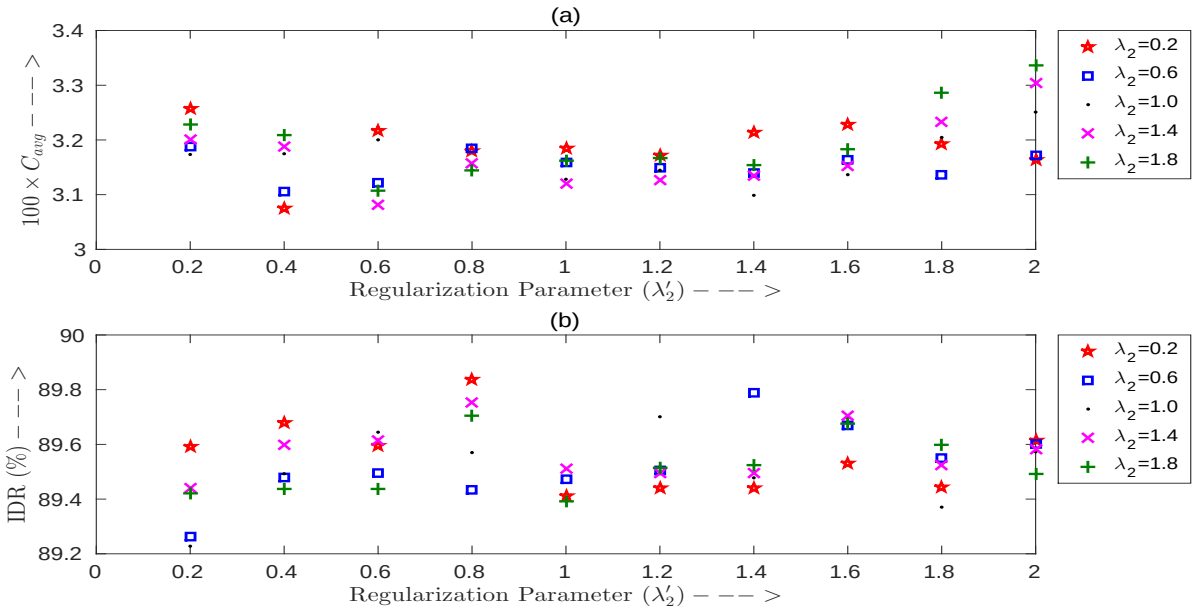

Figure C.11: Performance of the LR system using the sparse representation of WCCN compensated i-vector over OL learned dictionary (28 dictionary atoms) with tuning of regularization coefficient (λ_2') in dictionary learning and regularization coefficient (λ_2) in s-vector extraction for fixed $\lambda_1' = 0.015, \lambda_1 = 0.001$. The ENet based sparse coding is performed. (a) C_{avg} (b) IDR

achieved with $\lambda_2' = 1$ and $\lambda_2 = 0.6$ while for language identification task, the parameters $\lambda_2' = 2$ and $\lambda_2 = 0.6$ gives best result as shown in Figure C.10 (b). The tuning of parameters are done by keeping fixed $\lambda_1' = 0.05$ and $\lambda_1 = 0.001$.

We have also explored the OL dictionary employing ENet sparse coding with i-vector and GMM mean supervector utterance representation and the results are reported in Figures C.11 and C.12. Considering i-vector representation, the best language detection performance is achieved with $\lambda_2' = 0.4$ and $\lambda_2 = 0.2$ which is shown in Figure C.11 (a) while for language identification the best performance is achieved with $\lambda_2' = 0.8$ and $\lambda_2 = 0.2$ and is shown in Figure C.11(b). Both the performances are obtained by keeping fixed regularization coefficients $\lambda_1' = 0.015$ and $\lambda_1 = 0.001$. However, with JFA-supervector, the best language detection performance is achieved with $\lambda_2' = 2$ and $\lambda_2 = 0.2$ as shown in Figure C.12 (a) while for language identification task, the parameters $\lambda_2' = 0.8$ and $\lambda_2 = 1.8$ gives best result which is shown in Figure C.12 (b). In both the cases, the tuning of parameters are done by keeping fixed $\lambda_1' = 0.05$ and $\lambda_1 = 0.02$.

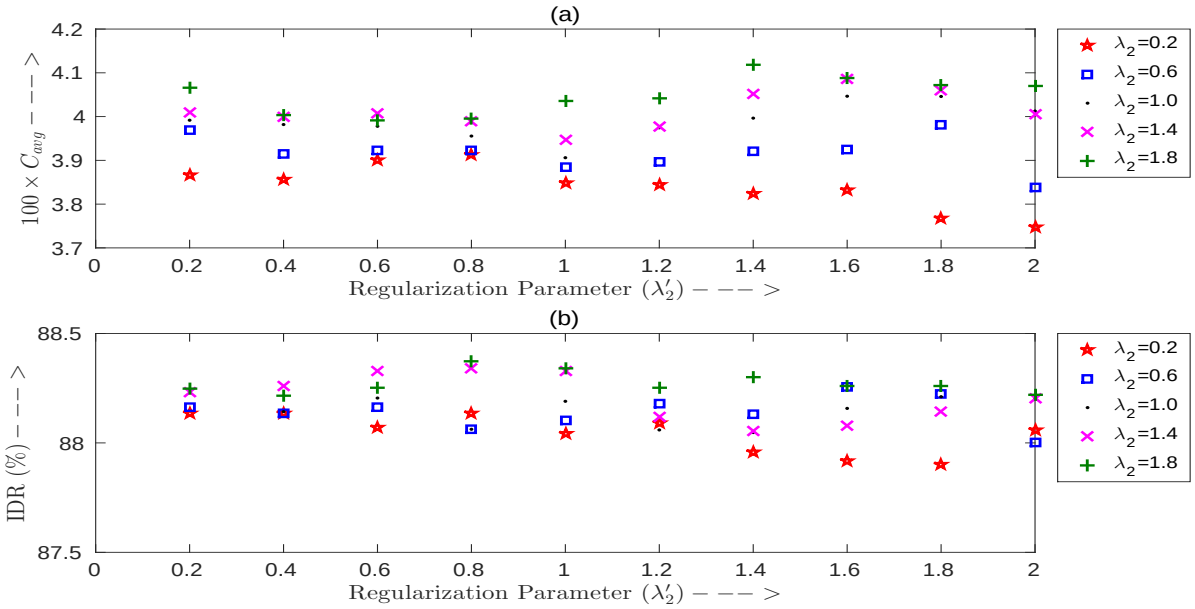

Figure C.12: Performance of the LR system using the sparse representation of JFA-supervector over OL learned dictionary (28 dictionary atoms) with tuning of regularization coefficient (λ_2') in dictionary learning and regularization coefficient (λ_2) in s-vector extraction for fixed $\lambda_1' = 0.05, \lambda_1 = 0.02$. The ENet based sparse coding is performed. (a) C_{avg} (b) IDR

Bibliography

[1] H. Li, B. Ma, and K. A. Lee, "Spoken language recognition: from fundamentals to practice," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1136–1159, 2013.

[2] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[3] M. Elad, *Sparse and redundant representations: From theory to applications in signal and image processing.* New York: Springer, 2010.

[4] Z. Tang, D. Wang, Y. Chen, L. Li, and A. Abel, "Phonetic temporal neural model for language identification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 134–144, 2018.

[5] J. Zhao, H. Shu, L. Zhang, X. Wang, Q. Gong, and P. Li, "Cortical competition during language discrimination," *NeuroImage*, vol. 43, no. 3, pp. 624–633, 2008.

[6] V. Fromkin, R. Rodman, and N. Hyams, *An Introduction to Language (9th ed.).* Wadsworth Cengage Learning, 2011.

[7] Y. K. Muthusamy, N. Jain, and R. A. Cole, "Perceptual benchmarks for automatic language identification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1994, pp. 333–336.

[8] E. Ambikairajah, H. Li, L. Wang, B. Yin, and V. Sethu, "Language identification: A tutorial," *IEEE Circuits and Systems Magazine*, vol. 11, no. 2, pp. 82–108, 2011.

[9] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, Aug 1980.

[10] B. Bielefeld, "Language identification using shifted delta cepstrum," in *Proc. 14th Annual Speech Research Symposium*, 1994.

[11] M. A. Kohler and M. Kennedy, "Language identification using shifted delta cepstra," in *Proc. Midwest Symposium on Circuits and Systems(MWSCAS)*, vol. 3, 2002, pp. 69–72.

[12] R. G. Leonard and G. R. Doddington, "Automatic language identification," Technical Report RADC-TR-74-200, Air Force Rome Air Development, Tech. Rep., Aug. 1974.

[13] ——, "Automatic language identification," Technical Report RADC-TR-75-264, Air Force Rome Air Development Center, Tech. Rep., Oct. 1975.

[14] ——, "Automatic language discrimination," Technical Report RADC-TR-78-5, Air Force Rome Air Development Center, Tech. Rep., Jan. 1978.

[15] R. G. Leonard, "Language recognition test and evaluation," Technical Report RADC-TR-80-83, Air Force Rome Air Development Center, Tech. Rep., March 1980.

[16] A. S. House and E. P. Neuburg, "Toward automatic identification of the language of an utterance. I. preliminary methodological considerations," *The Journal of the Acoustical Society of America*, vol. 62, no. 3, pp. 708–713, 1977.

[17] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE Acoustics, Speech, and Signal Processing Magazine (ASSP-M)*, vol. 3, no. 1, pp. 4–16, 1986.

[18] K. Li and T. Edwards, "Statistical models for automatic language identification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, Apr. 1980, pp. 884–887.

[19] D. Cimarusti and R. Ives, "Development of an automatic identification system of spoken languages: Phase I," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 7, May 1982, pp. 1661–1663.

[20] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, April 1975.

[21] J. Foil, "Language identification using noisy speech," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 11, Apr. 1986, pp. 861–864.

[22] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[23] F. J. Goodman, A. F. Martin, and R. E. Wohlford, "Improved automatic language identification in noisy speech," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, May 1989, pp. 528–531.

[24] Y. K. Muthusamy, "A review of research in automatic language identification," 1992.

[25] Y. K. Muthusamy, R. A. Cole, and M. Gopalakrishnan, "A segment-based approach to automatic language identification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Apr 1991, pp. 353–356.

[26] E. Barnard and R. A. Cole, "A neural-net training program based on conjugate-radient optimization," Technical Report CS/E 89-014, Oregon Graduate Institute, Beaverton, OR., Tech. Rep., 1989.

[27] M. Sugiyama, "Automatic language recognition using acoustic features," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, Apr. 1991, pp. 813–816.

[28] S. Nakagawa, Y. Ueda, and T. Seino, "Speaker-independent, text-independent language identification by HMM," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, vol. 2, Oct. 1992, pp. 1011–1014.

[29] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Trans. on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[30] L. F. Lamel and J. L. Gauvain, "Cross-lingual experiments with phone recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, Apr. 1993, pp. 507–510.

[31] L. F. Lamel, J. luc Gauvain, and M. Eskenazi, "BREF, a large vocabulary spoken corpus for French," in *Proc. European conference on speech communication and technology (Eurospeech)*, 1991, pp. 505–508.

[32] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proc. Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[33] L. F. Lamel and J. L. Gauvain, "Language identification using phone-based acoustic likelihoods," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Apr 1994, pp. 293–296.

[34] Y. K. Muthusamy, R. A. Cole, and B. T. Oshika, "The OGI multi-language telephone speech corpus," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, vol. 92, 1992, pp. 895–898.

[35] Y. K. Muthusamy, K. M. Berkling, T. Arai, R. A. Cole, and E. Barnard, "A comparison of approaches to automatic language identification using telephone speech," in *Proc. European conference on speech communication and technology (Eurospeech)*, vol. 1, 1993, pp. 1307–1310.

[36] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *The Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[37] M. Zissman, "Automatic language identification using Gaussian mixture and hidden Markov models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, 1993, pp. 399–402.

[38] K. M. Berkling, T. Arai, and E. Barnard, "Analysis of phoneme-based features for language identification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Apr 1994, pp. 289–292.

[39] R. C. Tucker, M. J. Carey, and E. S. Parris, "Automatic language identification using sub-word models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, 1994, pp. 301–304.

[40] M. A. Zissman and E. Singer, "Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Apr 1994, pp. 305–308.

[41] M. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. on Speech and Audio Processing*, vol. 4, no. 1, pp. 31–44, Jan 1996.

[42] S. Kadambe, "Spontaneous speech language identification with a knowledge of linguistics," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, 1994, pp. 1879–1882.

[43] S. Kadambe and J. L. Hieronymus, "Language identification with phonological and lexical models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, May 1995, pp. 3507–3510.

[44] T. Schultz, I. Rogina, and A. Waibel, "LVCSR-based language identification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 1996, pp. 781–784.

[45] J. Navrátil and W. Zuhlke, "Phonetic-context mapping in language identification," in *PProc. European conference on speech communication and technology (Eurospeech)*, 1997, pp. 71–74.

[46] E. Wong, J. Pelecanos, S. Myers, and S. Sridharan, "Language identification using efficient Gaussian mixture model analysis," in *Proc. International conference on Speech, Science and Technology*, 2000, pp. 78–83.

[47] J. Navratil, "Spoken language recognition-a step toward multilinguality in speech processing," *IEEE Trans. on Speech and Audio Processing*, vol. 9, no. 6, pp. 678–685, 2001.

[48] E. Singer, P. A. Torres-Carrasquillo, T. P. Gleason, W. M. Campbell, and D. A. Reynolds, "Acoustic, phonetic, and discriminative approaches to automatic language identification." in *Proc. Interspeech*, 2003.

[49] W. M. Campbell, E. Singer, P. A. Torres-Carrasquillo, and D. A. Reynolds, "Language recognition with support vector machines," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2004.

[50] W. Campbell, J. Campbell, D. Reynolds, E. Singer, and P. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech & Language*, vol. 20, pp. 210 – 229, 2006.

[51] T. Nagarajan and H. A. Murthy, "Language identification using parallel syllable-like unit recognition," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 2004, pp. 401–404.

[52] ——, "Language identification using acoustic log-likelihoods of syllable-like units," *Speech Communication*, vol. 48, no. 8, pp. 913–926, 2006.

[53] K.-Y. E. Wong, "Automatic spoken language identification utilizing acoustic and phonetic speech information," Ph.D. dissertation, Queensland University of Technology, 2004.

[54] L. Lee and R. C. Rose, "Speaker normalization using efficient frequency warping procedures," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, 1996, pp. 353–356.

[55] P. Zhan and A. Waibel, "Vocal tract length normalization for large vocabulary continuous speech recognition," Carnegie-Mellon University Pittsburgh PA School of Computer Science, Tech. Rep., 1997.

[56] J.-L. Rouas, J. Farinas, F. Pellegrino, and R. André-Obrecht, "Rhythmic unit extraction and modelling for automatic language identification," *Speech Communication*, vol. 47, no. 4, pp. 436–456, 2005.

[57] C.-Y. Lin and H.-C. Wang, "Language identification using pitch contour information," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 2005, pp. 601–604.

[58] W. Zhang, B. Li, D. Qu, and B. Wang, "Automatic language identification using support vector machines," in *8th International Conference on Signal Processing*, vol. 1, 2006, pp. 16–20.

[59] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE signal processing letters*, vol. 13, no. 5, pp. 308–311, 2006.

[60] C. H. You, H. Li, and K.-A. Lee, "A GMM-supervector approach to language recognition with adaptive relevance factor," in *Proc. European Signal Processing Conference (EUSIPCO)*, Aug. 2010, pp. 1993–1997.

[61] H. Li, B. Ma, and C.-H. Lee, "A vector space modeling approach to spoken language identification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 271–284, 2007.

[62] B. Ma, H. Li, and R. Tong, "Spoken language recognition using ensemble classifiers," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2053–2062, 2007.

[63] P. Matejka, L. Burget, O. Glembek, P. Schwarz, V. Hubeika, M. Fapso, T. Mikolov, and O. Plchot, "BUT system description for NIST LRE 2007," in *Proc. 2007 NIST Language Recognition Evaluation Workshop*, 2007, pp. 1–5.

[64] K. C. Sim and H. Li, "On acoustic diversification front-end for spoken language identification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 1029–1037, 2008.

[65] L. Mary and B. Yegnanarayana, "Extraction and representation of prosodic features for language and speaker recognition," *Speech Communication*, vol. 50, no. 10, pp. 782 – 796, 2008.

[66] F. Castaldo, S. Cumani, P. Laface, and D. Colibro, "Language recognition using language factors," in *Proc. Interspeech*, 2009.

[67] T. Mikolov, O. Plchot, O. Glembek, L. Burget, and J. Cernockỳ, "PCA-based feature extraction for phonotactic language recognition," in *Proc. Odyssey: The Speaker and Language and Language Recognition Workshop*, 2010.

[68] A. Stolcke, M. Akbacak, L. Ferrer, S. Kajarekar, C. Richey, N. Scheffer, and E. Shriberg, "Improving language recognition with multilingual phone recognition and speaker adaptation transforms," in *Proc. Odyssey: The Speaker and Language and Language Recognition Workshop*, 2010, pp. 256–262.

[69] P. A. Torres-Carrasquillo, E. Singer, W. M. Campbell, T. P. Gleason, A. McCree, D. A. Reynolds, F. Richardson, W. Shen, and D. E. Sturim, "The MITLL NIST LRE 2007 language recognition system." in *Proc. Interspeech*, 2008, pp. 719–722.

[70] P. A. Torres-Carrasquillo, E. Singer, T. Gleason, A. McCree, D. A. Reynolds, F. Richardson, and D. Sturim, "The MITLL NIST LRE 2009 language recognition system," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010, pp. 4994–4997.

[71] M. Penagarikano, A. Varona, L. J. Rodríguez, and G. Bordel, "Dimensionality reduction for using high-order n-grams in SVM-based phonotactic language recognition," in *Proc. Interspeech*, 2011, pp. 853–856.

[72] F. S. Richardson and W. M. Campbell, "NAP for high level language identification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011, pp. 4392–4395.

[73] S. Jothilakshmi, V. Ramalingam, and S. Palanivel, "A hierarchical language identification system for indian languages," *Digital Signal Processing*, vol. 22, pp. 544–553, May 2012.

[74] R. W. Ng, T. Lee, C.-C. Leung, B. Ma, and H. Li, "Spoken language recognition with prosodic features," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 21, no. 9, pp. 1841–1853, 2013.

[75] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 15, no. 4, pp. 1435–1447, May 2007.

[76] ——, "Speaker and session variability in GMM-based speaker verification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1448–1460, May 2007.

[77] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of inter-speaker variability in speaker verification," *IEEE Trans. on Audio, Speech and Lanuage Processing*, vol. 16, no. 5, pp. 980–988, July 2008.

[78] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, May 2011.

[79] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language recognition in i-vectors space," in *Proc. Interspeech*, Aug 2011, pp. 861–864.

[80] N. Dehak, P. A.Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Proc. Interspeech*, Aug. 2011, pp. 857–860.

[81] J. Yang, X. Zhang, H. Suo, L. Lu, J. Zhang, and Y. Y., "Low-dimensional representation of Gaussian mixture model supervector for language recognition," *EURASIP Journal on Advances in Signal Processing*, pp. 1–7, 2012.

[82] M. Soufifar, M. Kockmann, L. Burget, O. Plchot, O. Glembek, and T. Svendsen, "iVector approach to phonotactic language recognition," in *Proc. Interspeech*, 2011, pp. 2913–2916.

[83] O. Plchot, M. Karafiát, N. Brümmer, O. Glembek, P. Matejka, E. de Villiers, and J. Cernockỳ, "Speaker vectors from subspace Gaussian mixture model as complementary features for language identification," in *Proc. Odyssey: The Speaker and Language and Language Recognition Workshop*, 2012, pp. 330–333.

[84] D. Povey, "A tutorial-style introduction to subspace Gaussian mixture models for speech recognition," Tech. Rep., August 2009.

[85] D. Martínez, L. Burget, L. Ferrer, and N. Scheffer, "ivector-based prosodic system for language identification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012, pp. 4861–4864.

[86] D. Martinez, E. Lleida, A. Ortega, and A. Miguel, "Prosodic features and formant modeling for an ivector-based language recognition system," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 6847–6851.

[87] Q. Zhang, G. Liu, and J. H. Hansen, "Robust language recognition based on diverse features," in *Proc. Odyssey: The Speaker and Language and Language Recognition Workshop*, 2014, pp. 152–157.

[88] B. Desplanques, K. Demuynck, and J.-P. Martens, "Combining joint factor analysis and ivectors for robust language recognition," in *Proc. Odyssey: The Speaker and Language and Language Recognition Workshop*, 2014, pp. 73–80.

[89] M. Li and W. Liu, "Speaker verification and spoken language identification using a generalized i-vector framework with phonetic tokenizations and tandem features," in *Proc. Interspeech*, 2014, pp. 1120–1124.

[90] M. Diez, A. Varona, M. Penagarikano, L. J. Rodriguez-Fuentes, and G. Bordel, "On the use of phone log-likelihood ratios as features in spoken language recognition," in *Proc. Spoken Language Technology (SLT) Workshop*, 2012, pp. 274–279.

[91] ——, "New insight into the use of phone log-likelihood ratios as features for language recognition," in *Proc. Interspeech*, 2014, pp. 1841–1845.

[92] ——, "On the projection of PLLRs for unbounded feature distributions in spoken language recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1073–1077, 2014.

[93] O. Glembek, L. Burget, P. Matejka, M. Karafiát, and P. Kenny, "Simplification and optimization of i-vector extraction," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4516–4519, May 2011.

[94] S. Cumani and P. Laface, "Fast and memory effective i-vector extraction using a factorized sub-space," in *Proc. Interspeech*, 2013, pp. 1599–1603.

[95] M. Li and S. Narayanan, "Simplified supervised i-vector modeling with application to robust and efficient language identification and speaker verification," *Computer Speech and Language*, vol. 28, no. 4, pp. 940–958, 2014.

[96] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[97] I. Naseem, R. Togneri, and M. Bennamoun, "Sparse representation for speaker identification," in *Proc. International Conference on Pattern Recognition (ICPR)*, 2010, pp. 4460–4463.

[98] J. M. K. Kua, E. Ambikairajah, J. Epps, and R. Togneri, "Speaker verification using sparse representation classification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011, pp. 4548–4551.

[99] J. M. K. Kua, J. Epps, and E. Ambikairajah, "i-vector with sparse representation classification for speaker verification," *Speech Communication*, vol. 55, no. 5, pp. 707–720, 2013.

[100] B. Jiang, Y. Song, W. Guo, and L.-R. Dai, "Exemplar-based sparse representation for language recognition on i-vectors," in *Proc. Interspeech*, Sep. 2012.

[101] B. C. Haris and R. Sinha, "Speaker verification using sparse representation over KSVD learned dictionary," in *Proc. National Conference on Communications (NCC)*, 2012, pp. 1–5.

[102] B. Haris and R. Sinha, "Sparse representation over learned and discriminatively learned dictionaries for speaker verification," in *Proc. ICASSP*, March 2012, pp. 4785–4788.

[103] Y. L. Gwon, W. M. Campbell, D. E. Sturim, and H. Kung, "Language recognition via sparse coding," in *Proc. Interspeech*, 2016, pp. 2920–2924.

[104] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: Algorithms and applications," *IEEE Access*, vol. 3, pp. 490–530, 2015.

[105] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 5337–5341.

[106] Y. Song, B. Jiang, Y. Bao, S. Wei, and L.-R. Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

[107] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 1695–1699.

[108] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[109] R. Fer, P. Matejka, F. Grezl, O. Plchot, and J. Cernocky, "Multilingual bottleneck features for language recognition," *Proc. Interspeech*, pp. 389–393, 2015.

[110] S. H. Shum, D. F. Harwath, N. Dehak, and J. R. Glass, "On the use of acoustic unit discovery for language recognition," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1665–1676, Sept 2016.

[111] O. Ghahabi, A. Bonafonte, J. Hernando, and A. Moreno, "Deep neural networks for i-vector language identification of short utterances in cars," *Proc. Interspeech*, pp. 367–371, 2016.

[112] A. Sizov, K. A. Lee, and T. Kinnunen, "Discriminating languages in a probabilistic latent subspace," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2016, pp. 81–88.

[113] T. Gunawan and M. Kartiwi, "On the comparison of line spectral frequencies and mel-frequency cepstral coefficients using feedforward neural network for language identification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 10, pp. 168–175, Apr. 2018.

[114] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, 2015.

[115] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[116] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[117] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors," in *Odyssey: The Speaker and Language Recognition Workshop, Les Sables dŠOlonne*, 2018.

[118] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[119] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[120] Y. Song, B. Jiang, Y. Bao, S. Wei, and L. R. Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, November 2013.

[121] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12 – 40, 2010.

[122] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1993.

[123] S. Furui, "Comparison of speaker recognition methods using statistical features and dynamic features," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 342–350, Jun. 1981.

[124] J. Mason and X. Zhang, "Velocity and acceleration features in speaker recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1991, pp. 3673–3676.

[125] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. D. Jr., "Approaches to language identification using Gaussian mixture models and shifted delta cepstal features," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, Sep. 2002.

[126] A. E. Rosenberg, C.-H. Lee, and F. K. Soong, "Cepstral channel normalization techniques for HMM-based speaker verification," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, 1994.

[127] S. Tibrewala and H. Hermansky, "Multi-band and adaptation approaches to robust speech recognition," in *Proc. European Conference on Speech Communication and Technology*, 1997.

[128] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication*, vol. 25, no. 1, pp. 133–147, 1998.

[129] M. Alam, P. Ouellet, P. Kenny, and D. OŠShaughnessy, "Comparative evaluation of feature normalization techniques for speaker verification," *Advances in Nonlinear Speech Processing*, pp. 246–253, 2011.

[130] D. Kim, S. Umesh, M. Gales, T. Hain, and P. Woodland, "Using VTLN for broadcast news transcription," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, 2004.

[131] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Trans. on Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589, Oct 1994.

[132] P. Matejka, P. Schwarz, J. Cernockỳ, and P. Chytil, "Phonotactic language identification using high quality phoneme recognition." in *Proc. Interspeech*, 2005, pp. 2237–2240.

[133] R. Tong, B. Ma, D. Zhu, H. Li, and E. S. Chng, "Integrating acoustic, prosodic and phonotactic features for spoken language identification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 2006, pp. I–I.

[134] S. Kadambe and J. L. Hieronymus, "Language identification with phonological and lexical models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, 1995.

[135] D. Matrouf, M. Adda-Decker, L. F. Lamel, and J.-L. Gauvain, "Language identification incorporating lexical information," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, 1998.

[136] S. Konishi and G. Kitagawa, *Information Criteria and Statistical Modeling.* Springer Science & Business Media, 2008.

[137] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *IEEE Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[138] D. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[139] V. Vapnik, *The Nature of Statistical Learning Theory.* Springer Science & Business Media, 2013.

[140] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[141] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[142] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[143] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE Signal Processing Letters*, vol. 13, pp. 308–311, 2006.

[144] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Trans. on Speech and Audio Processing*, vol. 13, no. 3, pp. 345–354, 2005.

[145] A. O. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for SVM-based speaker recognition," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, 2006, pp. 1471–1474.

[146] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proc. International Conference on Machine Learning (ICML)*, 2004, p. 78.

[147] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

[148] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech*, 2011.

[149] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. Odyssey: The Speaker and Language Recognition Workshop.*

[150] N. Brummer and D. A. Van Leeuwen, "On calibration of language recognition scores," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, 2006, pp. 1–8.

[151] M. F. BenZeghiba, J.-L. Gauvain, and L. Lamel, "Language score calibration using adapted gaussian back-end," in *Proc. Interspeech*, 2009.

[152] D. A. V. Leeuwen and N. Brummer, "Channel-dependent GMM and multi-class logistic regression models for language recognition," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, June 2006, pp. 1–8.

[153] N. Brummer, "Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scores - tutorial and user manual." 2007. [Online]. Available: http://sites.google.com/site/nikobrummer/focalmulticlass.

[154] "The 2007 NIST language recognition evaluation plan." 2007. [Online]. Available: http://www.itl.nist.gov/iad/mig//tests/lre/2007/LRE07EvalPlan-v8b.pdf

[155] P. Roach, S. Arnfield, W. Barry, J. Baltova, M. Boldea, A. Fourcin, W. Gonet, R. Gubrynowicz, E. Hallum, L. Lamel, and K. Marasek, "BABEL: An Eastern European multi-language database," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, vol. 3, 1996, pp. 1892–1893.

[156] "The 2009 NIST language recognition evaluation plan." 2009. [Online]. Available: http://www.itl.nist.gov/iad/mig//tests/lre/2009/LRE09EvalPlan-v6.pdf

[157] Z. Tang, D. Wang, Y. Chen, and Q. Chen, "AP17-OLR challenge: Data, plan, and baseline," *CoRR*, vol. abs/1706.09742, 2017.

[158] D. Wang, L. Li, D. Tang, and Q. Chen, "AP16-OL7: A multilingual database for oriental languages and a language recognition baseline," *arXiv preprint arXiv:1609.08445*, 2016.

[159] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. International Conference on Machine Learning (ICML)*. ACM, 2009, pp. 689–696.

[160] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[161] S. Kong and D. Wang, "Online discriminative dictionary learning for image classification based on block-coordinate descent method," *arXiv preprint arXiv:1203.0856*, 2012.

[162] K. Shu and W. Donghui, "A brief summary of dictionary learning based approach for classification," *arXiv preprint arXiv:1205.6391*, 2012.

[163] M. Yang, L. Zhang, J. Yang, and D. Zhang, "Metaface learning for sparse representation based face recognition," in *Proc. IEEE International Conference on Image Processing*, Sept 2010, pp. 1601–1604.

[164] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 3501–3508.

[165] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2691–2698.

[166] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent K-SVD," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1697–1704.

[167] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 543–550.

[168] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conf. Signals, Systems and Computers,Pacific Grove, CA*, vol. 1, Nov 1993, pp. 40–44.

[169] B. Efron *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[170] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, April 1995.

[171] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, Jan. 2001.

[172] M. A. Saunders, B. Kim, C. Maes, S. Akle, and M. Zahr, "PDCO: Primal-dual interior method for convex objectives," *Software available at http://www. stanford. edu/group/SOL/software/pdco. html*, 2002.

[173] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[174] I. Drori and D. L. Donoho, "Solution of l1 minimization problems by LARS/homotopy," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[175] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society, Series B*, vol. 67, pp. 301–320, 2005.

[176] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[177] N. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 234–253, 2001.

[178] D. L. Candes, Emmanuel J and Donoho, "Curvelets: A surprisingly effective nonadaptive representation for objects with edges," DTIC Document, Tech. Rep., 2000.

[179] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Trans. on Image Processing*, vol. 14, no. 12, pp. 2091–2106, 2005.

[180] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, 1999, pp. 2443–2446.

[181] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 19–60, 2010.

[182] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Sept 2009, pp. 2272–2279.

[183] B. Shen, B.-D. Liu, and Q. Wang, "Elastic net regularized dictionary learning for image classification," *Multimedia Tools and Applications*, vol. 75, no. 15, pp. 8861–8874, 2016.

[184] I. Kviatkovsky, M. Gabel, E. Rivlin, and I. Shimshoni, "On the equivalence of the LC-KSVD and the D-KSVD algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 39, no. 2, pp. 411–416, 2017.

[185] X. Wang and X. Tang, "Random sampling for subspace face recognition," *International Journal of Computer Vision*, vol. 70, pp. 91–104, 2006.

[186] W. Jiang, Z. Li, and H. Meng, "An analysis framework based on random subspace sampling for speaker verification," in *Proc. Interspeech*, Aug 2011, pp. 253–256.

[187] N. Li, X. Zeng, Z. Li, W. Jiang, and Y. Qiao, "An analysis framework of two-level sampling subspace for speaker verification," in *Proc. IEEE Region 10 Conference (TENCON)*, Oct 2013, pp. 1–5.

[188] J. Zhong, W. Jiang, H. Meng, N. Li, and Z. Li, "An integration of random subspace sampling and fishervoice for speaker verification," in *Proc. Odyssey: The Speaker and Language Recognition Workshop*, June 2014, pp. 88–93.

[189] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[190] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Trans. on Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589, 1994.

[191] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[192] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, March 1989.

[193] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back propagating errors," *Nature*, vol. 323, pp. 533–536, 10 1986.

[194] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, vol. 4, 2005, pp. 2047–2052.

[195] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," in *Proc. International Conference on Artificial Neural Networks (ICANN)*, vol. 2, 1999, pp. 850–855.

[196] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, vol. 3, 2000, pp. 189–194.

[197] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Veselothers, "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[198] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *Proc. Interspeech*, 2014, pp. 2155–2159.

[199] G. Gelly, J.-L. Gauvain, V. B. Le, and A. Messaoudi, "A divide-and-conquer approach for language identification based on recurrent neural networks," in *Proc. Interspeech*, 2016, pp. 3231–3235.

[200] R. Zazo, A. Lozano-Diez, J. Gonzalez-Dominguez, D. T. Toledano, and J. Gonzalez-Rodriguez, "Language identification in short utterances using long short-term memory (LSTM) recurrent neural networks," *PloS one*, vol. 11, no. 1, p. e0146917, 2016.

[201] Z. Z. Dong Wang, Xuewei Zhang, "THCHS-30 : A free Chinese speech corpus," 2015. [Online]. Available: http://arxiv.org/abs/1512.01882

[202] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Trans. on Speech and Audio Processing*, vol. 13, no. 3, pp. 345–354, 2005.

[203] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar, "Dictionary optimization for block-sparse representations," *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2386–2395, 2012.

[204] G. Sreeram, B. Haris, and R. Sinha, "Improved speaker verification using block sparse coding over joint speaker-channel learned dictionary," in *Proc. IEEE Region 10 Conference (TENCON)*, 2015, pp. 1–5.

[205] N. Kumar and R. Sinha, "Improved structured dictionary learning via correlation and class based block formation," *IEEE Transactions on Signal Processing*, vol. 66, no. 19, pp. 5082–5095, 2018.

List of Publications

Journal Publications

1. O. P. Singh and R. Sinha, "Sparse Coding of I-vector/JFA Latent Vector over Ensemble Dictionaries for Language Identification Systems", *International Journal of Speech Technology*, Springer, 2017, pp. 1–16.

Book Chapters

1. O. P. Singh and R. Sinha, "Sparse coding of bottleneck features based i-vector representation for language recognition", *Lecture Notes in Electrical Engineering*, Springer, 2019.

Conference Publications

1. O. P. Singh and R. Sinha, " Sparse Representation Classification over Discriminatively Learned Dictionary for Language Recognition," *in Proc. IEEE Region 10 Conference (TENCON)*, Nov. 2017

2. O. P. Singh and R. Sinha, "Sparse Representation Classification based Language Recognition using Elastic Net,"*in Proc. Signal Processing and Integrated Networks (SPIN)*, Feb. 2017 .

3. O. P. Singh and R. Sinha, "Language Recognition via Sparse Coding over Learned Dictionary,"*in Proc. Signal Processing and Integrated Networks (SPIN)*, Feb. 2017.

4. O. P. Singh and R. Sinha, " Low Complexity Language Recognition Exploiting Ensemble of Random Subspace," *in Proc. Signal Processing and Communications (SPCOM)*, June 2016.

5. O. P. Singh, Haris BC and R. Sinha, " Language identification using sparse representation: A comparison between GMM supervector and i-vector based approaches,"*in Proc. Annual IEEE India Conference (INDICON)*, Dec. 2013.

Other Publication

1. O. P. Singh, Haris B C, B. Chettri, A. Pradhan, and R. Sinha, " Sparse Representation based Language Identification using Prosodic Features for Indian Languages," *in Proc. Annual IEEE India Conference (INDICON)*, Dec. 2013.