Discriminative Manifold Learning for Automatic Speech Recognition

Vikrant Singh Tomar

McGill ID: 260394445

Department of Electrical and Computer Engineering
McGill University
Montreal, Canada

December 2015

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Dedicated to my parents
Satyaveer and Premvati

Abstract

Manifold learning techniques have received a lot of attention in recent literature [1]–[4]. The underlying assumption of these techniques is that the high-dimensional data can be considered as a set of geometrically related points lying on or close to the surface of a smooth low-dimensional manifold embedded in the ambient space. These techniques have been used in a wide variety of application domains, such as face recognition [5]–[8], speaker [9] and speech recognition [10], [11]. In automatic speech recognition (ASR), previous studies on this topic have primarily focused on unsupervised manifold learning techniques for dimensionality reducing feature space transformations [12]–[14]. The goal of these techniques is to preserve the underlying manifold based geometrical relationship existing in the speech data during the transformation. However, these techniques fail to exploit the discriminative structure between the classes of speech sounds. The work in this thesis has investigated incorporating inter-class discrimination into manifold learning techniques. The contributions of this thesis work can be divided in two major categories. The first is the discriminative manifold learning (DML) techniques for dimensionality reducing feature space transformation. The second is to use the DML based constraints to regularize the training of deep neural networks (DNN).

The first contribution of this thesis is a framework for DML based feature space transformations for ASR. These techniques attempt to preserve the local manifold based nonlinear relationships between feature vectors while maximizing a criterion related to separating speech classes [15]. Two different techniques are proposed. The first is the locality preserving discriminant analysis (LPDA) [16]. In LPDA, the manifold domain relationships between feature vectors are characterized by a Euclidean distance based kernel. The second technique is the correlation preserving discriminant analysis (CPDA), which uses a cosine-correlational kernel [17]. The LPDA and CPDA techniques are compared to two well known approaches for dimensionality reducing transformations, linear discriminant analysis (LDA) and locality preserving projection (LPP), on two separate tasks involving noise corrupted utterances of both connected digits and read newspaper text. The proposed approaches are found to provide up to 30% reductions in word error rates (WER) with respect to LDA and LPP.

The manifold learning techniques, both unsupervised as well as the proposed DML techniques, suffer from two major issues. The first is the high computational complexity

associated with the computation of the manifold based neighborhood graphs [18]–[20], and the second is the performance degradation in the presence of noise [21]. In this thesis, a class of random projections based approximate near neighborhood estimation techniques known as locality sensitive hashing (LSH) is investigated for addressing the issue of computational complexity [22]–[24]. Application of LSH is shown to provide a 10 times speedup for LPDA and a 9 times speedup for CPDA with minimal impact on their ASR performance [25], [26]. To address the issue of noise sensitivity, the interaction between acoustic noise conditions and the shape and size of local neighborhoods which are used in manifold learning to define local relationships among feature vectors is studied. It is shown through experimental analysis that the performance degradation can be traced to the choice of the size of the Gaussian kernel scale factor used for defining the local neighborhood structures. Based on this analysis, a noise aware manifold learning (NaML) procedure for reducing the impact of varying acoustic conditions on manifold learning is proposed and evaluated on a speech in noise task [27]. It is shown that the NaML approach significantly reduces the speech recognition WER over LPDA, particularly at low signal-to-noise ratios.

The final contribution of this thesis is to apply the DML based constraints to optimize the training of DNNs for ASR [28], [29]. DNNs have been successfully applied to a variety of ASR tasks, both in discriminative feature extraction and hybrid acoustic modeling scenarios [30]–[33]. Despite the rapid progress in DNN research, a number of challenges remain in training DNNs. In this part of the thesis, a manifold regularized deep neural network (MRDNN) training approach is proposed that constrains the network learning to preserve the underlying manifold based relationships between speech feature vectors. This is achieved by incorporating manifold based locality preserving constraints in the objective criterion of the network. Empirical evidence is provided to demonstrate that training a network with manifold constraints strengthens the learning of manifold based neighborhood preservation and preserves structural compactness in the hidden layers of the network. The ASR WER obtained using these networks is evaluated on a connected digits speech in noise task and a read news speech in noise task. Compared to DNNs trained without manifold constraints, the MRDNNs provides 10 to 38.64% reductions in ASR WERs.

Résumé

Les techniques d'apprentissage de variétés suscitent beaucoup d'intérêt dans la littérature récente [1]–[4]. L'hypothèse sous-jacente à ces techniques est que les données de grande dimension peuvent être considérées comme un ensemble géométrique de points disposés sur (ou proches de) la surface d'une variété de dimension inférieure incluse dans l'espace ambiant. Ces techniques ont été utilisées dans de nombreux domaines d'application, comme la reconnaissance de visages [5]–[8]et de locuteurs [9] et la reconnaissance vocale [10], [11]. Pour la reconnaissance vocale automatique (RVA), les études existantes sont principalement axées sur l'apprentissage non supervisé de variétés aux fins de réduire la dimensionnalité de l'espace objet [12]–[14]. L'objectif de ces techniques est de préserver, au cours de la transformation, la relation géométrique sous-jacente à la variété existant dans les données vocales. Cependant, ces techniques n'exploitent pas les structures discriminatoires entre les différentes classes de sons dans la parole. Les travaux de cette thèse portent sur l'ajout de la discrimination interclasse dans l'apprentissage de variétés. Les contributions de cette thèse peuvent être divisées en deux principales parties. La première regroupe les méthodes d'apprentissage discriminatoire de variétés (DML) pour les transformations de l'espace objet réduisant la dimensionnalité. La seconde est l'application des contraintes de DML pour l'apprentissage des réseaux de neurones profonds (DNN).

La première contribution de cette thèse est de fournir un cadre pour l'apprentissage discriminatoire de variétés (DML) pour les transformations de l'espace objet en RVA. Ces techniques tentent de préserver les relations non linéaires basées sur une variété locale entre les vecteurs objets tout en maximisant un critère de séparation des classes de paroles [15]. Deux techniques différentes sont proposées. La première est l'analyse discriminante préservant la localité (LPDA) [16]. Avec LPDA, les relations entre les vecteurs objets propres à la variété sont caractérisées par un noyau utilisant la distance euclidienne. La seconde méthode est l'analyse discriminatoire préservant la corrélation (CPDA), qui utilise un noyau basé sur la similarité cosinus [17]. Les méthodes LPDA et CPDA sont comparées à deux approches éprouvées pour les transformations réduisant la dimensionnalité, l'analyse discriminatoire linéaire (LDA) et la projection préservant la localité (LPP), sur deux tâches distinctes avec des enregistrements bruités de chiffres connectés et de journaux lus. Les approches proposées réduisent jusqu'à 30% le taux de mots erronés (WER) par rapport à LDA et LPP.

Les méthodes d'apprentissage de variétés, tant les méthodes non supervisées que les méthodes DML, présentent deux défauts majeurs. D'une part, la complexité des calculs de graphes de voisinage des variétés est grande [18]–[20], et d'autre part, les performances se dégradent en présence de bruit [21]. Dans cette thèse, une classe de méthodes d'estimation de voisinages par projections aléatoires, appelée hachage sensible à la localité (LSH), est examinée pour résoudre le problème de complexité algorithmique [22]–[24]. L'utilisation de LSH accélère jusqu'à 10 fois LPDA et 9 fois CPDA, avec une modification minimale de leurs performances en RVA [25], [26]. Pour pallier la sensibilité au bruit, on étudie la relation entre le bruit audio et la forme et taille des voisinages locaux qui sont utilisés dans l'apprentissage de variétés pour définir les relations entre vecteurs objets. Il est montré au travers d'expériences que la dégradation des performances peut être liée au choix de la taille du facteur de normalisation du noyaux gaussien utilisé pour définir les structures des voisinages locaux. En s'appuyant sur cette analyse, une procédure d'apprentissage de variétés résistance au bruit (NAML) est proposée pour réduire l'impact des conditions acoustiques, et évaluée dans le cas d'un problème de parole avec bruit [27]. Il est montré que l'approche NAML réduit de manière significative le WER pour LPDA, particulièrement pour un rapport signal sur bruit faible.

La contribution finale de cette thèse est la mise en application des contraintes DML pour optimiser l'apprentissage de réseaux de neurones profonds (DNN) pour la RVA [28], [29]. Les DNN ont été utilisés avec succès sur un large éventail de problèmes de RVA, à la fois pour l'extraction discriminatoire de caractéristiques et des scénarios de modalisation acoustique hybrides [30]–[33]. Malgré l'avancée rapide de la recherche sur les DNN, de nombreux problèmes se posent toujours pour leur apprentissage. Dans cette partie de la thèse, une approche d'apprentissage pour un réseau de neurones profond régularisé par une variété (MRDNN) est proposée. Elle permet de contraindre l'apprentissage du réseau à préserver les relations propres à la variété sous-jacente entre les vecteurs objets, grâce à l'ajout de contraintes de préservation de la localité dans le critère objectif du réseau. Des preuves empiriques viennent étayer le fait que l'apprentissage d'un réseau avec des contraintes sur les variétés accroît l'apprentissage de la préservation de voisinage basés sur la variété et préserve la structure compacte des couches cachées du réseau. Le WER pour la RVA obtenu avec ces réseaux est évalué sur un problème de chiffres connectés avec bruit et un problème de journaux lus avec bruit. Comparés aux réseaux entraînés sans ces contraintes de variétés, les MRDNN réduisent de 10 à 38.64% le WER pour la RVA.

Contents

List of Figures

List of Tables

Repetitively used Acronyms

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CDHMM	Continuous Density Hidden Markov Model
Cos-LSH	Cosine Correlation based Locality Sensitive Hashing
CPDA	Correlation Preserving Discriminant Analysis
DML	Discriminative Manifold Learning
DNN	Deep Neural Network
E2LSH	Exact Euclidean Locality Sensitive Hashing
EBP	Error Back Propagation
EM	Expectation Maximization
GMM	Gaussian Mixture Model
HLDA	Heteroschedastic Linear Discriminant Analysis
HMM	Hidden Markov Model
LapRLS	Laplacian Regularized Least Square
LapSVM	Laplacian Support Vector Machine
LDA	Linear Discriminant Analysis
LPDA	Locality Preserving Discriminant Analysis
LPP	Locality Preserving Projection
LSH	Locality Sensitive Hashing
LT	Linear Transformation
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel Filtered Cepstrum Coefficients
ML	Manifold Learning
MLE	Maximum Likelihood Estimation
MLP	Multi Layer Perceptron
MRDNN	Manifold Regularized Deep Neural Network

MVCSR	Medium Vocabulary Continuous Speech Recognition
NaML	Noise aware Manifold Learning
N-LPDA	Noise aware Locality Preserving Discriminant Analysis
NN	Neural Network
PCA	Principal Components Analysis
ReLU	Rectified Linear Units
RLS	Regularized Least Square
SNR	Signal to Noise Ratio
STC	Semi-Tied Covariance
WER	Word Error Rate

List of Related Publications

- Vikrant Singh Tomar and Richard C. Rose, "Manifold regularized deep neural networks for automatic speech recognition", *submitted to ASRU*, 2015
- Vikrant Singh Tomar and Richard C. Rose, "Manifold regularized deep neural networks", *InterSpeech*, (Singapore), 2014
- Vikrant Singh Tomar and Richard C. Rose, "A family of discriminative manifold learning algorithms and their application to speech recognition", *IEEE/ACM transactions on Audio, Speech and Language Processing*, Jan. 2014
- Vikrant Singh Tomar and Richard C. Rose, "Locality Sensitive Hashing for Fast Computation of Correlational Manifold Learning based Feature space Transformations", *InterSpeech*, (Lyon, France), 2013
- Vikrant Singh Tomar and Richard C. Rose, "Noise aware manifold learning for robust speech recognition", *ICASSP: IEEE International Conference on Acoustics Speech and Signal Processing*, (Vancouver, BC, Canada), 2013
- Vikrant Singh Tomar and Richard C. Rose, "Efficient manifold learning for speech recognition using locality sensitive hashing", *ICASSP: IEEE International Conference on Acoustics Speech and Signal Processing*, (Vancouver, BC, Canada), 2013
- Vikrant Singh Tomar and Richard C. Rose, "A correlational discriminant approach to feature extraction for robust speech recognition", *InterSpeech*, (Portland, USA), 2012
- Vikrant Singh Tomar and Richard C. Rose, "Application of a locality preserving discriminant analysis approach to ASR", *International Conference on Information Science, Signal Processing, and their Applications (ISSPA)*, (Montreal, QC, Canada), 2012

Preface

Most of the continuous density hidden Markov models (CDHMM) systems in this thesis are trained using version 3.4.1 of hidden Markov model toolkit (HTK) [34]. HTK is developed at University of Cambridge and is available as a free C source download from: http://htk.eng.cam.ac.uk/. For some of the experiments, modifications to HTK source was done. In addition to HTK, quite a bit of code was written in MATLAB, C++ and Python as original scholarship. The work described in Chapters 3 and 5 is original scholarship. The code for exact Euclidean locality sensitive hashing (E2LSH) used in 4 was written by Alex Andoni of Massachusetts Institute of Technology, USA. Incorporation of locality sensitive hashing (LSH) in discriminative manifold learning (DML) framework is original scholarship. Some parts of the code for training the baseline deep neural networks (DNN) systems in Chapter 6 was written by George Dahl of University of Toronto. The code for the manifold regularized deep neural networks (MRDNN) described in Chapter 6 is original scholarship written in Python using libraries such as Numpy, gNumpy, and Cudamat [35], [36].

Chapter 1

Introduction

This thesis investigates the application of manifold learning based approaches for robust automatic speech recognition (ASR). Manifold learning is used to exploit the local geometric relationships for acoustic feature estimation in order to achieve better ASR accuracy and noise robustness than conventional techniques. The contributions of this thesis work can be divided in two major categories, namely the development of discriminative manifold learning based acoustic feature space dimensionality reduction techniques and manifold regularized training for deep neural networks (DNN).

This chapter provides an introduction to the main points of interest in this dissertation. Section 1.1 provides a general introduction to manifold learning, which is the primary topic of interest to this thesis. Section 1.2 introduces feature estimation and feature space transformation techniques for ASR and highlights their limitations. An introduction to manifold learning and related feature space transformations is provided in Section 1.1 followed by an overview of DNNs and their applications to ASR in Section 1.3. Section 1.4 summarizes the contributions made during this thesis work. Section 1.5 provides an outline for the remainder of the thesis.

1.1 Manifold Learning

A manifold is a locally Euclidean topological space, *i.e.*, each point of a manifold has a neighborhood that can be continuously mapped to the Euclidean space of the same dimensions and vice-versa. For example, the surface of the Earth can be locally mapped to a 2-dimensional flat Euclidean space. Thus, surface of the Earth is a 2-dimensional

manifold embedded in a 3-dimensional space. In this sense, manifold learning refers to a class of techniques that learn a low-dimensional embedding of the data points lying in a high dimensional space while preserving original characteristics of the data. These techniques are motivated by suggestions that any interesting high-dimensional data can be considered as a set of geometrically related points lying on or close to the surface of a smooth low-dimensional manifold embedded in the ambient space [1], [19], [37].

Manifold learning techniques have been primarily used for unsupervised dimensionality reducing transformations, where the local relationships between the data points in the original space are preserved during the transform. These techniques have been successfully applied to a wide range of application domains, such as image recognition [8], [20], phone and speaker recognition [13], [38]. These techniques have also been used for regularizing learning models such as regularized least squares (RLS) and support vector machines (SVM) [39].

In this thesis, manifold learning is used for discriminative dimensionality reducing transformations for robust feature estimation and for regularizing the training of deep neural networks for acoustic modeling. These methods are further discussed in the rest of the chapter.

1.2 Robust Feature Estimation in ASR

The goal of ASR is to find the most likely sequence of symbols such as words or sub-word speech units from a stream of acoustic data. In a successful human-machine communication scenario, a machine should be able to develop a functional equivalent of the speaker's intended message as effortlessly as humans can. However, this is a complex problem. Speech is a dynamic signal with a high degree of variability. Utterances can differ substantially in length, intensity and the frequency spectrum even if the same words are spoken by the same person. There are significant spectral differences between the speech of different speakers.

Combined with the aforementioned issues, the presence of acoustic noise severely affects the performance of a speech recognizer, often rendering the system unusable. Some examples of background acoustic noise are engine and wind noise in a running car, mixed noises in places like a train station or on a street. While humans are able to hold successful verbal communication in a variety of difficult acoustic environments, performance of an ASR system severally degrades in the presence of noise. In fact, a human listener's ability

to recognize speech far exceeds that of a modern speech recognition system [40]. Additional causes of acoustic distortion include filtering and other limitations of a recording device, reverberation caused by multi-path propagation in far-field recording settings and distortions caused by fading and other nonlinear effects in a varying communication channel.

Feature analysis or estimation refers to signal analysis performed in order to parameterize a given speech waveform. Another goal of feature analysis is to find a feature space or feature representation that is both good at separating different classes of speech sounds and effective at suppressing irrelevant sources of variation. A carefully engineered feature estimation process should be able to minimize the degradation caused by microphone recording, analog to digital conversion and sampling, and channel and acoustic environment. Developing a feature estimation technique that produces well-behaved features for a variety of task domains and noise conditions is a primary point of interest to this dissertation.

1.2.1 Mel Frequency Cepstrum Coefficients

One of the most widely used feature representations for ASR is the Mel-frequency cepstrum. Mel-frequency cepstrum coefficient (MFCC) features and the corresponding Mel-filterbank are motivated by the firing patterns of the hair cells situated on the cochlea in the inner ear [41], [42]. MFCCs are defined as the real cepstrum of the short time frequency spectrum of a speech signal. A block diagram of the MFCC feature extraction procedure is depicted in Fig. 1.1. A detailed description of MFCC feature estimation is provided in Section 2.2.2.

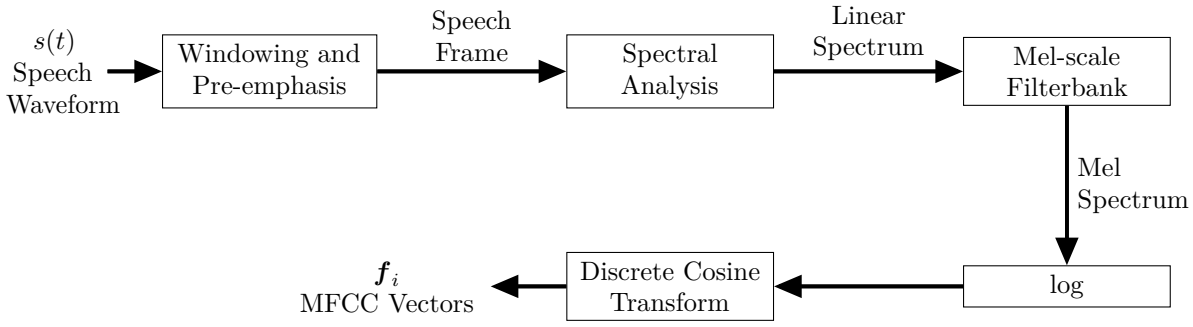

Fig. 1.1 Various building blocks of MFCC feature estimation

MFCCs give a fairly accurate measure of static features of speech; however, they fail to capture the time evolution of the speech spectrum, also referred to as the dynamics of speech.

In hidden Markov model (HMM) based speech recognition, observations or feature vectors are assumed to be conditionally independent, given the HMM state, although temporal correlation exists between frames [43], [44]. Therefore, it is desirable to capture this dynamic spectral information in speech by other means. One of the most common techniques for capturing these dynamics is to concatenate a set of static and dynamic features for each speech frame, where the static features are computed using cepstral analysis and the dynamic features are computed by the first and second order differences of the static features [45].

1.2.2 Dimensionality Reducing Feature space Transformations in ASR

While the MFCC feature vectors concatenated with the differences can capture the dynamics of speech to some degree, the resultant vectors are known to have a high degree of correlation among their components. This violates the diagonal covariance assumptions in ASR (see Section 2.3.2.2). Another way of capturing the dynamics of speech is by concatenating multiple consecutive MFCC feature vectors to form high-dimensional feature vectors that may represent on the order of 100s of milliseconds of speech. These vectors can have dimensionality as high as 1000, which may lead to significant problems when performing a pattern recognition task. This is commonly known as the *curse of dimensionality* [46]. Therefore, it is a good practice to perform some sort of dimensionality reduction before applying a particular pattern recognition algorithm to these features. An example of such a transform is shown in Fig. 1.2, where a concatenated vector $\boldsymbol{x}_i$ is transformed into a lower dimensional vector $\boldsymbol{y}_i$.

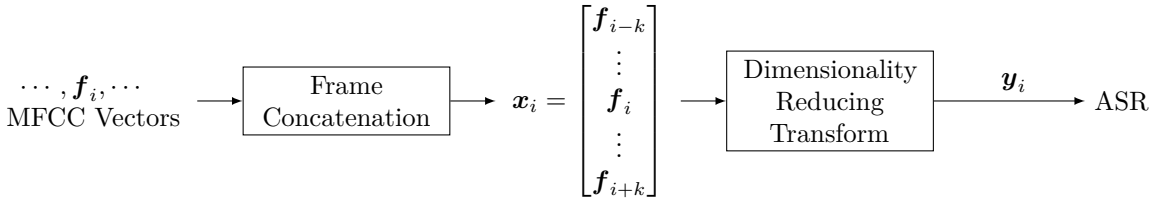

Fig. 1.2 Example of a dimensionality reducing transformations for ASR feature estimation. Feature vectors from multiple consecutive speech frames are concatenated in order to capture the time evolution of speech spectrum. This concatenated vector is then projected in to a lower dimensional space using a linear transform. The output low-dimensional features are fed onto a speech recognition system.

Intuitively, a good dimensionality reduction algorithm should also be able to preserve

important information from the original feature space in the low dimensional transformed feature vectors. Thus the dimensionality reduction problem entails finding a good feature space, where, for example, similar features are clustered together and/or features belonging to different classes are well separated. This has motivated the use of subspace learning for feature extraction and dimensionality reduction in ASR. When estimating projections from an original high-dimensional feature space to a low dimensional feature space, subspace learning establishes optimization constraints so that the desired data relations and distributions are emphasized.

A widely used family of dimensionality reducing algorithms is supervised discriminative techniques. Linear discriminant analysis (LDA) [47]–[49] and heteroscedastic linear discriminant analysis (HLDA) [50] are two examples of many such algorithms that have been widely used in ASR for reducing feature space dimensionality while maximizing a criterion related to the separability between classes of speech features. In ASR, these supervised techniques are applied to continuous density hidden Markov model (CDHMM; see Section 2.3) based acoustic models by associating feature vectors with classes corresponding to HMM states or clusters of states. One common issue with discriminative feature space transformations is their inability to capture the geometric and local distributional structure of the feature space. This has motivated the use of manifold learning based techniques for dimensionality reduction and feature space transformation in ASR. These techniques are discussed next.

1.2.3 Manifold Learning based Feature Estimation for ASR

The application of manifold learning methods to ASR is supported by the argument that speech is produced by the movements of loosely constrained articulators [13], [51]. Motivated by this, manifold learning techniques are used for dimensionality reducing feature space transformations with the goal of preserving manifold domain relationships between data vectors. One such example technique is locality preserving projections (LPP) [14], [19]. LPP aims to preserve manifold constrained relationships among data vectors during the dimensionality reducing feature space transformation so that feature vectors close to each other in the original space are also close to each other in the target space.

Manifold learning techniques are inherently unsupervised and non-discriminative. As a result, feature vectors belonging to different classes may not be optimally separated in the resultant space. Discriminating between different speech classes is a crucial aspect of speech

recognition. The better separated the speech classes are in a feature space, the easier it is to perform recognition. Another important issue with manifold learning based feature estimation techniques is that most of such approaches provide linear transformations, which may be incapable of effectively capturing the highly nonlinear nature of speech manifolds. It has been shown that exploiting these underlying nonlinear structures of speech production system can lead to significant gains in ASR performance [10], [13]. The third issue with the application of manifold learning techniques to speech processing is their very high computational complexity [19], [20]. This complexity originates from the need to calculate a pair-wise similarity measure between feature vectors to construct nearest neighborhood graphs, which are essential to all manifold learning techniques.

In conclusion, the state of the art feature estimation and transformation techniques suffer from a number of shortcomings. While the MFCC based simple techniques fail to capture the dynamics of speech, the discriminative feature space transformation techniques, such as LDA, ignore the underlying manifold based relationships between speech feature vectors. The manifold based techniques, such as LPP, do not discriminate between various classes of feature vectors and thus may not be able to optimally separate the speech features. The ASR performance of all of these techniques further degrades in the presence of noise. Motivated by these issues, the first part of this thesis research proposes discriminative manifold learning (DML) techniques that investigate the potential merit to integrating discriminative learning into manifold preservation algorithms in ASR. The DML techniques are discussed in Chapter 3 through 5.

1.3 Deep Neural Networks for ASR

Artificial neural networks, also referred to simply as neural networks (NNs), have been widely applied to acoustic modeling in ASR [52], [53] (also see Section 2.7). However, it was not until recently that researchers observed worthwhile gains in ASR performance by using these models. The quintessential step in this direction has been the successful application of deep neural networks (DNNs) to ASR tasks [54]–[56]. DNNs, as the name suggests, are feed-forward neural networks with multiple hidden layers. The high modeling capacity of DNNs provides an effective way to deal with the large variety of speech, speaker, channel, and environmental conditions typically encountered in an ASR task [57]. As DNNs are essentially multi-layer perceptrons (MLPs), they can be trained with the well-known

error back propagation (EBP) procedure. The higher number of hidden layers give DNNs incredible modeling capacity; however, it also means very large number of parameters and high computational complexity. Therefore, deep networks are prone to over-fitting and getting trapped in poor local optima [58], [59].

DNN training algorithms have received considerable attention in the recent literature [58], [59]; this includes the approaches for optimizing DNN training. Hinton et. al [60] proposed deep belief networks (DBNs) as an effective way of initializing DNN training. The DBN pre-training procedure treats each consecutive pair of layers as a restricted Boltzmann machine (RBM). The RBM parameters are trained in an unsupervised fashion by maximizing the likelihood of input and outputs using a contrasive divergence algorithm [61], [62]. Bengio et. al. [63] proposed another layer-wise training mechanism. In this approach, the DNN is grown layer-by-layer by performing discriminative EBP training only on the last two layers at a time. In a similar work, Seide et. al. [64] reported better ASR performance than DBN pre-training by using a layer-by-layer discriminative training; however, contrary to Bengio et. al., they updated the weights of the entire network after adding a new layer. Another pre-training method is the stacked auto-encoder based pre-training [65], [66], and yet another is presented in [67]. Many recent studies have shown that the network pre-training has little impact on ASR performance if enough observation vectors, typically on the order of hundreds of millions, are available for training [68], [69]. Therefore, there is no general agreement on the importance of pre-training for DNNs, and an effective training of DNNs for ASR remains an important open problem.

Another important issue with DNN training is the impact of the local structure of the feature space on EBP optimization, including the features provided to the input of the DNN as well as features produced at the output of hidden layers of the DNN. It has been suggested that features' propagation to higher layers of the DNN is well behaved if the input feature space has a strong local structure [57]. Furthermore, de-noising auto-encoder has been described as a mechanism for learning a low-dimensional manifold based representation of the training data, albeit without explicitly imposing any such constraints [58], [65], [70], [71].

Considering that manifold based constraints emphasize the underlying local structure of speech features, manifold learning in DNN framework has the potential to address several important learning issues. The research conducted in the second part of this thesis attempts to address the issues related to DNN training by incorporating the local manifold based

constraints into DNN training. Because of their distributed and nonlinear architecture, neural networks are capable of learning highly nonlinear structures with large parameter spaces and are flexible enough to combine diverse features [52]. Manifold regularized deep neural networks are discussed in the Chapter 6.

1.4 Contributions

The first contribution of this thesis is a discriminative manifold learning framework for feature space transformations in ASR. This is presented in Chapter 3. The second contribution, as described in Chapter 4, is the application of locality sensitive hashing (LSH) based fast neighborhood search algorithms to manifold learning techniques for reducing the computational complexity of these methods. The third contribution, as given in Chapter 5, is the analysis of the impact of noise on manifold learning and discriminative manifold learning methods and the development of a noise aware manifold learning (NaML) scheme. The fourth contribution of this thesis work is the development of efficient training algorithms of DNNs using manifold regularization. This work is presented in Chapter 6. The rest of this section provides a summary of these contributions.

1.4.1 Discriminative Manfiold Learning based Feature Space Transformations

The first contribution of this thesis is presented as a framework that incorporates a discriminative component into manifold learning techniques in order to maximize the separability between different classes while preserving the within-class local manifold constrained relationships of the feature vectors. This results in a feature space where similar feature vectors are close together and feature vectors belonging to different classes are far apart. This is discussed in Chapter 3. Discriminative manifold learning framework acts by embedding feature vectors into two separate high-dimensional graphs and then optimizing the structure of these graphs under a set of constraints. In the graphs, the feature vectors are represented by the nodes and closeness or affinity between two feature vectors is represented by the weight connecting the two nodes [16], [18]. In this work, two different metrics have been used to define the affinity weights leading to two different approaches. The first, locality preserving discriminant analysis (LPDA), defines the affinity between nodes as a Euclidean distance metric [15], [16]. The second, correlation preserving discriminant analysis (CPDA), uses a cosine-correlation distance metric to define the manifold domain affinity between nodes

[15], [17]. The use of the cosine-correlation based distance metric is motivated by studies where cosine distance based metrics have been found to be more robust to noise corruption than Euclidean distances [9], [20], [72]. Therefore, CPDA is expected to demonstrate a performance advantage over LPDA in high noise scenarios.

Although this is the first study of applying discriminative manifold learning techniques to ASR, there have been some work on extending manifold based algorithms with some notion of discriminative power in other application domains [8], [18], [20].

1.4.2 Locality Sensitive Hasing for Manifold Learning

Chapter 4 of this thesis discusses the use of locality sensitive hashing (LSH) based methods for fast construction of the neighborhood graphs in order to address the issue of high computational complexity in manifold learning techniques. LSH creates hashed signatures of vectors in order to distribute them into a number of discrete buckets such that vectors close to each other are more likely to fall into the same bucket [22]–[24]. In this manner, one can efficiently perform similarity searches by exploring only the data-points falling into the the same or close-by buckets.

1.4.3 Noise aware Manifold Learning

Chapter 5 of this thesis provides an analysis of the effect of noise on various feature estimation techniques for ASR, namely MFCC, LDA, LPP and LPDA. It is shown that even though the performance of all linear feature space transformation approaches degrades when applied to noise corrupted speech, the discriminative manifold learning techniques are the least affected [15]–[17], [27]. In addition, a NaML approach is presented that addresses the interaction between acoustic noise conditions and structure of the local neighborhoods used in manifold learning. It is shown that NaML is an effective way to apply manifold learning techniques to varying acoustic environments in ASR [27].

1.4.4 Manifold Regularized Deep Neural Networks (MRDNN)

In Chapter 6 of this thesis, a manifold learning approach to regularize DNN back-propagation training is proposed. This training procedure emphasizes local relationships among speech feature vectors along a low dimensional manifold while optimizing network parameters. This is achieved by imposing manifold based locality preserving constraints on the network

outputs. The study is conducted in the context of DNN based feature estimation in a tandem ASR configuration. Results are presented demonstrating the impact of MRDNN training on both the ASR word error rates (WER) obtained from MRDNN trained models and on the behavior of the associated nonlinear feature space mapping.

1.5 Organization of the Thesis

This chapter has presented a brief introduction to this thesis work. The remainder of this thesis is structured as follows. Chapter 2 provides the necessary background for the rest of the thesis. Chapter 3 presents the DML framework and the LPDA and CPDA feature space transformation techniques. Chapter 4 presents the work done in investigating LSH for addressing the issue of high computational complexity of manifold learning methods. Chapter 5 presents NaML as an approach to address the issue of noise sensitivity in manifold learning techniques. The application of manifold learning to regularize deep neural training is presented in Chapter 6. Finally, Chapter 7 concludes the thesis and presents possible topics of future work.

Chapter 2

Background

This chapter reviews the concepts and theory for automatic speech recognition systems, manifold learning and it's application in ASR, deep neural networks, and other required background for the work presented in this thesis. An overview of automatic speech recognition as a statistical pattern recognition task is given in Section 2.1, followed by a discussion of the main elements of a speech recognition system including front-end processing in Section 2.2, acoustic modeling in Section 2.3, language modeling in Section 2.4, and decoding in Section 2.5. As a background of the discriminative manifold learning basedtechniques proposed in this thesis, feature space transformation techniques are discussed in Section 2.6. The second set of approaches investigated in this thesis involves the development of a new class of algorithms for training deep neural networks. The discussion of these techniques requires a background in neural networks and their application to ASR. This is discussed in Section 2.7.

2.1 Statistical Framework for ASR

Modern ASR systems are based on statistical modeling approaches, where the ASR problem translates to finding the most likely sequence of symbols, such as word string, $\hat{\mathcal{W}} = w_1, w_2, \ldots, w_N$, or other sub-word unit string, $\hat{\mathcal{U}} = u_1, u_2, \ldots, u_{N'}$, from all possible such sequences given an acoustic observation sequence, $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T]^\top$, where each $\boldsymbol{x}_t$ is a row vector representing the feature vector extracted from an observation window beginning at time t, and $\top$ signifies the matrix transpose operation; see Section 2.2 for a detailed discussion on feature extraction. There are a large number of word or phoneme sequences

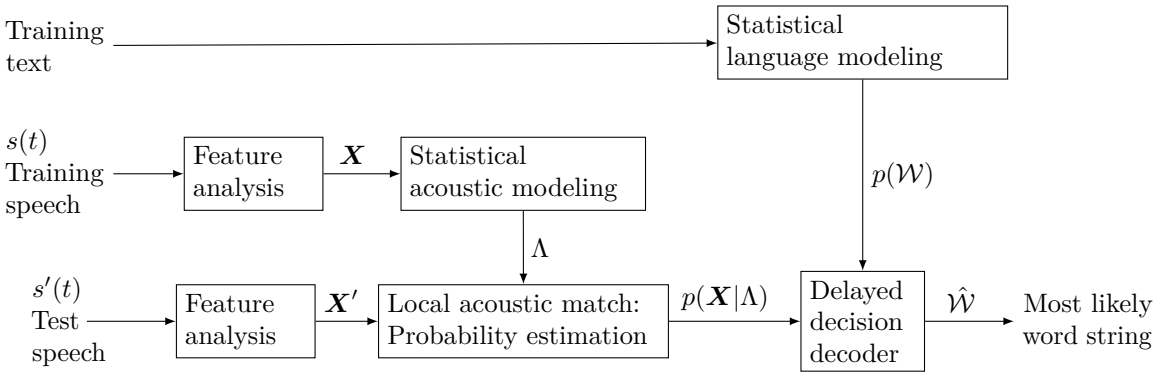

Fig. 2.1 An overview of automatic speech recognition.

which an ASR system need to search to find the most likely sequence, therefore the process consists of a number of modules that are shown in Figure 2.1 and described throughout this chapter.

Formally, an ASR system maps the observation vectors, $\boldsymbol{X}$, to the optimum sequence of words, $\hat{\mathcal{W}}$, which satisfies the *maximum a posteriori* (MAP) decision rule given as [73], [74],

$$\hat{\mathcal{W}} = \arg\max_{\mathcal{W}} p(\mathcal{W}|\boldsymbol{X}), \tag{2.1}$$

where $\mathcal{W}$ represents a potential speech sequence, $p(\mathcal{W}|\boldsymbol{X})$ is the posterior probability of the word sequence given the acoustic input. The conditional probability is evaluated over all possible word sequences from the lexicon. The posterior probability can be expressed using Bayes rule as,

$$p(\mathcal{W}|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\mathcal{W})p(\mathcal{W})}{p(\boldsymbol{X})}, \tag{2.2}$$

where

- $p(\boldsymbol{X}|\mathcal{W})$: probability of observing $\boldsymbol{X}$ under the assumption that $\mathcal{W}$ is the true utterance,
- $p(\mathcal{W})$: prior probability that sequence $\mathcal{W}$ is uttered, and
- $p(\boldsymbol{X})$: average probability that $\boldsymbol{X}$ will be observed.

Note that the denominator, $p(\boldsymbol{X})$, does not depend on the utterance $\mathcal{W}$ and is invariant for all candidate word sequences. Therefore, it does not affect the outcome of finding the

optimal sequence, and can be safely ignored. Thus, the final decision rule is,

$$\hat{\mathcal{W}} = \arg\max_{\mathcal{W}} \left\{ \underbrace{p(\boldsymbol{X}|\mathcal{W})}_{acoustic\ \ model\ \ probability} \times \underbrace{p(\mathcal{W})}_{language\ \ model\ \ probability} \right\}. \tag{2.3}$$

Eq. (2.3) shows that the entire decision model can be decomposed into acoustic model and language model probabilities. If phone or other sub-word level acoustic models are used, Eq. (2.3) can be written as,

$$\hat{\mathcal{W}} = \arg\max_{\mathcal{W}} \left\{ \underbrace{p(\boldsymbol{X}|\mathcal{U})}_{acoustic\ \ model} \times \underbrace{p(\mathcal{U}|\mathcal{W})}_{pronunciation\ \ model} \times \underbrace{p(\mathcal{W})}_{language\ \ model} \right\}, \tag{2.4}$$

where

- $p(\boldsymbol{X}|\mathcal{U})$: acoustic probability of the sub-word acoustic model, and
- $p(\mathcal{U}|\mathcal{W})$: pronunciation model probability that a word string $\mathcal{W}$ is expanded into a sub-word units string $\mathcal{U}$.

For simplicity and tractability purposes, ASR systems assume independence between the parameters of the acoustic model and language model. Both the acoustic and language models are trained on labeled training sets. As with other machine learning systems, an ideal ASR system should generalize well to unseen test data. During the test phase, a delayed decision decoder uses the acoustic and language models to search for the most likely word string for a given utterance. Typically words are expanded into phoneme context dependent acoustic models to characterize co-articulation phenomena in continuous speech [42], [73].

2.2 Front-end Processing

The first stage for an ASR system is to capture the continuous speech signal input and parameterize it to an appropriate form for signal analysis and recognition. Typically the speech signal is sampled by hardware devices such as computer sound cards into digital signals. This involves both time discretization (sampling) and value quantization. This digitized signal is analyzed for feature extraction to obtain a compact representation of speech.

The goal of this process is to capture all relevant information necessary for distinguishing speech units while discarding any irrelevant information. The front-end processing consists of a number of steps as illustrated in Figure 2.2. The rest of this section describes these steps.

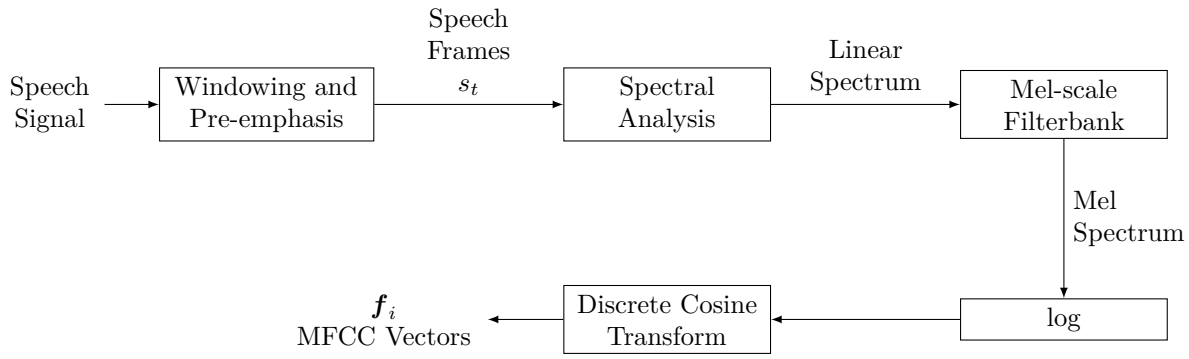

Fig. 2.2 Schematics of the MFCC feature extraction process.

2.2.1 The Spectrogram

The signal analysis of the present day ASR systems is based on short term spectral analysis [41], [75]. The first step of the signal analysis is to partition the speech signal into a series of short frames. The speech waveform, though highly non-stationary, is assumed to be piecewise stationary. A common practice is to use a 25 ms wide Hamming window with 15 ms overlap between frames. Hamming window is used as a tapering function to reduce the discontinuities and spectral leakage at the frame edges. A pre-emphasis filter is also used to boost the energy at higher frequencies. Generally, a first order moving average high pass filter is used. In simple form it can be implemented as $out[t] = in[t] - (1-\alpha)in[t-1]$, where $\alpha \in [0.9, 1]$. Sliding window discrete Fourier transform (DFT) is then applied to each speech frame to obtain the discrete *short-time Fourier transform (STFT)* or *linear spectrum* as,

$$\begin{aligned} S_t[k] &= S_t(\omega)|_{\omega=\frac{2\pi k}{T}} \\ &= \sum_{m=-\infty}^{+\infty} s[m]w[t-m]e^{-j\omega t}|_{\omega=\frac{2\pi k}{T}}, \end{aligned} \tag{2.5}$$

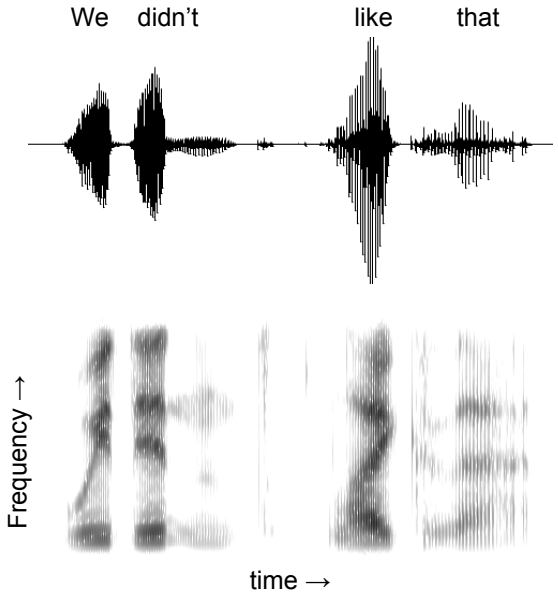

Fig. 2.3 Waveform and corresponding spectrogram for an utterance 'we didn't like that'.

where $s[m]w[t-m]$ is a short-time section of the speech signal $s[m]$ at time t. Eq. (2.5) should be interpreted as sampling the discrete-time STFT at T discrete frequencies, $\omega_k = 2\pi k/T$, in a manner similar to how DFT is sampling the discrete-time Fourier transform at T discrete frequencies.

The *spectrogram* of the speech signal is then obtained by keeping the log magnitude of the discrete STFT,

$$\mathcal{S}_t[k] = \log |S_t[k]|^2. \tag{2.6}$$

An example of spectrogram is given in Figure 2.3. Spectrogram represents the relative energy content in the different frequency bins at different time instants. The next step in the front-end processing of a speech signal is the Mel filter bank [76], which converts the linear discrete STFT spectrum to Mel frequency spectrum. This is described in the next section.

2.2.2 Mel-Frequency Cepstrum Coefficients (MFCC)

As discussed in Section 1.2.1, Mel-frequency cepstrum coefficients (MFCC) is one of the most common feature representation for speech. A block diagram of the MFCC feature

extraction process is shown in Figure 2.2. The discrete STFT of each speech frame is processed through a Mel-scaled filter bank consisting of a set of overlapping triangular masks to form a vector of output coefficients. Note that the Mel-filter bank is not a filter in true sense and is only applied to the magnitude of the spectra. Phase information is lost at this stage. The relation between linear frequency (Hz), and Mel frequency is given by,

$$\begin{aligned} f_{Mel} &= 1127 \log\left(1 + \frac{f_{Hz}}{700}\right) \\ \text{or,} \quad f_{Mel} &= 2595 \log_{10}\left(1 + \frac{f_{Hz}}{700}\right), \end{aligned} \tag{2.7}$$

where f_{Mel} and f_{Hz} correspond to the frequencies in Mel and linear domains respectively.

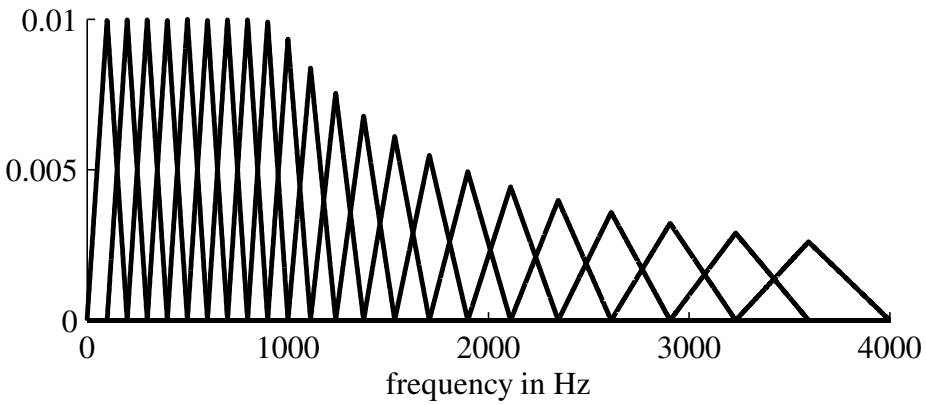

Fig. 2.4 A Typical Mel filter bank

The structure of a typical Mel filter-bank is shown in Fig. 2.4. The filter-bank emphasizes the lower frequencies and has higher resolution at lower frequencies and lower resolution at higher frequencies. This is motivated from the cochlear processing in human ear [77], [78]. The logarithm of the amplitudes of the filter-bank outputs gives a vector of filter-bank coefficients. If there are a total of $\mathfrak{M}$ filters in the filter-bank, the log energy of the $\mathfrak{m}^{\text{th}}$ filter's output is given as,

$$S_t[\mathfrak{m}] = \log\left(\sum_{k=0}^{T-1} |S_t[k]|^2 H_{\mathfrak{m}}[k]\right) \quad 0 \leq \mathfrak{m} < \mathfrak{M}, \tag{2.8}$$

where $H_{\mathfrak{m}}[k]$ is the transfer function of $\mathfrak{m}^{\text{th}}$ filter.

The log compressed output of the Mel filter bank is then converted into a vector of cepstral coefficients by applying discrete cosine transform (DCT) to the $\mathfrak{M}$ filter outputs,

$$c_{t,d} = \sum_{\mathfrak{m}=0}^{\mathfrak{M}-1} S_t[\mathfrak{m}] cos\left(\frac{\pi d(\mathfrak{m}-0.5)}{\mathfrak{M}}\right) \quad \text{for } d = 1, \ldots, \mathfrak{M}, \tag{2.9}$$

where $c_{t,d}$ is the d^{th} coefficient of the resultant feature vector obtained from speech frame at time t. DCT also helps de-correlating the features within a frame. The output vectors of this process are known as MFCC features of the speech signal and constitutes the basis of most ASR systems. In speech recognition, usually only 12 cepstral coefficients are used along with the zeroth cepstral coefficient, or the log energy, to give a 13 dimensional MFCC feature vector.

2.2.3 Post Processing

MFCC features model the smoothed spectral envelope over the short-time stationary frames of speech. MFCC is one of the most widely used feature representation in speech. However, there are a number of post-processing steps that are crucial for high performing ASR systems. This is particularly true for CDHMM based acoustic models. Two of the most common post-processing procedures are discussed here.

2.2.3.1 Capturing the Dynamics of Speech Spectrum

In CDHMM based speech recognition, observations or feature vectors are assumed to be conditionally independent, given the CDHMM state, although temporal correlation exists between frames. This temporal evolution of speech spectrum is also found to be crucial in the human auditory system [43], [44]. Therefore, it is desirable to capture this dynamic spectral information in speech. This is usually done by augmenting the static MFCC feature vectors either by the first and higher order differences or by concatenating multiple consecutive static feature vectors to form high dimensional super-vectors.

The first order dynamic, also known as difference or delta, coefficients can be calculated as follows,

$$\Delta \boldsymbol{c}_t = \frac{\sum_{\delta=1}^{\Delta} \delta(\boldsymbol{c}_{t+\delta} - \boldsymbol{c}_{t-\delta})}{2\sum_{\delta=1}^{\Delta} \delta^2} \tag{2.10}$$

The same equation can also be applied to the delta coefficients to obtain the second order

difference or delta-delta coefficients, and in a similar manner to obtain other higher order coefficients. The final vector is then written as follows,

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{c}_t \\ \vdots \\ \Delta \boldsymbol{c}_t \\ \vdots \\ \Delta\Delta \boldsymbol{c}_t \end{bmatrix}. \tag{2.11}$$

2.2.3.2 Cepstral Mean and Variance Normalization

Another important post-processing step is to normalize the mean and variance of the cepstral features making them zero mean and unit variance. For cepstral mean normalization (CMN), long term mean, typically that of a sentence, is subtracted from the coefficients in order to remove the effects of time-invariant convolutional distortions, such as those introduced by the transmission channel, the recording device and lip-radiation. Similarly, cepstral variance normalization (CVN) is used to remove the effect of noise to some degree. CMN, CVN, or jointly cepstral mean and variance normalization (CMVN) are computed as,

$$\text{CMN}: \quad \boldsymbol{x}_t \leftarrow \boldsymbol{x}_t - \boldsymbol{\mu}, \tag{2.12}$$

$$\text{CVN}: \quad \boldsymbol{x}_t \leftarrow \boldsymbol{x}_t ./ \boldsymbol{\sigma}, \tag{2.13}$$

$$\text{CMVN}: \quad \boldsymbol{x}_t \leftarrow (\boldsymbol{x}_t - \boldsymbol{\mu}) ./ \boldsymbol{\sigma}, \tag{2.14}$$

where $\boldsymbol{\mu}$ is the maximum likelihood sample mean estimator computed as $\boldsymbol{\mu} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t$, $\boldsymbol{\sigma} = \frac{1}{T-1}\sum_{t=1}^{T} ||\boldsymbol{x}_t - \boldsymbol{\mu}||_2^2$ is the maximum likelihood sample variance estimator, and $./$ signifies element wise division. Eq. (2.12) - (2.14) can be trivially used for off-line use. For on-line use, the moments are approximated by a running estimate from the past frames of the current utterance. Both CMN and CVN are commonly used in ASR.

2.3 HMM based Acoustic Modeling of Speech

An acoustic model is a statistical representation of speech sounds that is capable of learning from and representing the information present in the feature vectors. This information leads to estimation of probability of observing an acoustic feature vector given that a particular

utterance is spoken. A good model should also take into account the diverse speaking styles and background conditions, while at the same time be robust against acoustic variability. Researchers have explored a number of approaches for acoustic modeling speech, such as dynamic time warping (DTW) [79], [80], neural networks (NNs) [52], [53] and hidden Markov models [74], [81], [82]. Typically, HMMs are used to model the sequential properties of speech, complemented by a second stochastic process, such as Gaussian mixture models (GMMs) or NNs, for modeling the local properties of speech [33], [83], [84]. For the scope of this thesis, GMMs are used for modeling the local properties of speech.

HMM based systems provide a rich and flexible mathematical framework for building speech recognition systems and easily accommodate different levels of phonological and syntactical constraints. These statistical machine learning models are usually based on speech units such as words or sub-word units and are trained on a large amount of speech data containing many occurrences of the speech units in a variety of contexts. This deals with the temporal and spectral variations as well as the coarticulation in speech [74]. Incorporating data from a large number of speakers in model training results in a speaker independent model that should be able to take the inter-speaker variabilities into account.

An HMM is a finite state machine in which each state j corresponds to a unit of speech. HMM is a generative model of speech; each state has an associated output distribution that allows the acoustic data, $\boldsymbol{x}$, to be *probabilistic outputs of a (hidden) Markov chain* with an observation probability $p(\boldsymbol{x}|state = j)$. The models transitions to the next state with a state-transition probability, a. The underlying state sequence is hidden and only the emissions or outputs are observed. The models determines the *unknown* phone string from the *known* outputs of the process, *i.e.*, speech feature vectors. Two key assumptions are made in the application of HMMs to ASR:

- Conditional independence: The probability of observing $\boldsymbol{x}_t$ is conditionally independent of all past observations and states given the current state, q_t.
- First order Markov: The probability of transitioning to the next state, q_{t+1}, is independent of the past states, $\mathcal{Q}_1^{t-1} = q_1, q_2, \cdots, q_{t-1}$, given the current state, q_t:

$$p(q_t|q_{t-1}, \cdots, q_1) = p(q_t|q_{t-1}).$$

The above two assumptions are not valid for speech; however, they are important for

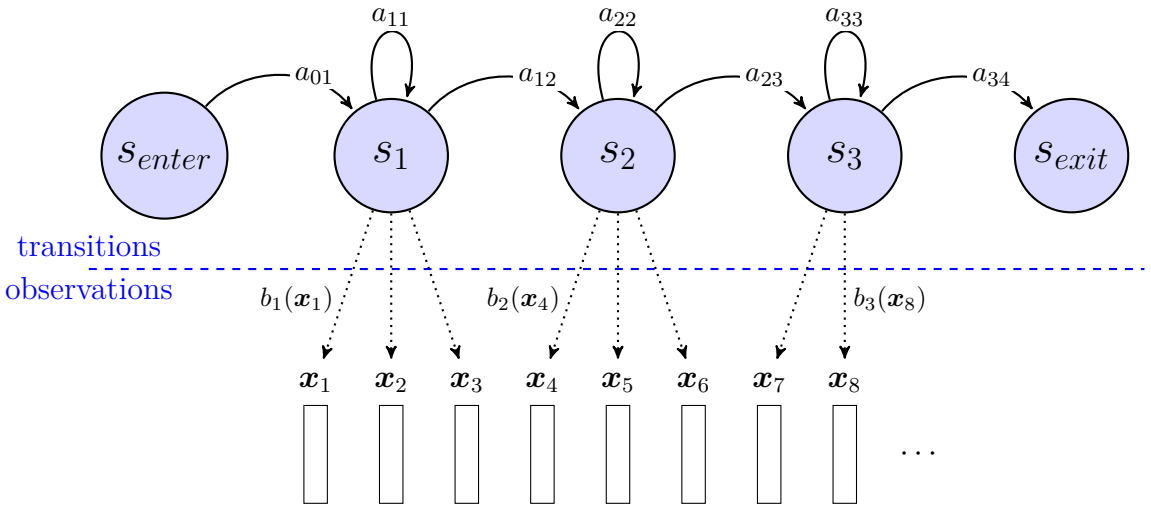

Fig. 2.5 An example of the left-to-right topology of HMM for ASR along with state-transition and observation probabilities.

mathematical convenience. Despite these disadvantages, HMMs have proven to be the most successful statistical models for speech.

The objective of HMM based acoustic modeling is to accurately estimate the parameters of the HMMs from a training dataset and efficiently compute the output probability $p(\boldsymbol{X}|\mathcal{W})$. Typically, each phone or other sub-word unit of speech is modeled by a multi-state left-to-right HMM, in which the only transitions allowed are the transition to the same state or to the next right state [74]. An example of this topology is shown in Figure 2.5. The first and last states are non-emitting enter and exit states, respectively. Speech signal is assumed to be piece-wise stationary and producing independent and identically distributed (iid) feature vectors within each HMM state. For example, when phone level models are generated, each phone is represented by an HMM that consists of states corresponding to segments of the phone such as onset, steady and release portion of vowels. An HMM, Λ, can be characterized by following variables:

1. State of the chain at time t, q_t, and total number of states, J.

2. The 1-step state-transition probability matrix, $\boldsymbol{A} = [a_{jk}]_{J\times J}$, where,

$$a_{jk} = p(q_t = k|q_{t-1} = j, \Lambda).$$

3. Observation probability for an arbitrary feature vector $\boldsymbol{x}$ in each state, $\boldsymbol{B} = \{b_j(\boldsymbol{x})\}$,

where $b_j(\boldsymbol{x}) = p(\boldsymbol{x}|q_t = j, \Lambda)$ indicates the probability of observing $\boldsymbol{x}$ in state j at time t. This is also known as the state emission probability. These probabilities represent the local properties of speech spectrum. This probability estimation is performed assuming some parametric form of the HMM states such as GMM or NN. For instance, for a GMM model, the probability is represented by the similarity between the given vector and the mean vectors of the Gaussians modeling the HMM state and is measured as the negative exponent of Mahalanobis distance[1]. This is further discussed in Section 2.3.2.2.

4. Initial state distribution, $\boldsymbol{\Pi} = \{\pi_j\}, \quad \pi_j = p(q_1 = j, \Lambda)$.

Thus, an HMM ASR model is denoted as:

$$\Lambda = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\Pi}\}. \tag{2.15}$$

Application of HMMs to speech can be understood as a 3-parts problem:

- *evaluate* the acoustic likelihood of the model, Λ, given an observation sequence, $\boldsymbol{X}$[2]
- *estimate* the optimal set of parameters in order to maximize the acoustic likelihood, *i.e., train* the model given an observation sequence, $\boldsymbol{X}$, and initial parameters of the HMM model, Λ.
- *decode* the most likely word string given a model, Λ, and an observation sequence, $\boldsymbol{X}$

These issues are discussed below.

[1] For two random vectors $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ belonging to the same distribution with covariance matrix $\boldsymbol{\Sigma}$, Mahalanobis distance is given by, $d(\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}) = \sqrt{(\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}})^T \boldsymbol{\Sigma}^{-1} (\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}})}$

[2] Throughout this thesis, $\boldsymbol{X}$, is used to denote all the feature vectors in a given set, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T]^\top$. Any sub- or super-script is avoided for simplicity. Where necessary, $\boldsymbol{X}_{from}^{to}$ is used to denote a range of feature vectors. The same is true for other sequences such as $\mathcal{W}$, $\mathcal{Q}$, etc.

2.3.1 Evaluating an HMM – Forward Algorithm

Evaluating an HMM refers to computing the score or likelihood $p(\boldsymbol{X}_1^T|\Lambda)$ given the observation sequence, $\boldsymbol{X}_1^T = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T]^\top$,

$$\begin{aligned} p(\boldsymbol{X}_1^T|\Lambda) &= \sum_{\mathcal{Q}_1^T} p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\Lambda) \\ &= \sum_{\mathcal{Q}_1^T} \prod_{t=1}^{T} p(q_t|q_{t-1}, \Lambda) p(\boldsymbol{x}_t|q_t, \Lambda). \end{aligned} \tag{2.16}$$

Calculating the likelihood as per Eq. (2.16) requires summation over all $O(J^T)$ possible state sequences. This can be computed efficiently, $O(J^2T)$, by using forward recursion:

$$\begin{aligned} \alpha_j(t) &= p(\boldsymbol{X}_1^t, q_t = j|\Lambda) \\ &= p(\boldsymbol{x}_t|q_t = j, \Lambda) \sum_{i=1}^{J} p(q_t = j|q_{t-1=i}, \Lambda) p(\boldsymbol{X}_1^{t-1}, q_{t-1} = i|\Lambda) \\ &= b_j(\boldsymbol{x}_t) \sum_{i=1}^{J} a_{ij} \alpha_i(t-1), \end{aligned} \tag{2.17}$$

where $\alpha_j(t)$ is the *forward probability* of being in state j at time t after observing the sequence $\boldsymbol{X}_1^{t-1}$. Therefore, the final probability is given by summing over the probability of observing all possible state sequences at time $t = T$,

$$p(\boldsymbol{X}_1^T|\Lambda) = \sum_{j=1}^{J} \alpha_j(T) \tag{2.18}$$

2.3.2 Training an HMM – Expectation-Maximization Algorithm

Training an HMM refers to learn the parameters of model, $\Lambda = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\Pi}\}$, to best describe the observed data, $\boldsymbol{X}$, or maximize the log likelihood,

$$\mathcal{L}(\Lambda) = \log p(\boldsymbol{X}|\Lambda). \tag{2.19}$$

Unfortunately, there is no closed form analytical solution for Eq. (2.19). This is because the data is incomplete as the states are unknown (hidden). This problem is solved by iterative Baum-Welch or forward-backward algorithm [81], [85], [86]. The Baum-Welch algorithm is

a special case of expectation-maximization (EM) algorithm [87], [88].

The EM algorithm can be seen as a generalization of the maximum likelihood estimation (MLE) method when the observable data is incomplete [41]. The observable data and the unobservable or hidden data, $\mathcal{Q}$, is jointly referred to as complete data. The observed data, $\boldsymbol{X}$, itself is referred to as incomplete data. The EM algorithm iteratively works in two steps to maximize the likelihood of the complete data. In the E (expectation) step, posterior distribution of the hidden data is estimated by using the current model parameters. In the M (maximization) step, updates to the model parameters are determined such that the complete data likelihood given the current estimate of the hidden data is maximized. The updated model $\hat{\Lambda}$ guarantees that $\mathcal{L}(\hat{\Lambda}) \geq \mathcal{L}(\Lambda)$.

M step:

Maximizing the complete data likelihood is achieved by maximizing the following auxiliary or Q-function[3],

$$\begin{aligned} Q(\Lambda, \hat{\Lambda}) &= \mathbb{E}_{\mathcal{Q}_1^T|\boldsymbol{X}_1^T,\Lambda}[\log p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\hat{\Lambda})] \\ &= \sum_{\mathcal{Q}_1^T} p(\mathcal{Q}_1^T|\boldsymbol{X}_1^T, \Lambda) \log p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\hat{\Lambda}) \\ &= \sum_{\mathcal{Q}_1^T} \frac{p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\Lambda)}{p(\boldsymbol{X}_1^T|\Lambda)} \log p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\hat{\Lambda}), \end{aligned} \tag{2.20}$$

where $\mathbb{E}_{\mathcal{Q}_1^T|\boldsymbol{X}_1^T,\Lambda}$ denotes the conditional expectation over the hidden data, $\mathcal{Q}_1^T$, given the observation sequence, $\boldsymbol{X}_1^T$, and parameterized on the current model, Λ.

For an HMM based speech recognition system, the joint probability of the complete data can be written as,

$$\begin{aligned} p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\hat{\Lambda}) &= \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{x}_t), \text{or} \\ \log p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\hat{\Lambda}) &= \sum_{t=1}^{T} a_{q_{t-1}q_t} + \sum_{t=1}^{T} b_{q_t}(\boldsymbol{x}_t). \end{aligned} \tag{2.21}$$

Therefore, the auxiliary function can be written in terms of the model parameters associated

[3]The Q function guarantees an increase in the log-likelihood unless a maxima is reached [87].

with the state transition probabilities and observation probabilities,

$$Q(\Lambda, \hat{\Lambda}) = Q_{\boldsymbol{a}_i}(\Lambda, \hat{\boldsymbol{a}}_i) + Q_{\boldsymbol{b}_j}(\Lambda, \hat{\boldsymbol{b}_j}). \tag{2.22}$$

This allows separate maximization of the transition and observation parameters.

E step:
In the E step, the joint probability $p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\Lambda)$ is computed for all possible state and observation sequences. A forward-backward algorithm is used to compute this recursively. To this end, similar to the forward probability defined in Section 2.3.1, a *backward probability* is defined as,

$$\begin{aligned}\beta_i(t) &= p(\boldsymbol{X}_{t+1}^T|q_t = i, \Lambda) \\ &= \sum_{j=1}^{J} p(\boldsymbol{x}_{t+1}|q_{t+1} = j, \Lambda) p(q_{t+1} = j|q_t = i, \Lambda) p(\boldsymbol{X}_{t+2}^T|q_{t+1} = j, \Lambda) \\ &= \sum_{j=1}^{J} a_{ij} b_i(\boldsymbol{x}_{t+1}) \beta_j(t+1).\end{aligned} \tag{2.23}$$

$\beta_i(t)$ is the probability of observing $\boldsymbol{X}_{t+1}^T$ given that HMM is in state i at time t; for initialization, $\beta_i(T) = 1/J$. Using the forward and backward probabilities, one can write,

$$p(\boldsymbol{X}_1^T|\Lambda) = \sum_{j=1}^{J} \alpha_j(t)\beta_j(t). \tag{2.24}$$

To compute the parameter updates for HMM, another quantity, $\gamma_{(i,j)}(t)$, is defined, which refers to the probability of transitioning from state i to state j at time t given observation sequence and parameterized at the current model, *i.e.*,

$$\begin{aligned}\gamma_{(i,j)}(t) &= p(q_{t-1} = i, q_t = j|\boldsymbol{X}_1^T, \Lambda) \\ &= \frac{\alpha_i(t-1) a_{ij} b_j(\boldsymbol{x}_t) \beta_j(t)}{\sum_{j=1}^{J} \alpha_j(T)}\end{aligned} \tag{2.25}$$

The EM algorithm is used in speech recognition to update the parameters of the HMM model and those of the underlying model such as a GMM. For HMMs, $\boldsymbol{X}$ is observable but the state sequences $\mathcal{Q}$ is hidden. For GMMs, the mixture indices are hidden. The following

sections describe the estimation methods for the HMM and GMM parameters separately.

2.3.2.1 Updating HMM Transition Probabilities

Considering Eq. (2.20), (2.21) and (2.22), the auxiliary function for estimating the HMM transition probabilities can be written as,

$$Q_{\boldsymbol{a}_i}(\Lambda, \hat{\boldsymbol{a}}_i) = \sum_{t=1}^{T} \sum_{i=1}^{J-1} \sum_{j=1}^{J} \frac{p(\boldsymbol{X}_1^T, q_{t-1} = i, q_t = j|\Lambda)}{p(\boldsymbol{X}_1^T|\Lambda)} \log \hat{a}_{ij} \tag{2.26}$$

The updated transition probabilities are obtained by maximizing Eq. (2.26) with respect to the following constraints,

$$\sum_{j=1}^{J} a_{ij} = 1 \quad \forall i. \tag{2.27}$$

Using Lagrange multipliers based constrained optimization, the state transition probabilities for the new model are estimated as,

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{(i,j)}(t)}{\sum_{t=1}^{T} \sum_{k=1}^{J} \gamma_{(i,k)}(t)}. \tag{2.28}$$

2.3.2.2 Updating Observation Probabilities – Gaussian Mixture Models

A suitable and commonly used distribution for representing the observation probability vector, $b_j(\boldsymbol{x}_t)$, is the multivariate Gaussian distribution. The probability of observing a feature vector $\boldsymbol{x}_t$ in state j for an HMM model Λ is estimated as,

$$\begin{aligned} p(\boldsymbol{x}_t|q_t = j, \Lambda) &= \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ &= \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}_j)\right], \end{aligned} \tag{2.29}$$

where $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ denote the mean vector and covariance matrix of the j^{th} state, respectively.

In modern ASR, the observations are assumed to be derived from a continuous distribution and the resultant structures are referred to as continuous density HMMs (CDHMMs). A Gaussian distribution, however, provides a limited representation of the observation probabilities of a CDHMM. By combining multiple weighted Gaussian densities, one can obtain a more flexible and comprehensive distribution, Gaussian mixture models (GMM).

Historically, the GMM-HMM model, where each state of the CDHMM is modeled by a mixture of Gaussians, has been the most common architecture used in ASR. GMMs are particularly useful because they are capable of approximating any continuous probability density function [46]. For a GMM, the observation probabilities are given as,

$$\begin{aligned} b_j(\boldsymbol{x}_t) &= \sum_{m=1}^{M} c_{jm} N(\boldsymbol{x}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \\ &= \sum_{m=1}^{M} c_{jm} b_{jm}(\boldsymbol{x}_t) \end{aligned} \tag{2.30}$$

where M indicates the total number of Gaussians or mixture components in state j, each having its own density function $b_{jm}(\boldsymbol{x}_t)$ mean vector $\boldsymbol{\mu}_{jm}$ and covariance structure $\boldsymbol{\Sigma}_{jm}$. The mixture weight of the m^{th} Gaussian of j^{th} state is given by c_{jm} such that $0 \leq c_{jm} \leq 1$ and $\sum_{m=1}^{M} c_{jm} = 1$. In order to keep the number of parameters tractable, the components of a feature vector are assumed to be independent which results in diagonal covariance structures for the GMMs. This is a poor assumption for ASR; however, it helps in reducing the computational complexity of HMM based speech recognition. For MFCC features, this is achieved by virtue of DCT as discussed in Section 2.2.2. For other types of features, either the whitening principal component analysis (PCA) transform is used or the semi-tied covariance (STC) transform is used. STC is discussed in Section 2.6.3.

Similar to HMM, the GMM parameters updates are estimated using the iterative Baum-Welch or EM algorithm. In the presence of multiple Gaussian mixture components per

state, Eq. (2.21) can be written with respect to each mixture component as,

$$\begin{aligned} p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T|\hat{\Lambda}) &= \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{x}_t) \\ &= \prod_{t=1}^{T} a_{q_{t-1}q_t} \sum_{m_t=1}^{M} c_{q_t m_t} b_{q_t m_t}(\boldsymbol{x}_t) \\ &= \sum_{m_1=1}^{M} \sum_{m_2=1}^{M} \cdots \sum_{m_T=1}^{M} \left\{ \prod_{t=1}^{T} a_{q_{t-1}q_t} c_{q_t m_t} b_{q_t m_t}(\boldsymbol{x}_t) \right\} \\ &= \sum_{\mathcal{M}_1^T} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t m_t}(\boldsymbol{x}_t) c_{q_t m_t} \\ &= \sum_{\mathcal{M}_1^T} p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T, \mathcal{M}_1^T|\hat{\Lambda}), \end{aligned} \tag{2.31}$$

where $\mathcal{M}_1^T$ denotes all possible mixture components, and m_t indexes the mixture components corresponding to the state q_t. Similar to Eq. (2.20), an auxiliary function for the current and updated model is defined as,

$$Q(\Lambda, \hat{\Lambda}) = \sum_{\mathcal{Q}_1^T} \sum_{\mathcal{M}_1^T} \frac{p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T, \mathcal{M}_1^T|\Lambda)}{p(\boldsymbol{X}_1^T|\Lambda)} \log p(\boldsymbol{X}_1^T, \mathcal{Q}_1^T, \mathcal{M}_1^T|\hat{\Lambda}). \tag{2.32}$$

Similar to Eq. (2.22), the auxiliary function can be separated in three parts for the transition probabilities, component weights, and observation probabilities. The updates for the transition probabilities, $\hat{a}_{ij}$, have already been discussed in the previous section. The updates for the component weights, $\hat{c}_{jm}$, are obtained by maximizing the corresponding part with respect to the constraint $\sum_{m=1}^{M} c_{jm} = 1$ and are given as,

$$\hat{c}_{jm} = \frac{\sum_{t=1}^{T} \zeta_{jm}(t)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \zeta_{jm}(t)} \tag{2.33}$$

The updates for the Gaussian components' means and covariances are obtained by maximizing the auxiliary function's part corresponding to the observation probabilities with

respect to the constraint $\int_{-\infty}^{\infty} b_j(\boldsymbol{x}_t)d\boldsymbol{x}_t = 1$ and are given as,

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{t=1}^{T} \zeta_{jm}(t)\boldsymbol{x}_t}{\sum_{t=1}^{T} \zeta_{jm}(t)}, \tag{2.34}$$

$$\hat{\boldsymbol{\Sigma}}_{jm} = \frac{\sum_{t=1}^{T} \zeta_{jm}(t)(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_{jm})(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_{jm})^T}{\sum_{t=1}^{T} \zeta_{jm}(t)}, \tag{2.35}$$

where $\zeta_{jm}(t)$ is the probability of being in component m of state j at time t and is defined as,

$$\zeta_{jm}(t) = \frac{p(\boldsymbol{X}_1^T, q_t = j, m_t = m|\Lambda)}{p(\boldsymbol{X}_1^T|\Lambda)} = \frac{\sum_{i=1}^{J} \alpha_i(t-1)a_{ij}c_{jm}b_{jm}(\boldsymbol{x}_t)\beta_j(t)}{\sum_{i=1}^{J} \alpha_i(T)}. \tag{2.36}$$

GMMs are powerful tools for modeling distributions whose underlying distribution is unknown, as is the case in acoustic modeling. The distribution itself is parameterized by a balance between the number of mixtures, M, and estimation of observation probability distributions. A large value of M can lead to over-fitting, whereas, a smaller value may not produce a good representation of data. Modeling different states with different number of mixture components may be even more beneficial.

2.3.3 Decoding an HMM – Viterbi Decoding

Decoding an HMM refers to finding the best path or state sequence, $\hat{\mathcal{Q}}_1^T$, that maximizes the probability of generating a given observation sequence, $\boldsymbol{X}_1^T$. To this end, a dynamic programming algorithm known as Viterbi decoding is used [89]. Unlike to the forward algorithm discussed in Section 2.3.1 that sums probabilities from different paths coming to the same destination state, Viterbi algorithm chooses the best path at any time, t:

$$\begin{aligned} v_j(t) &= p(\boldsymbol{X}_1^t, \mathcal{Q}_t^{t-1}, q_t = j|\Lambda) \\ &= \max_{1\leq j\leq J}[a_{ij}v_i(t-1)]b_j(\boldsymbol{x}_t), \end{aligned} \tag{2.37}$$

where $v_j(t)$ is the probability of the partial sequence $\hat{\mathcal{Q}}_1^{t-1}$ that is most likely to generate the partial observation sequence $\mathcal{O}_1^T$ and end in state j at time t. To obtain the final best state sequence, a back-pointer is used to track the previous most likely states to generate

the partial observation sequence $\boldsymbol{X}_1^t$ and end in state j at time t,

$$B_j(t) = \arg\max_{1 \le j \le J}[a_{ij} v_i(t-1)], \tag{2.38}$$

where $B_j(t) = 0$. Finally, the best state sequence is retrieved as,

$$\hat{q}_t = B_{\hat{q}_{(t+1)}}(t+1). \tag{2.39}$$

The computational complexity of Viterbi decoding is $O(J^2T)$. It should be noted that in practice Viterbi algorithm can also be used as an approximate yet effective method to evaluate HMMs instead of the forward algorithm. This is further discussed in Section 2.5.

2.4 Language Model, Vocabulary and Lexicon

Previous sections of this chapter have discussed the acoustic modeling for CDHMM based speech recognition. This section presents the language model and related components of a speech recognition system. Section 2.4.1 discussed the language models used in CDHMM based speech recognition systems. Section 2.4.2 defines the vocabulary and lexicon.

2.4.1 Language Modeling of Speech

A language model is a statistical representation of the syntactic and pragmatic constraints essential to a language. It provides the *a-priori* probabilities, $p(\mathcal{W}_1^N)$, of a word sequence, $\mathcal{W}_1^N = w_1, w_2, \cdots, w_N$. Generally statistical models are used for this purpose. These models typically require a large corpus to obtain good probability estimates.

One such widely accepted model is the n-gram language model that assumes the language to obey an (n-1)$^{\text{th}}$ order Markov process. The probability of a word depends on its (n-1) predecessors, *i.e.*,

$$p(\mathcal{W}_1^N|\Lambda^{lm}) = \prod_{i=1}^{N} p(w_i|\mathcal{W}_{i-n+1}^{i-1}, \Lambda^{lm}), \tag{2.40}$$

where Λ^{lm} denotes a particular language model. Though higher order models, n, provide more structured constraints, they also increase the risk of encountering word sequences that are not seen in the training corpus. These sequences then have unknown probability. There are two common approach to deal with unseen sequences. The first is *discounting*, which

pre-allocates some probability mass for unseen sequences. The second is *back-off*, which start with a higher order model but backs off to a lower order if a given sequences cannot be allocated probability as per the higher order model.

2.4.2 Vocabulary and Lexicon

Vocabulary refers to the set of words that an ASR system learns and can recognize. During decoding, the recognizer selects the best word sequences from the vocabulary. The pronunciation or word to phoneme or other sub-word units mapping of these words are defined in the dictionary. An expansion is only needed if sub-word units based CDHMM models are used. The capacity of an ASR system is typically measured in terms of the size of vocabulary. Therefore, speech tasks are categorized as small, medium or large vocabulary tasks. The Aurora-2 task used in this thesis has a vocabulary size of 12 and is therefore a small vocabulary task [90]. The Aurora-4 task used in this thesis contains a vocabulary of 5000 words and is therefore referred to as a medium vocabulary task [91]. These tasks are further described in Section 2.8. A corpus with more than 10000 words is considered to be a large vocabulary task.

Vocabulary and the language model are very dependent on a given task domain. For example, an ASR system trained on a news corpus is not very likely to perform well in a medical setting. Words that a system has not seen during training are called out-of-vocabulary (OOV) words and negatively impact the performance of the recognizer. Having a very large vocabulary also increases the size of the search space for the recognizer. Therefore, it is important to choose the right size and contents of a vocabulary in order to achieve high performance from an ASR system.

2.5 Scoring and Global Search

The final piece of an ASR system is to combine the acoustic model probabilities, $p(\boldsymbol{X}|\mathcal{W})$ or $p(\boldsymbol{X}|\Lambda)$, and the language model probabilities, $p(\mathcal{W})$, in order to find the optimum word sequence that best represents the underlying acoustic sequence, $p(\mathcal{W}|\boldsymbol{X})$. This can be seen as a two-part process described below.

2.5.1 Local Acoustic Match

Local acoustic match refers to the observation probabilities associated with the states of a CDHMM model. As discussed in Section 2.3.2.2, GMM is one of the most suitable models for this purpose; some new models use a DNN instead [33]. Each of these probabilities or comparisons can be viewed as a local acoustic match. Thus, for a given feature vector, this module generates a vector of likelihoods for every possible state.

2.5.2 Delayed Decision Decoder – Viterbi Approximation

This module transforms the local hypotheses into a global decision. Typically, the acoustic model score or likelihood is given by the forward probability, as shown in Eq. (2.16), where the probability over all possible states sequences is summed. During the decoding, the *most likely word sequence* can be approximated by the *most likely state sequence*. This is achieved by approximating the summation with a maximum to find the best state sequence. With this approximation, Eq. (2.3) becomes,

$$\begin{aligned}\hat{\mathcal{W}} &= \arg\max_{\mathcal{W}} \{p(\boldsymbol{X}|\mathcal{W}) \quad \times \quad p(\mathcal{W})\} \\ &= \arg\max_{\mathcal{W}} \left\{ \sum_{\forall \mathcal{Q}} p(\boldsymbol{X}, \mathcal{Q}|\mathcal{W}) \quad \times \quad p(\mathcal{W}) \right\} \\ &\approxeq \arg\max_{\mathcal{W}} \left\{ \max_{\forall \mathcal{Q}} p(\boldsymbol{X}, \mathcal{Q}|\mathcal{W}) \quad \times \quad p(\mathcal{W}) \right\}, \end{aligned} \tag{2.41}$$

where $\approxeq$ signifies approximation. This equation is referred to as the Viterbi approximation for decoding.

This section completes the discussion on the building blocks of a modern ASR system. The next section discusses a number of well known approaches for feature-space transformations as applied to ASR.

2.5.3 Word Error Rate

The work in this thesis uses word error rate (WER) as the metric of evaluating the performance of ASR systems. WER is one of the most commonly used metric in the ASR community. It takes into account three kinds of errors in a speech recognition systems, namely substitution, deletion and insertion. For WER computation, first the recognized

word string is aligned against the correct word string, and then the WER is computed as,

$$WER = \frac{Subs + Dels + Ins}{\text{Number of words in the correct string}}. \tag{2.42}$$

The alignment algorithm is known as maximum substring matching problem. This is handled using dynamic programming [41]. Due to this alignment, WER can also be applied to continuous speech recognition systems. WER gives a fairly accurate measure of real-world performance of ASR systems.

2.6 Feature Space Transformations for ASR

As motivated in Section 1.2.2, finding a good feature representation is an important aspect of ASR. Conventional MFCCs are not able to capture the spectral evolution of speech. Finding a feature representation which is able to capture the spectral evolution while retaining the robustness of MFCCs is an active area of research in ASR. A number of techniques in this area build upon the MFCC framework by applying some sort of feature space transformations to high dimensional feature vectors derived from concatenating multiple speech frames. These transformations aim at reducing the dimensionality of the concatenated feature vectors and finding a feature space where it relatively easy to separate different speech classes.

This section provides brief summaries of two such frameworks, namely the discriminative and manifold learning based feature space transformations. Linear discriminant analysis (LDA) [47], [49] and locality preserving projections (LPP) [19] are presented as well known examples of discriminative and manifold based feature space projections, respectively. None of these techniques produce transformed features whose distributions are consistent with the diagonal covariance assumption of the GMM-HMM based ASR. The semi-tied covariance (STC) [92] procedure is presented as a means for reducing the impact of this mismatch.

The general problem of dimensionality reduction and feature space transformation can be defined as follows. Consider a set of labeled or unlabeled feature vectors represented in the form of a matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots \boldsymbol{x}_T]^\top$, where each row denotes a vector in the high-dimensional source space $\mathbb{R}^D$. For labeled data, each vector $\boldsymbol{x}_j$ would also be associated with a class/label $c(\boldsymbol{x}_j) \in \{c_1, c_2, \cdots, c_{M_c}\}$, where M_c is the number of classes, and each class c_i contains T_i

of the total T vectors[4]. This data, $\boldsymbol{X}$, (with class labels, if available) is referred to as the training set. The goal of a dimensionality reducing algorithm is to estimate the optimal projection matrix $\boldsymbol{P} \in \mathbb{R}^{D\times d}$, with $D \ll d$, to transform vectors from the D-dimensional source space onto an d-dimensional target space. The transformation is performed according to

$$\boldsymbol{y}_i = \boldsymbol{P}^\top \boldsymbol{x}_i \qquad \forall i = 1, 2, \cdots, T, \tag{2.43}$$

where $\boldsymbol{x}_i$ is an arbitrary vector in the source space, and $\boldsymbol{y}_i$ is the corresponding low dimensional vector in the target space.

2.6.1 Discriminative Techniques – Linear Discriminant Analysis

Discriminative algorithms, such as linear discriminant analysis (LDA) and heteroscedastic LDA (HLDA) [50], attempt to maximize the discrimination between classes of feature vectors. While HLDA has in some cases demonstrated performance improvements with respect to LDA, there is some debate as to whether similar effects can be achieved by applying a semitied covariance transform (STC) (discussed in Section 2.6.3) with LDA [92]–[94]. For this reason, LDA, in combination with STC, is selected as a representative of discriminant algorithms in this work. The following discussion provides a brief description of LDA.

Suppose that for the aforementioned training set, each class, c_i, is characterized by its mean vector, $\boldsymbol{\mu}_i$, and the covariance matrix, $\boldsymbol{\Sigma}_i$. The prior probability of each class is given by $p_i = T_i/T$. If $\boldsymbol{\mu}$ is the total sample mean of $\boldsymbol{X}$, then the within and between class scatter matrices are defined as [47],

$$\boldsymbol{S}_W = \sum_{i=1}^{M_c} p_i \boldsymbol{\Sigma}_i \tag{2.44a}$$

$$\boldsymbol{S}_B = \frac{1}{T} \sum_{i=1}^{M_c} \left(T_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top - \boldsymbol{\mu}\boldsymbol{\mu} \right). \tag{2.44b}$$

[4]From this section onwards, subscripts i and j are also used to index the feature vectors unlike t in previous sections. This is done for easily representing feature vectors at multiple time instances such as $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ instead of to having to write $\boldsymbol{x}_{t1}$ and $\boldsymbol{x}_{t2}$. The total number of feature vectors are still represented by T.

LDA optimizes a class separability criterion by maximizing the following objective function,

$$\boldsymbol{P}_{lda} = \arg\max_{\boldsymbol{P}} \left\{ tr(|\boldsymbol{P}^\top S_W \boldsymbol{P}|^{-1} |\boldsymbol{P}^\top S_B \boldsymbol{P}|) \right\}. \tag{2.45}$$

Eq. (2.45) can be solved as a generalized eigenvector problem as given in Eq. (2.46),

$$\boldsymbol{S}_B \boldsymbol{p}_{lda}^j = \lambda_j \boldsymbol{S}_W \boldsymbol{p}_{lda}^j \tag{2.46}$$

where $\boldsymbol{p}_{lda}^j$ is the j^{th} column of the LDA transformation matrix $\boldsymbol{P}_{lda}$, which is formed from the eigenvectors associated with the m largest eigenvalues. Further discussion of LDA can be found in [47].

It should be evident that the within-class scatter is a measure of the average variance of the data within each class, while the between-class scatter represents the average distance between the means of the data in each class and the global mean. Thus, LDA aims to preserve the global class relationships; however, it does not capture the intrinsic local structure of the underlying manifold.

2.6.2 Manifold Learning

As discussed in Section 1.1, manifold learning techniques assume the data points to lie on or close to the surface of one or more low dimensional manifolds. The underlying idea of manifold learning based feature transformations is to extend the manifold constrained relationships that exist among the input data vectors to the vectors in the projected space. Consider, for example, the set of four data points in Figure 2.6. Points A, B, C, and D are depicted to be lying on a 2-dimensional manifold represented by the curve. For neighboring points on the manifold, such as C and D, the closeness between the two points can be approximated by the Euclidean distance directly. However, for points that are well separated on the manifold, such as A and D, the direct Euclidean distance measured between the two points will be much different than the distance measured along the manifold curve. A simple dimensionality reduction algorithm may project the points A and D close together in the target space. However, a manifold learning transformation would project the points A and D far from each other, thus preserving the manifold based relationships, as illustrated in the figure.

Manifold based relationships can be characterized by a high dimensional graph connecting

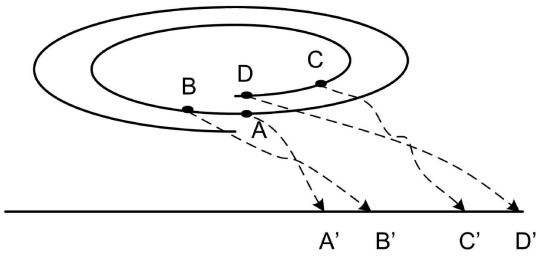

Fig. 2.6 Illustration of dimensionality reduction for two-dimensional data embedded in a nonlinear manifold space with relative position information preserved [14].

neighborhoods of feature vectors. This process is referred to as graph embedding (GE) [18]. In this graph, feature vectors, $\boldsymbol{X}$, correspond to nodes of the graph. The graph edge-weights denote the relationships among the nodes and are given by the affinity edge-weight matrix $\boldsymbol{\Omega} = [\omega_{ij}]_{T\times T}$, where the $\{i,j\}^{\text{th}}$ element of the affinity matrix, ω_{ij}, defines the weight of the edge connecting the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Such an embedding provides a strong mathematical framework to represent the distribution and geometrical structure of data.

For a generic graph $\mathcal{G}$, the relative scatter measure in the target space can be given by,

$$\begin{aligned}\mathcal{F}_{\mathcal{G}}(\boldsymbol{P}) &= \sum_{i,j=1}^{T} d\{f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)\}\omega_{ij} \\ &= \sum_{i,j=1}^{T} d\{\boldsymbol{y}_i, \boldsymbol{y}_j\}\omega_{ij} \end{aligned} \tag{2.47}$$

where $d\{\cdot,\cdot\}$ is a distance measure between two vectors. $\boldsymbol{y}_i = f(\boldsymbol{x}_i)$ is the vector in the target space that corresponds to the source-space vector $\boldsymbol{x}_i$; for linear transforms, $\boldsymbol{y}_i = \boldsymbol{P}^\top \boldsymbol{x}_i$. Depending on whether the goal is to preserve or discard the concerned graph properties, the optimal projection matrix, $\boldsymbol{P}$, can be obtained by minimizing or maximizing the scatter in Eq. (2.47). A detailed study and generalization of various graph embedding based techniques can be found in [18].

2.6.2.1 Locality Preserving Projections

Locality preserving projections (LPP) is chosen here as an example of manifold learning based feature space transformation. Following Eq. (2.47), the optimal projection matrix is obtained by minimizing a cost function related to Euclidean distances between the projected

feature vectors,

$$\mathcal{F}_{lpp}(\boldsymbol{P}) = \sum_{i,j=1}^{T} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i)||^2 \omega_{ij}, \tag{2.48}$$

$$\text{or, } \boldsymbol{P}_{lpp} = \arg\min_{\boldsymbol{P}} \sum_{i,j=1}^{T} ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2 \omega_{ij}, \tag{2.49}$$

$$\text{where } \omega_{ij} = \begin{cases} \exp\left(\frac{-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{\rho}\right) & ; \ \boldsymbol{x}_j \text{ is in the near neighborhood of } \boldsymbol{x}_i \\ 0 & ; \text{ Otherwise.} \end{cases} \tag{2.50}$$

The weight, ω_{ij}, is referred to as a Gaussian heat kernel, and ρ is a scale factor controlling the width of the kernel. The vector $\boldsymbol{x}_j$ is said to be in the neighborhood of $\boldsymbol{x}_i$ if it lies within the k-nearest neighbors of $\boldsymbol{x}_i$.

The optimal value of the projection matrix, $\boldsymbol{P}$, for minimizing the objective function in Eq. 2.49 can be obtained by solving the following general eigenvalue problem,

$$\boldsymbol{X}\boldsymbol{\mathcal{L}}\boldsymbol{X}^\top \boldsymbol{p}^j_{lpp} = \lambda \boldsymbol{X}\boldsymbol{\mathcal{D}}\boldsymbol{X}^\top \boldsymbol{p}^j_{lpp} \tag{2.51}$$

where $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{D}} - \boldsymbol{\Omega}$ is the Laplacian of the similarity matrix, $\boldsymbol{\mathcal{D}}$ is a diagonal matrix whose elements are the corresponding column sums of the matrix $\boldsymbol{\Omega}$, given by $\mathcal{D}_{ii} = \sum_{j=1}^{T} \omega_{ij}$.. The vector $\boldsymbol{p}^j_{lpp}$ is the j^{th} column of the linear transformation matrix, $\boldsymbol{P}_{lpp}$, which is formed from the eigenvectors associated with the d smallest non-zero eigenvalues. A general discussion of LPP can be found in [19], with ASR specific implementation in [14].

2.6.2.2 Manifold Regularization

Regularization is a common machine learning technique that is used to avoid over-fitting the training data and control the capability of a classifier. Typically, the objective function of the classifier is modified by adding a penalty term. Perhaps the most common form of regularization is L2-norm regularization, where the L2-norm of the model parameters is penalized in order to obtain a smooth decision function. Manifold regularization is a data dependent regularization framework that is capable of exploiting the underlying manifold based geometry of the data distribution. Manifold regularization framework is introduced in [39] as an approach for incorporating manifold learning based structural constraints into regularization based classification tasks. The authors have also presented manifold

extended versions of regularized least squares and support vector machines algorithms for a number of text and image classification tasks as example implementations of this framework. This section briefly describes two example implementations of the manifold regularization framework.

Laplacian Regularized Least Squares

Laplacian regularized least squares (LapRLS) is an extension of regularized least squares (RLS) classifier obtained by adding a manifold based regularization term [39]. Given a set of examples $(\boldsymbol{x}_i, c_i), i = 1, \ldots, T$ (c_i is the class label of $\boldsymbol{x}_i$), if $f : \boldsymbol{x} \rightarrow f(\boldsymbol{x})$ represents a mapping in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, then the fully supervised regularization framework estimates an unknown function by optimizing,

$$\min_{f \in \mathcal{H}} \frac{1}{T} \sum_{i=1}^{T} V(\boldsymbol{x}_i, c_i, f) + \gamma_1 ||f||_K^2 + \gamma_2 \frac{1}{T^2} \sum_{i,j=1}^{T} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||^2 \omega_{ij}, \tag{2.52}$$

where V is some loss function such as squared loss, $(c_i - f(\boldsymbol{x}_i))^2$, for regularized least square (RLS). K is a positive-semidefinite kernel function. The second part of Eq. (2.52) represents the norm on the original ambient space; this helps in maintaining smoothness for the assumed continuity of the source space. γ_1 is the corresponding regularization coefficient that controls the complexity of mapping in the ambient space. The last term in Eq. (2.52) represents the manifold based constraints. Note that it is similar to Eq. (2.48). γ_2 is the manifold regularization coefficient. Similar to the case of standard RLS, the final solution for the LapRLS is given as [39],

$$\boldsymbol{\alpha}^* = (\boldsymbol{J}\boldsymbol{K} + \gamma_1 T \boldsymbol{I} + \gamma_2 \boldsymbol{\mathcal{L}}\boldsymbol{K})^{-1} \boldsymbol{c} \tag{2.53}$$

where $\boldsymbol{J}$ is a diagonal identity matrix, $\boldsymbol{J} = diag(1, \ldots, 1)$ of the same dimensionality as the gram matrix $\boldsymbol{K}$; $\boldsymbol{c} = [c_1, \ldots, c_T]$ is the label vector. $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{D}} - \boldsymbol{\Omega}$ is the graph Laplacian defined over the manifold graph $\mathcal{G} = \{\boldsymbol{X}, \boldsymbol{\Omega}\}$, where $\boldsymbol{\mathcal{D}}$ is the diagonal sum matrix of $\boldsymbol{\Omega}$ given by $\mathcal{D}_{ii} = \sum_{j=1}^{T} \omega_{ij}$.

Laplacian Support Vector Machines

The manifold regularization framework can also be extended to other approaches, for example, to support vector machines (SVMs) to produce Laplacian SVMs (LapSVMs) in

[39]. LapSVMs extends conventional SVMs by adding a manifold regularization term to the hinge-loss function and solving for the objective criterion given as follows,

$$\min_{f\in\mathcal{H}} \frac{1}{T}\sum_{i=1}^{T}(1 - c_i f(\boldsymbol{x}_i)) + \gamma_1 ||f||_K^2 + \frac{1}{T^2}\gamma_2 \sum_{i,j=1}^{T} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||^2 \omega_{ij}. \tag{2.54}$$

Both LapRLS and LapSVM are primarily targeted at binary classification tasks where the goal is to obtain a closest to truth classification $f(\boldsymbol{x}_i) \cong c_i$ for every feature vector x_i. The manifold regularization framework can also be utilized for feature space transformations, where $f(\boldsymbol{x}_i)$ represents a target feature representation corresponding to $\boldsymbol{x}_i$. Similar work has been utilized for a phone recognition task in [10], however, no similar application to an ASR task exists. It is one of the goals of this dissertation to utilize similar formalisms for acoustic modeling for automatic speech recognition.

2.6.3 Semi-Tied Covariance

A common issue associated with all feature space transformation techniques when applied to ASR is the fact that the transformed features are not guaranteed to be statistically independent. Recall from Section 2.3.2.2 that GMM-HMM ASR systems assume the feature vector dimensions to be approximately uncorrelated and impose the diagonal covariance assumption on the continuous Gaussian observation densities. As a result, the distribution of feature vectors is mismatched with respect to the densities used for CDHMMs. Therefore, there is a need to incorporate one of a set of procedures that maximizes the likelihood of the data with respect to the diagonal Gaussian models. These procedures, including the maximum likelihood linear transformation (MLLT) [94] and semi-tied covariances (STC) [92], are applied in the transformed feature space. This work has adopted the STC approach for this purpose.

STC approximates full covariance modeling by allowing a number of full covariance matrices to be shared across many Gaussian components, instead of using component specific full covariance matrices. Effectively, each component maintains its own diagonal covariance. Each component consists of two elements, a component specific diagonal covariance matrix, $\boldsymbol{\Sigma}_{diag}^{(m)}$ and a semi-tied regression class dependent non-diagonal matrix, $\boldsymbol{A}^{(r)'}$. The form of

the resultant covariance matrix is given by

$$\boldsymbol{\Sigma}^{(m)} = \boldsymbol{A}^{(r)'}\boldsymbol{\Sigma}^{(m)}_{diag}\boldsymbol{A}^{(r)'\top} \tag{2.55}$$

where m specifies the corresponding mixture index, and r refers to the regression class. A detailed discussion of STC can be found in [92].

2.7 Artificial Neural Networks for ASR

GMM-HMM based ASR have been developed to the point where impressive speech recognition performance can be achieved for LVCSR tasks in laboratories. However, speech recognition is not a solved problem. Even the state-of-the-art ASR systems give low performance when applied to real life scenarios. Some researchers consider GMMs to be a limiting factor in the long-term goal of successful human-machine dialog. Furthermore, GMM-HMM based ASR requires unrealistic assumptions about speech, for instance, the uncorrelated features assumption discussed in Section 2.3. For these reasons, a number of researchers have started to look for alternate avenues for acoustic modeling.

One of the alternative approaches that have been used for this purpose is artificial neural networks (NNs) [52], [53]. Neural networks[5] are highly flexible structures capable of learning complex nonlinear relationships among data vectors. The most common NN architecture used for speech recognition is multi-layer perceptions (MLPs). MLPs are feed-forward neural networks with one visible input layer, one or more hidden layers, and one output layer, as illustrated in Figure 2.7. Each layer consists of a number of units that simulates the behavior of a neuron. The connections between units of different layers are governed by the weights connecting them. The weights connecting layer l to layer $l+1$ are given by matrix $\boldsymbol{W}^l$. The *activation* of a neuron unit is typically a nonlinear function of the weighted sum of its inputs; the activation of all the units in $(l+1)^{\text{th}}$ layer can be denoted in vector form as as $\boldsymbol{z}^{l+1} = f(\boldsymbol{W}^{l\top}\boldsymbol{z}^l)$, where $\boldsymbol{z}^l$ is the *activation* or *emission* vector of the l^{th} layer, and f is the *activation function* of the units. One of the most often used activation functions is

[5] In this work, the terms artificial neural networks, neural networks, neural nets, and nets are used interchangeably.

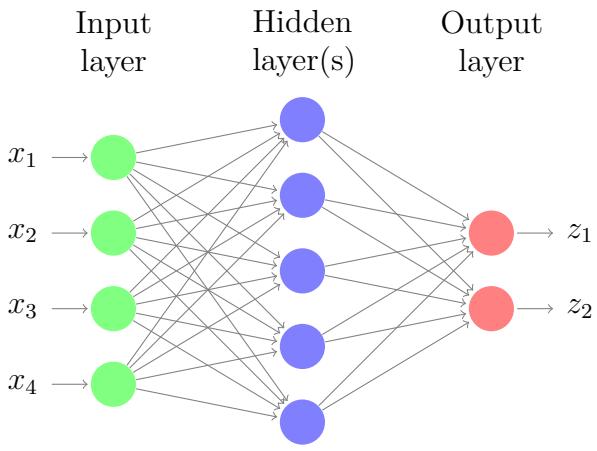

Fig. 2.7 A generic MLP. $\boldsymbol{x} = [x_1, \ldots, x_4]$ denotes the input vector, and $\boldsymbol{z} = [z_1, z_2]$ is the corresponding output vector. The outputs are posterior probabilities of the target labels if the net has been trained under certain conditions, as described in the text.

the sigmoid nonlinearity,

$$\boldsymbol{z}^{l+1} = f_s({\boldsymbol{W}^l}^\top \boldsymbol{z}^l) = \frac{1}{1 + \exp({\boldsymbol{W}^l}^\top \boldsymbol{z}^l)}. \tag{2.56}$$

In recent literature, rectified linear units (ReLU) has emerged as another important activation function for the application of neural networks to ASR [95],

$$\boldsymbol{z}^{l+1} = f_r({\boldsymbol{W}^l}^\top \boldsymbol{z}^l) = \max(0, {\boldsymbol{W}^l}^\top \boldsymbol{z}^l). \tag{2.57}$$

ReLU units have a number of advantages over sigmoids particularly for DNNs. These are discussed in detail in Section 6.4.1.

MLPs are trained to associate a desired output vector with an input vector. Nets with enough hidden units can, in principle, learn any mapping between the inputs and outputs. Typically, in ASR, these nets are used to produce a mapping $f_{mlp} : \boldsymbol{x} \rightarrow \boldsymbol{z}$ from the feature vectors at input nodes to the output activations. This mapping is gradually improved to minimize an error criterion, $V(\boldsymbol{x}, \boldsymbol{c}, f_{mlp})$, between the network outputs, $\boldsymbol{z}$, and the sub-word units or labels, $\boldsymbol{c}$, at output nodes with multiple iterations of learning over the training data. This is achieved by a mechanism referred to as *error back propagation* (EBP) that uses a gradient descent procedure to adjust the weights of the net in order to minimize a

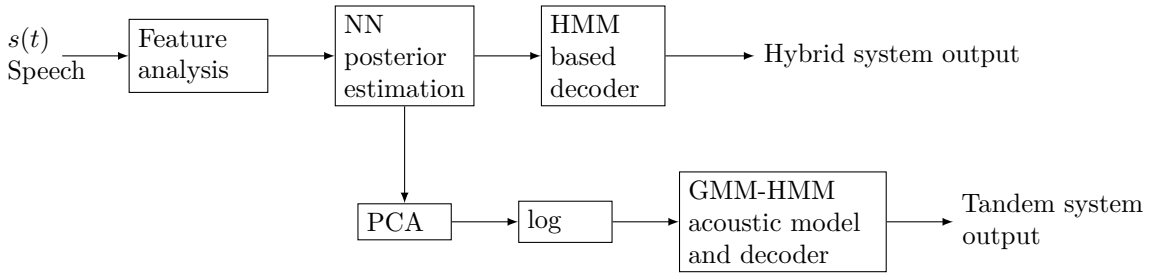

Fig. 2.8 Block diagram depicting applications of NNs to HMMs based ASR. Both hybrid NN-HMM architecture for acoustic modeling and tandem architecture for feature extraction are shown.

given error criterion. Many different criteria can be used for this purpose, for example, the minimum mean square error (MSE) between the net's outputs and target labels,

$$\mathcal{F}(\boldsymbol{W}) = \frac{1}{2}\sum_{i=1}^{M_c}(z_i - c_i)^2 = \frac{1}{2}||\boldsymbol{z} - \boldsymbol{c}||^2 \tag{2.58}$$

where M_c denotes the number of units in the output layer, which is equal to number of speech classes or sub-word unit labels in ASR. Minimizing the MSE is tantamount to increasing class based discrimination between feature vectors. $\boldsymbol{W}$ represents all the weights in the network. Net weights are initialized with random values and then changed using gradient-descent in the direction that will reduce the error in Eq. (2.58),

$$\boldsymbol{W}^l \leftarrow \boldsymbol{W}^l - \eta \nabla_{\boldsymbol{W}^l}\mathcal{F} \tag{2.59}$$

where η is the learning rate for gradient descent.

MLPs and other neural net architectures have been used in a variety of ways for speech processing. These approaches are collectively referred to as connectionist speech recognition [52]. Two of the common approaches to apply NNs to speech are discussed in this section. A block diagram of these system is given in Figure 2.8.

2.7.1 Hybrid modeling – NNs for Acoustic Modeling

The hybrid approaches attempt to replace GMMs by NNs for modeling the observation probabilities, $b_j(\boldsymbol{x}) = p(\boldsymbol{x}|q_t = j)$, in HMM based speech recognition. To this end, NNs are

used for discriminative estimation of HMM state posterior probabilities given the acoustic data, $p(q_t = j|\boldsymbol{x})$. The posteriors are then converted into observation probabilities by normalizing them using state prior probabilities following Bayes Rule. This enables NNs to be easily integrated into the HMM based approach to speech recognition, as shown in Figure 2.8. Such a system is usually referred to as a NN-HMM system.

There are many advantages in using hybrid NN-HMM systems instead of GMM. NN-HMM based hybrid approaches do not make any assumptions about independence of speech features. Unlike GMM-HMM systems which are based on likelihood maximization, NN-HMM systems directly estimate the state posteriors to maximize discrimination between speech classes.

Traditionally, only NNs with single hidden layer have been used in ASR, primarily because it is hard to train nets with higher number of layers. In recent years, a crucial development in the hybrid NN-HMM ASR has been the successful training of NNs with many hidden layers [33], [54]. These algorithms train hybrid DNN-HMM models.

2.7.2 Tandem – NNs for Feature Estimation

NNs can also be used to derive features for conventional GMM-HMM system. Such a combination is referred to as a *tandem* architecture [31], [32]. Figure 2.8 shows a block diagram of a tandem speech recognition system. Tandem approaches train neural nets to generate speech class posteriors for a given feature vector, and transforms these estimates as features for conventionally-trained GMM-HMM system. These approaches have demonstrated large improvements in word recognition performance [31], [96]. The tandem approaches thus incorporate the discriminative power of neural networks into feature extraction, while taking advantage of the large quantity of sophisticated tools and techniques available for GMM-HMM based ASR systems.

2.8 Task Domains and Baseline Systems

This section describes the speech corpora used to evaluate the ASR performances of the various techniques discussed in this thesis. Two different speech in noise corpora are discussed. The first is the Aurora-2 connected digits task [90], and the second is the Aurora-4 read newspaper task [91]. A summary of the baseline ASR systems along with details of both of these corpora are given here.

For all the experiments in this thesis, the baseline CDHMM ASR systems are trained using 12-dimensional static MFCC features augmented by normalized log energy, difference cepstrum, and second difference cepstrum resulting in 39-dimensional vectors. The ASR performance is reported in terms of WER. These GMM-HMM systems are also used for generating the state alignments. These alignments are then used as the target labels for deep networks training in related experiments.

Aurora-2 is a connected digit speech in noise corpus. In most of the experiments the Aurora-2 mixed-conditions training set is used for training [90]. Some experiments have also used the clean training set. Both the clean and mixed-conditions training sets contain a total of 8440 utterances by 55 male and 55 female speakers. In the mixed-conditions set, the utterances are corrupted by adding four different noise types to the clean utterances. The baseline ASR system for the Aurora-2 set is configured using the standard configuration specified in [90]. This corresponds to using 10 word-based CDHMMs for digits 0 to 9 with 16 states per word-model, and additional models with 3 states for silence and 1 state for short-pause. In total, there are 180 CDHMM states each modeled by a mix of 3 Gaussians. During the test phase, four different subsets are used corresponding to uncorrupted clean utterances and utterances corrupted with four different noise types, namely subway, car, train station and exhibition hall, at SNRs ranging from 5 to 20 dB. There are 1001 utterances in each subset. The ASR performance obtained for the baseline system configuration agrees with those reported elsewhere [90].

The second dataset used in this work is the Aurora-4 read newspaper speech-in-noise corpus. This corpus is created by adding noise to the Wall Street Journal corpus [91]. Aurora-4 represents a MVCSR task with a vocabulary size of 5000 words. This work uses the standard 16kHz mixed-conditions training set of the Aurora-4 [91]. It consists of 7138 noisy utterances from a total of 83 male and female speakers corresponding to about 14 hours of speech. One half of the utterances in the mixed-conditions training set are recorded with a primary Sennheiser microphone and the other half with a secondary microphone, which enables the effect of transmission channel. Both halves contain a mixture of uncorrupted clean utterances and noise corrupted utterances with the SNR levels varying from 10 to 20 dB in 6 different noise conditions (babble, street traffic, train station, car, restaurant and airport). A bi-gram language model is used with a perplexity of 147. Context-depedent cross-word triphone CDHMM models are used for configuring the baseline ASR system, and each CDHMM state is modeled by a mixture of 16 Gaussian components. Silence is

modeled by three state HMM models, and inter-word short pauses are modeled by single state models. Similar to the training set, the test set is recorded with the primary and secondary microphones. Each subset is further divided into seven subsets, where one subset is clean speech data and the remaining six are obtained by randomly adding the same six noise types as training at SNR levels ranging from 5 to 15 dB. Thus, there are a total of 14 subsets. Each subset contains 330 utterances from 8 speakers. The ASR performance for the baseline system agrees with those given in [91].

2.9 Conclusion

This chapter has provided a review of the common background required for the rest of this dissertation. Basic architecture and building blocks of a state of the art HMM system has been discussed. MFCC based common feature extraction scheme has been discussed along with the dominant GMM-HMM based ASR framework. LDA and LPP transformations have been discussed as examples of feature space transformation techniques. Finally, the application of NNs, shallow as well as deep, in connectionist speech recognition has been discussed.

Chapter 3

A Family of Discriminative Manifold Learning Techniques

The speech production process is constrained by limited dynamics of the articulators. It has been suggested that constrained by these limitations, the acoustic feature space is confined to lie on low dimensional nonlinear manifold [10], [13], [51]. Therefore, a feature space transformation technique that explicitly models and preserves the local relationships of data along the underlying manifold should be more effective for speech processing. Accordingly, multiple studies have demonstrated gains in ASR performance when using features derived from a manifold learning based approach (See Section 1.2.3 and 2.6.2). Manifold learning approaches such as LPP, discussed in Section 2.6.2, attempt to preserve local relationships among data vectors in the transformed space [14], [19]. However, most these approaches are unsupervised by nature, and only focus on preserving the manifold data similarities and fail to discover the discriminant structure of the data.

This chapter investigates a new supervised discriminative manifold learning framework for feature space transformation as applied to ASR. DML extends the conventional manifold learning approaches by incorporating a discriminative component. This results in a framework that attempts to maximize the separability between different classes while preserving the within-class local manifold constrained relationships of the data points. Techniques with some notion of DML have been used in other application domains, however, no such technique have been used for speech processing [18], [20]. The research presented here shows

[0]Parts of this chapter have been published in [15]–[17].

that these algorithms provide significant improvements in ASR WER for a speech in noise task as compared to well-known techniques such as LDA and LPP.

Assuming that the data distribution is supported on a low dimensional manifold, an optimality criterion is formulated based on manifold preservation and inter-class separability. In other words, the goal in DML is to estimate the parameters of a feature space mapping or projection $f : \boldsymbol{x} \in \mathbb{R}^D \rightarrow \boldsymbol{y} \in \mathbb{R}^d$, with $d \leq D$ (generally $d \ll D$), that maximizes the sub-manifold discrimination in the projected feature space while retaining the within-sub-manifold inherent data relations. A mapping designed in this manner exploits the underlying manifold domain geometry of the input distribution as well as the discriminative structure of the feature space. The geometrical relationships between feature vectors along this manifold are characterized by two partially-connected weighted graphs, namely intrinsic and penalty graphs [15], [18]. If the feature vectors, $\boldsymbol{X} \in \mathbb{R}^{T \times D}$, are represented by the nodes of the graphs, the intrinsic graph, $\mathcal{G}_{int} = \{\boldsymbol{X}, \boldsymbol{\Omega}_{int}\}$, characterizes the within-class or within-manifold relationships between feature vectors. The penalty graph, $\mathcal{G}_{pen} = \{\boldsymbol{X}, \boldsymbol{\Omega}_{pen}\}$, characterizes the relationships between feature vectors belonging to different speech classes. Therefore, the intrinsic graph corresponds to the relationships between the feature vectors that are to be preserved during the mapping, whereas the penalty graph refers to the relationships that are to be discarded.

The manifold domain relationships between graph nodes or feature vectors are characterized by the elements of the affinity matrices, $\boldsymbol{\Omega}_{int}$ and $\boldsymbol{\Omega}_{pen}$. For two arbitrary nodes $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$, the weight on the edge connecting them is given by the $\{u, v\}^{\text{th}}$ element, ω_{uv}, of the corresponding affinity matrix. It follows that the characteristics of a graph such as structure, connectivity and compactness are governed by these weights. These weights and the graph based relationships can be defined in terms of many different metrics. This work uses two different metrics to define the affinity weights. This leads to two different approaches of DML. The first, locality preserving discriminant analysis (LPDA), defines affinity between nodes in terms of a Euclidean distance metric [16]. The second approach uses a cosine-correlation distance metric to define the manifold domain affinity between nodes and is referred to as correlation preserving discriminant analysis (CPDA) [17]. The use of a cosine-correlation based distance metric is motivated by work where cosine distance metrics have been found to be more robust to noise corruption than those based on Euclidean distances [9], [20], [72]. For this reason, it is expected of CPDA to demonstrate a performance advantage over LPDA for high noise scenarios. These algorithms are described in the following sections.

3.1 Locality Preserving Discriminant Analysis

In LPDA, the elements of the intrinsic and penalty graph weight matrices are defined in terms of a Euclidean distance based Gaussian heat kernel as,

$$\omega_{ij}^{int} = \begin{cases} \exp\left(\frac{-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{\rho}\right) & ; c(\boldsymbol{x}_i) = c(\boldsymbol{x}_j), e(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 \\ 0 & ; \text{Otherwise} \end{cases} \tag{3.1a}$$

$$\omega_{ij}^{pen} = \begin{cases} \exp\left(\frac{-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{\rho}\right) & ; c(\boldsymbol{x}_i) \neq c(\boldsymbol{x}_j), e(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 \\ 0 & ; \text{Otherwise} \end{cases} \tag{3.1b}$$

where $c(\boldsymbol{x}_i)$ refers to the class or label of vector $\boldsymbol{x}_i$. The indicator function $e(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is true if $\boldsymbol{x}_i$ lies in the near neighborhood of $\boldsymbol{x}_j$. Closeness to a vector $\boldsymbol{x}_i$ can be measured either by k-nearest neighbors (kNN) or neighbors within certain radius r. ρ is the kernel heat parameter. In the intrinsic graph, $\mathcal{G}_{int}$, a node $\boldsymbol{x}_i$ is connected to the k_{int} nearest neighbors belonging to the same class $C(\boldsymbol{x}_i)$. Similarly, in the penalty graph, $\mathcal{G}_{pen}$, a node $\boldsymbol{x}_i$ is connected to the k_{pen} largest affinity neighbors *not* belonging to the class $c(\boldsymbol{x}_i)$.

Following Eq. (2.47), a scatter measure for a generic graph $\mathcal{G}$ in LPDA is defined in terms of the feature vectors in the transformed space, $\boldsymbol{y}_i = \boldsymbol{P}^\top \boldsymbol{x}_i$, and is given by,

$$\begin{aligned} \mathcal{F}_{\mathcal{G}}(\boldsymbol{P}) &= \sum_{i,j} ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2 \omega_{ij} \\ &= \sum_{i,j} (\boldsymbol{y}_i^2 + \boldsymbol{y}_j^2 - 2\boldsymbol{y}_i\boldsymbol{y}_j)\omega_{ij} \qquad (3.2) \\ &= \sum_i \boldsymbol{y}_i^2 \mathcal{D}_{ii} + \sum_j \boldsymbol{y}_j^2 \mathcal{D}_{jj} - 2\sum_{i,j} \boldsymbol{y}_i\boldsymbol{y}_j\omega_{ij} \\ &= 2\boldsymbol{P}^\top \boldsymbol{X}(\boldsymbol{\mathcal{D}} - \boldsymbol{\Omega})\boldsymbol{X}^\top \boldsymbol{P} \\ &= 2\boldsymbol{P}^\top \boldsymbol{X}\boldsymbol{\mathcal{L}}\boldsymbol{X}^\top \boldsymbol{P} \qquad (3.3) \end{aligned}$$

where $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{D}} - \boldsymbol{\Omega}$ is the $\mathcal{L}$aplacian matrix of the graph $\mathcal{G}$, and $\boldsymbol{\mathcal{D}}$ is the diagonal degree matrix whose elements correspond to the column sum of the edge-affinity matrix $\boldsymbol{\Omega}$, *i.e.*, $\mathcal{D}_{ii} = \sum_j \omega_{ij}$.

In LPDA, the goal is to estimate a projection matrix, $\boldsymbol{P}_{lpda} \in \mathbb{R}^{D\times d}$, that maximizes discrimination in the projected feature space while preserving the within-manifold local

relationships between feature vectors. It follows that the projection should maximize the scatter of the penalty graph $\mathcal{F}_{pen}(\boldsymbol{P})$, while at the same time minimize the scatter of the intrinsic graph $\mathcal{F}_{int}(\boldsymbol{P})$. To this end, a measure of class separability and graph-preservation is defined in terms of the ratio of $\mathcal{F}_{pen}(\boldsymbol{P})$ to $\mathcal{F}_{int}(\boldsymbol{P})$ as,

$$\begin{aligned}\mathcal{F}(\boldsymbol{P}) &= \frac{\mathcal{F}_{int}(\boldsymbol{P})}{\mathcal{F}_{pen}(\boldsymbol{P})} \\ &= \frac{\boldsymbol{P}^\top \boldsymbol{X}\boldsymbol{L}^{int}\boldsymbol{X}^\top \boldsymbol{P}}{\boldsymbol{P}^\top \boldsymbol{X}\boldsymbol{L}^{pen}\boldsymbol{X}^\top \boldsymbol{P}},\end{aligned} \tag{3.4}$$

where the superscripts *int* and *pen* signify 'intrinsic' and 'penalty' graphs, respectively. Thus, an optimal projection matrix is the one to minimize the expression in Eq. (3.4), *i.e.*,

$$\boldsymbol{P}_{LPDA} = \arg\min_{\boldsymbol{P}} \mathcal{F}(\boldsymbol{P}), \tag{3.5}$$

Or,

$$\arg\min_{\boldsymbol{P}} \left\{ \mathrm{tr}\left((\boldsymbol{P}\boldsymbol{X}\boldsymbol{\mathcal{L}}^{pen}\boldsymbol{X}^\top \boldsymbol{P})^{-1} (\boldsymbol{P}^\top \boldsymbol{X}\boldsymbol{\mathcal{L}}^{int}\boldsymbol{X}^\top \boldsymbol{P}) \right) \right\}. \tag{3.6}$$

Expression in 3.6 can be simplified into a generalized eigenvalue decomposition problem given as follows,

$$(\boldsymbol{X}\boldsymbol{\mathcal{L}}^{int}\boldsymbol{X}^\top)\boldsymbol{p}^j_{lpda} = \lambda_j (\boldsymbol{X}\boldsymbol{\mathcal{L}}^{pen}\boldsymbol{X}^\top)\boldsymbol{p}^j_{lpda}, \tag{3.7}$$

where the vector $\boldsymbol{p}^j_{lpda}$ forms the j^{th} column of the projection matrix $\boldsymbol{P}_{lpda} \in \mathbb{R}^{D\times d}$ and is the eigenvector associated with the j^{th} smallest eigenvalue. For a projection on to a d-dimensional space, the eigenvectors corresponding to the d smallest eigenvalues constitute the optimal LPDA projection matrix, $\boldsymbol{P}_{lpda}$.

3.2 Correlation Preserving Discriminant Analysis

This section discusses the second DML technique proposed in this work, referred to as correlation preserving discriminant analysis (CPDA). Unlike LPDA, where a Euclidean distance metric is used to characterize the manifold based relationships between feature vectors, CPDA uses a cosine-correlation based distance measure for this purpose.

The motivation for using a cosine-correlation based distance measure arises from studies indicating that the magnitude of cepstrum vectors and hence the Euclidean distances based acoustic models are highly susceptible to ambient noise [97]. Another way to visualize this

could be that noise causes the data points to scatter around their original position. As a result, the unseen test features may not obey the structure of the manifold learned from the training data during estimation of the feature space transformation. These factors contribute to misclassification during the feature space transformation. It has also been suggested that the angles between cepstrum vectors are comparatively more robust to noise [98], [99]. Furthermore, in many cases of nonparametric learning, Euclidean distance-based methods cannot clearly explain the distributional structure of the data. A more general case adapted to the nature of data is to consider the correlation between data vectors [9], [20], [72], which indicates the strength and direction of a relationship between two data points. This implies a potential advantage to a feature space transformation where relationships between feature vectors are defined using a cosine-correlation based distance measure, particularly in the context of noise robust ASR.

The goal in CPDA is to preserve the correlational relationships between feature vectors during the transformation while discriminating between classes of speech feature vectors. CPDA defines a projection of the feature vectors from the source d-dimensional hypersphere onto the target m-dimensional hypersphere. Note that this results in a nonlinear projection, where the projected features are given by $\boldsymbol{y}_i = \boldsymbol{P}^\top \boldsymbol{x}_i / ||\boldsymbol{P}^\top \boldsymbol{x}_i||$. In order to define a correlation based measure, first the feature vectors are projected onto the surface of a unit hypersphere. The algorithm formulation for CPDA is very similar to that for LPDA. The feature vectors are embedded into two graphs, namely, $\mathcal{G}_{int} = \{\boldsymbol{X}, \boldsymbol{\Omega}_{int}\}$ and $\mathcal{G}_{pen} = \{\boldsymbol{X}, \boldsymbol{\Omega}_{pen}\}$, where the elements of the affinity matrices are defined in terms of the cosine-correlation between feature vectors,

$$\omega_{ij}^{int} = \begin{cases} \exp\left(\frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - 1}{\rho}\right) & ; c(\boldsymbol{x}_i) = c(\boldsymbol{x}_j), e_c(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 \\ 0 & ; \text{Otherwise} \end{cases} \tag{3.8a}$$

$$\omega_{ij}^{pen} = \begin{cases} \exp\left(\frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - 1}{\rho}\right) & ; c(\boldsymbol{x}_i) \neq c(\boldsymbol{x}_j), e_c(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 \\ 0 & ; \text{Otherwise.} \end{cases} \tag{3.8b}$$

Variables in Eq. (3.8) follow the same nomenclature as those in Eq. (3.1) except that the similarity or nearest neighbors are defined in a cosine-correlation sense.

Following the cosine-correlation distance metric, a scatter measure for a generic graph $\mathcal{G}$ on the surface of the target hypersphere is defined as a function of the cosine correlation

among its nodes as,

$$\begin{aligned} F_{\mathcal{G}}(\boldsymbol{P}) &= \sum_{i\neq j} ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2 \omega_{ij} \\ &= \sum_{i\neq j} \left\| \frac{\boldsymbol{P}^\top \boldsymbol{x}_i}{||\boldsymbol{P}^\top \boldsymbol{x}_i||} - \frac{\boldsymbol{P}^\top \boldsymbol{x}_j}{||\boldsymbol{P}^\top \boldsymbol{x}_j||} \right\|^2 \omega_{ij} \\ &= 2\sum_{i\neq j} \left(1 - \frac{f_{ij}}{f_i f_j}\right) \omega_{ij}, \end{aligned} \tag{3.9}$$

where, for two arbitrary vectors $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$, $f_u = \sqrt{\boldsymbol{x}_u^\top \boldsymbol{P}\boldsymbol{P}^\top \boldsymbol{x}_u}$, and $f_{uv} = \boldsymbol{x}_u^\top \boldsymbol{P}\boldsymbol{P}^\top \boldsymbol{x}_v$. Following the DML framework, the goal in CPDA is to minimize the scatter of the intrinsic graph, $\mathcal{F}_{int}(\boldsymbol{P})$, while at the same time maximize the scatter of the penalty graph, $\mathcal{F}_{pen}(\boldsymbol{P})$. Note that because of the nonlinear nature of CPDA projection, the objective criterion cannot be formulated as the ratio of the scatter measures of the two graphs. Therefore, the difference of the intrinsic and penalty graph scatters is defined as a measure of manifold domain locality preservation and class separability,

$$\mathcal{F}(\boldsymbol{P}) \;=\; \mathcal{F}_{int}(\boldsymbol{P}) - \mathcal{F}_{pen}(\boldsymbol{P}) \;=\; 2\sum_{i\neq j} \left(1 - \frac{f_{ij}}{f_i f_j}\right) \cdot \omega_{ij}^{int-pen}, \tag{3.10}$$

where $\omega_{ij}^{pen-int} = \omega_{ij}^{int} - \omega_{ij}^{pen}$. The optimal projection matrix is the one to minimize the above function, *i.e.*,

$$\boldsymbol{P}_{CPDA} = \arg\min_{\boldsymbol{P}} \mathcal{F}(\boldsymbol{P}). \tag{3.11}$$

The objective criterion defined in Eq. (3.10) cannot be formulated as a generalized eigenvalue problem instead the optimal CPDA projection matrix is obtained by using the gradient descent rule as follows,

$$\begin{aligned} P \leftarrow & P - \eta \nabla_P \mathcal{F}, \text{with} \\ \nabla_P \mathcal{F} = & 2\sum_{i\neq j} \left[\frac{f_{ij}\boldsymbol{x}_i\boldsymbol{x}_i^\top}{f_i^3 f_j} + \frac{f_{ij}\boldsymbol{x}_j\boldsymbol{x}_j^\top}{f_i f_j^3} - \frac{\boldsymbol{x}_i\boldsymbol{x}_j^\top + \boldsymbol{x}_j\boldsymbol{x}_i^\top}{f_i f_j} \right] \boldsymbol{P} \cdot \omega_{ij}^{int-pen}, \end{aligned} \tag{3.12}$$

where η represents the gradient scaling factor, and $\nabla_P \mathcal{F}$ is the gradient of the objective criterion defined in Eq. (3.10) with respect to $\boldsymbol{P}$.

The objective criterion for $\mathcal{F}(\boldsymbol{P})$ corresponds to a nonlinear and non-convex error surface. This makes the gradient descent susceptible to converge to a local optima. This could be avoided by provided a good initialization for the parameters of the CPDA projection matrix. To this end, an initial projection is achieved by neglecting the normalization of the transformed features to approximate to a linear solution, *i.e.*, by setting $\boldsymbol{y}_i = \boldsymbol{P}^\top \boldsymbol{x}_i$. A closed-form solution for the initialization is then achieved by solving a generalized eigenvalue problem similar to one in the case of LPDA [16], [18],

$$(\boldsymbol{X}\boldsymbol{\mathcal{L}}^{int}\boldsymbol{X}^\top)\boldsymbol{p}_j = \lambda_j(\boldsymbol{X}\boldsymbol{\mathcal{L}}^{pen}\boldsymbol{X}^\top)\boldsymbol{p}_j, \tag{3.13}$$

where $\boldsymbol{\mathcal{L}}$ is a Laplacian matrix of the corresponding graph defined in a manner similar to that in Eq. (3.3). The superscripts *int* and *pen* signifies 'intrinsic' and 'penalty' matrices respectively. The vector $\boldsymbol{p}_j$ indicates the j^{th} column of the initial transformation matrix $\boldsymbol{P}$. Subsequent projection matrices are obtained by performing gradient descent as per Eq. (3.12) for a suitable number of iterations or until convergence is reached.

3.3 Experimental Study

This section describes the experimental study performed to evaluate the ASR performance of the DML approaches. Section 3.3.1 provides details of the task domain along with a summary of the CDHMM ASR systems used in the experiments. Section 3.3.2 presents comparisons of ASR performance of features obtained using LPDA and CPDA to those obtained for the more well known techniques, LDA and LPP, over a range of noise types and signal-to-noise ratios (SNRs). Furthermore, the impact of using the cosine-correlation based distance measure is evaluated by comparing the ASR performance of CPDA features to that of LPDA features.

3.3.1 System Configuration

The experiments in this chapter are conducted on the Aurora-2 and Aurora-4 speech in noise tasks described in the Section 2.8. The baseline CDHMM ASR systems are trained using 12-dimensional static Mel-frequency cepstrum coefficient (MFCC) features augmented by normalized log energy, difference cepstrum, and second difference cepstrum as described in the Section 2.8.

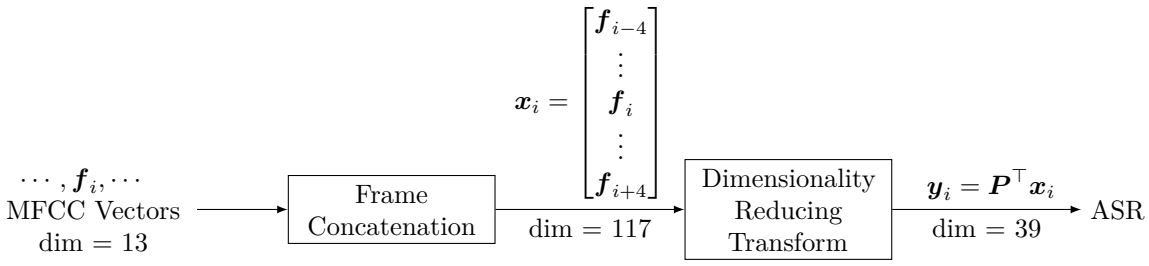

Fig. 3.1 System setup for dimensionality reducing feature space transformations for ASR. Multiple consecutive MFCC vectors are concatenated in order to capture the time evolution of speech spectrum. This concatenated vector is then projected on to a lower dimensional space using a linear transform. The output low-dimensional features are fed into an ASR system.

The system setup for the dimensionality reducing feature space transformations is demonstrated in Figure 3.1. At the input, 117-dimensional super-vectors are obtained by concatenating 9 frames of MFCCs augmented with log energy. This allows one to effectively capture the time evolution of speech spectrum. The 117-dimensional vectors are then transformed onto a 39-dimensional space by means of a projection matrix $\boldsymbol{P}$.

For both Aurora-2 and Aurora-4 datasets, a neighborhood size of $k = k_{int} = k_{pen} = 200$ is chosen for estimating the intrinsic and penalty graph weights. Values of the Gaussian kernel heat factor, ρ, have been chosen separately for each of the three manifold learning approaches LPP, LPDA and CPDA. The values used for ρ are: 900 for LPP, 1000 for intrinsic and 3000 for penalty graph for LPDA, and 10^{-2} for the intrinsic and penalty graphs of CPDA. These values are empirically determined using a development corpus. Note that the same choices of the kernel scale factor and number of neighbors are used for both datasets. Semitied Covariance transformations (STC), described in Section 2.6.3 and [92], are applied prior to recognition to account for the correlation introduced to the transformed features by the LDA, LPP, LPDA, and CPDA projections.

3.3.2 Results for Aurora-2 Connected Digit Corpus

The ASR WER comparisons of feature obtained from the proposed LPDA and CPDA techniques with that of features from LDA and LPP are provided in Table 3.1. The table provides ASR results for three noise types at a range of SNRs ranging from 5dB to 20dB. Four separate tables are displayed, one for each noise type (subway, exhibition hall and car), and one for average over all noise types. In each of these tables, ASR WERs for five different

systems are given. The first row in the tables displays the ASR WER for the baseline system. In this case, the MFCC features obtained from Aurora-2 mixed condition set are used for HMM training without the application of any feature space transformation. The baseline WERs displayed in Table 3.1 agree with those reported elsewhere [90], [100]. The results in the second row, labeled "LDA", correspond to feature obtained from the application of a LDA projection matrix to concatenated MFCC feature vectors as illustrated in Figure 3.1. Similarly, the third row, labeled "LPP", gives the WER results for features obtained as a result of applying a LPP transformation. The results for the technique presented in this chapter are given in the next two rows. The fourth row "LPDA" corresponds to the ASR WERs performance of the features obtained by applying the LPDA transformation to the concatenated super-vectors. The final row, labeled "CPDA", displays the ASR WERs for feature when CPDA is used as the feature space transformation technique.

It should be noted that STC transformations are performed to minimize the impact of the data correlation resulting from the application of the feature space transformations in all cases except that of the baseline features. This is important because ASR performances of all of the discussed transformations degrade if STC is not applied. For example, for the 20 dB subway noise case, the WER for LDA features increases by 40% relative if STC is not applied [16]. This is consistent with the discussion in Section 2.6.3 and results that other researchers have obtained when comparing the performance of feature space transformations with and without STC [16], [94], [101], [102].

Noise	Technique	SNR (dB)				
		clean	20	15	10	5
Subway	Baseline	1.76	2.99	4.00	6.21	11.89
	LDA	1.82	2.25	2.93	5.29	12.32
	LPP	1.66	2.33	3.50	5.71	13.26
	LPDA	**1.57**	**2.18**	3.29	5.28	11.73
	CPDA	1.60	2.30	**2.91**	**4.54**	**11.24**
Exhibition	Baseline	1.89	3.34	3.83	6.64	12.72
	LDA	1.83	2.63	3.37	6.67	14.29
	LPP	1.76	2.56	4.23	8.55	16.91
	LPDA	**1.23**	**2.22**	3.64	6.66	13.85
	CPDA	1.25	2.30	**2.95**	**5.37**	**12.59**
Car	Baseline	1.99	2.77	3.36	5.45	**12.31**
	LDA	2.29	2.83	3.45	5.69	15.92
	LPP	1.88	2.71	3.61	6.08	14.97
	LPDA	1.52	**2.30**	**2.77**	**5.19**	12.73
	CPDA	**1.50**	2.51	3.52	5.70	14.23
Average	Baseline	1.88	3.03	3.73	6.10	**12.31**
	LDA	1.98	2.57	3.25	5.88	14.18
	LPP	1.77	2.53	3.78	6.78	15.05
	LPDA	**1.44**	**2.23**	3.23	5.71	12.77
	CPDA	1.45	2.37	**3.13**	**5.20**	12.68

Table 3.1: Performance of DML algorithms on the Aurora-2 corpus. WERs for mixed noise training and testing for Baseline, LDA, LPP, LPDA and CPDA are given. The best performances are highlighted for each noise type per SNR level.

The advantages of the proposed DML approaches, LPDA and CPDA, over conventional techniques, LDA and LPP, are evident from the results in Table 3.1. A number of observations can be made in this regard. First, all techniques report reductions in WERs at high SNRs for most noise conditions. This is consistent with results reported for this task in [103]. Second, relatively smaller reductions in WERs are reported at low SNRs than at high SNRs for all techniques. The third observation that can be made is that the proposed DML techniques, LPDA and CPDA, perform better than the conventional techniques, LDA and LPP, in most noise conditions. The relative improvements in WERs range from 6 to 30%. The ASR performance gains reported here support the assertion that integration of discriminative aspects into manifold learning algorithms results in a robust and well-behaved transformed feature space. The fourth observation is made by comparing the fourth and the fifth rows of Table 3.1. This comparison demonstrate that CPDA provides a larger reduction in WERs than its Euclidean counterpart LPDA at all except the highest SNR for most noise types. This suggests that CPDA feature space transformation exhibits a higher level of robustness against noise when compared to the techniques based on Euclidean distance metrics such as LPDA and LDA. This is the expected advantage of CPDA that highlights the noise robustness of the cosine-correlation based distance measure as compared to Euclidean distance in DML. The noise type "Car" is a notable exception where all the feature space transformation techniques are found to be less effective.

The statistical significance of the differences in WERs for selected system pairs in Table 3.1 are reported using the Gillick and Cox matched-pairs significance test [104]. The difference in WERs obtained using LPDA features and LDA features for 20 dB subway noise is found to be statistically significant at a confidence level of 99.5%. In fact, the performance gains in WERs reported for LPDA with respect to LDA systems are found to be statistically significant for all conditions except for the subway and exhibition hall noise types at 10 dB SNR. For LPDA and CPDA performance comparisons, the difference in WERs are found to be statistically significant with a confidence level of 99.99% for all conditions except for the subway and exhibition hall 20 dB SNR cases.

Another important observation that can be made from Table 3.1 is that LPP features results in relatively high error rates compared to all other techniques for most conditions. At first, this might appear to contradict results reported by Tang and Rose in [14]. However, it should be noted that the results in [14] are reported on a task involving relatively clean training and test conditions whereas the results in Table 3.1 are reported on mixed noisy

condition training and noisy testing scenarios. For a fair comparison, the results for a clean training and clean test scenario as an average over clean subsets from Aurora-2 are presented in Table 3.2. Along with WERs, the table also presents relative WER improvements with respect to the baseline. In these results, LPP shows a better performance than LDA, which is in line with results reported in [14]. However, it should be noted that the DML algorithms, LPDA and CPDA, report even higher performance gains than LPP. The results in Table 3.2, in conjunction with those in Table 3.1, suggest that though the local geometry of the data plays an important role for clean testing, the discriminative training becomes important in the presence on noise.

3.3.3 Results for Aurora-4 Read News Corpus

The ASR WER performance of the proposed techniques on the Aurora-4 large vocabulary task is given in Table 3.3. The recognition performance of LPDA transformed features is compared with that of LDA transformed features and the baseline system configuration for the Aurora-4 task.

Table 3.3 displays ASR WER performance for seven test subsets, one clean and six noisy test scenarios. In these experiments, only recordings with the primary close-talk Sennheiser microphone are used for both training and testing, therefore there is no effect of channel. The corresponding labels are given the first column of the table. Each of these test sets consists of utterances with SNRs ranging from 5dB to 20dB. The second column of the table presents ASR WER performance for the baseline MFCC features when no feature space transformation is performed. The third column, labeled "LDA", gives the ASR WERs for the features obtained by applying the LDA projection matrix to the

Technique	Avg. WER	(Rel. Improvement)
Baseline	1.07	–
LDA	0.93	(13.08)
LPP	0.90	(15.89)
LPDA	**0.83**	**(22.42)**
CPDA	**0.82**	**(23.36)**

Table 3.2: WER for clean training and clean testing on Aurora-2 speech corpus for LDA, LPP, LPDA and CPDA. The best performances are highlighted in bold.

concatenated MFCC feature vectors as illustrated in Figure 3.1. In a similar fashion, LPDA transformations is applied to the concatenated MFCC features are the results are given the fourth column, labeled "LPDA", of the table. The last column of the table gives the relative WER reductions obtained by using the LPDA features with respect to the LDA features.

Note that similar to the Aurora-2 experiments, STC transformations are applied to the resultant features of both LDA and LPDA transformations in order to minimize the impact of the data correlation.

Noise Type	Technique		
	Baseline	LDA	LPDA (rel. LDA)
Clean	15.34	15.09	**13.97** (7.44)
Car	15.90	16.34	**14.53** (11.08)
Babble	26.62	25.37	**21.56** (15.02)
Restaurant	28.28	28.77	**24.51** (14.81)
Street	31.59	29.87	**27.46** (8.07)
Airport	23.65	23.65	**18.96** (19.83)
Train Stn.	32.08	29.96	**28.60** (4.54)
Average	24.78	24.15	**21.37** (11.51)

Table 3.3: Performance of DML algorithms for mixed conditions training on the Aurora-4 corpus. WER for Baseline, LDA, and LPDA and WER improvement relative to LDA are given. The best performances are highlighted for each noise type.

The results presented in Table 3.3 demonstrate the effectiveness of DML based LPDA over conventional LDA for a LVCSR task. LPDA features are reported to show improved WER performance over LDA across all noise types. The performance trends in Table 3.3 are similar to those in Table 3.1. The relative WER improvement of LPDA with respect to LDA ranges from 4.54 to 19.83%. The statistical significance of the differences in WERs presented in Table 3.3 are tested for the Gillick and Cox matched-pairs test [104]. It is found that the WER improvements using LPDA features with respect to LDA features are statistically significant at a confidence level of 99.98% or more for all conditions. The performance of CPDA transformations is not evaluated on this task because of it's high computational complexity requirements. However, based on the similar performance trends observed for the Aurora-2 and Aurora-4 corpora, one might expect reductions in WER on

this corpus that are similar to those observed in Table 3.1 for Aurora-2.

Results presented in Table 3.3 and the corresponding observations suggest that the relative performance gains obtained for the DML approaches generalize across task domains and speaker populations. Furthermore, as mentioned in Section 3.3.1, the optimal values of neighborhood sizes and kernel scale factors are empirically determined. From the presented results, it seems that these values also generalize reasonably well across task domains. This is important when considering the application of these techniques to new corpora. However, while these parameters are found to be robust with respect to task domains, Chapter 5 demonstrates that they might not be robust with respect to noise conditions.

3.4 Discussion and Issues

There are several aspects of the DML algorithms that lead to their demonstrated performance gains over the conventional approaches. The primary factor contributing toward these gains is the fact that these techniques not only maximize class separability as is done in all discriminant transformations, but also preserve local relationships that exist among input data vectors. These local relationships are described by the intrinsic and penalty graphs, $\mathcal{G}_{int}$ and $\mathcal{G}_{pen}$. LPDA and CPDA essentially use nonlinear mapping functions for feature extraction and, therefore, have a greater capability for exploiting geometrical relationships inherent to the feature space than is possible for linear techniques for feature extraction. There are a number of other factors contributing to the effectiveness of the proposed DML approaches. This section highlights some of the important factors and issues affecting these techniques.

3.4.1 Graph Embedding

There are two distinct advantages that graph embedding brings to the DML algorithms. The first is that it enables a mathematical representation of the distribution and geometrical structure of data. The structure of these graphs can be manipulated to achieve the desirable feature space transformations. The second is that by formulating an ASR feature analysis problem in terms of graph structures and scatters one can avoid making any assumption about the distribution of data. This is important as common dimensionality reduction approaches such as PCA and LDA work under the assumption of class conditional Gaussian

distribution of the data [47]. However, the high degree of variability in speech production results in a much more complex distribution; graph embedding avoids such assumptions.

3.4.2 Comparison to Existing Techniques

There are several general statements that can be made concerning the potential advantages of the DML algorithms relative to the LDA like approaches. First, discriminant manifold learning algorithms not only maximize class separability as is done in all discriminant transformations, but also preserve local relationships and nonlinear structure inherent to the data [1]. These local relationships are described by the intrinsic graph $\mathcal{G}_{int}$. Second, by virtue of the graph embedding scheme, there is no assumption on the data distribution, therefore it is a more general form of discriminant analysis.

In comparison to LPP, DML algorithms have two distinct advantages. First, this new framework takes advantage of the discriminant structure of the data, which LPP entirely neglects. This is done by maximizing the scatter of the penalty graph $\mathcal{G}_{pen}$. Second, LPP is an unsupervised approach that attempts to preserve the manifold based structure of the entire dataset without considering distribution of the data among various classes. This would work well when the data vectors belonging to different classes are already well separated on the surface of the manifold. This, however, may not be true in many practical scenarios. The presented DML algorithms make no such assumption and only preserve within-class manifold based geometrical structures as illustrated by the affinity weight matrices $\mathbf{\Omega}_{int}$ and $\mathbf{\Omega}_{pen}$.

3.4.3 Cosine-correlation distance measure

The improvements in noise robustness reported in Section 3.3 for CPDA relative to LPDA are supported by many previous studies. Previous studies have presented evidence which suggests that adding noise to clean speech results in reduction in the magnitude of cepstrum feature vectors but has a relatively small effect on the correlation between cepstrum vectors [97]. Cosine-correlation based distance measures have also been implemented in CDHMM based ASR decoders and have been found to achieve lower WERs on speech in noise tasks than the standard Euclidean based measure [98]. It has also been shown in other application domains that cosine-correlation based distance metrics outperform Euclidean or $l1$-distance metrics for classification tasks [20], [72]. Therefore, the CPDA results obtained in Section

3.3 are consistent with previous studies.

3.4.4 Exponentially decaying weights

The graph edge-weights and the scatter measure are two other components of the DML framework that contribute to the reported performance gains. One clear advantage of using the exponentially decaying weights is that even within the selected nearest neighbors, the nodes which are closely located in the source feature space are naturally given more importance when structuring the graphs. This further enforces the notion of locality preservation. This would not be the case if fixed binary weights were used to label whether two nodes were connected:

$$\omega_{ij} = \begin{cases} 1 & ; \quad e(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 \\ 0 & ; \quad e(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0. \end{cases} \tag{3.14}$$

The importance of the weights can further be observed by analyzing the scatter measure in Eq. (2.47) separately for the intrinsic and penalty graphs, as given in the Eq. (3.15a) and Eq. (3.15b):

$$\mathcal{F}_{int}(\boldsymbol{P}) = \min_{\boldsymbol{P}} \left\{ \sum_{i \neq j} d\{\boldsymbol{y}_i, \boldsymbol{y}_j\} \omega_{ij}^{int} \right\} \tag{3.15a}$$

$$\mathcal{F}_{pen}(\boldsymbol{P}) = \max_{\boldsymbol{P}} \left\{ \sum_{i \neq j} d\{\boldsymbol{y}_i, \boldsymbol{y}_j\} \omega_{ij}^{pen} \right\} \tag{3.15b}$$

The intrinsic weights, ω_{ij}^{int}, also factored in Eq. (3.1a) and Eq. (3.8a), penalize the intrinsic objective function if the vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are located far apart in the projected space despite actually being in the same class. The penalty weights, ω_{ij}^{pen}, also factored in Eq. (3.1b) and Eq. (3.8b), penalize the penalty graph objective function if the neighboring vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are mapped close to one another in the projection space even though they belong to separate classes. Furthermore, because of the exponentially decaying weights, the closer two vectors the higher the penalty upon such mis-projection. Thus, maximizing the ratio or difference of penalty graph scatter to intrinsic graph scatter ensures that if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are close and in the same class then $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ will be close as well, and even if two nodes from different classes are close, their projections will be far apart.

3.4.5 Sensitivity to Noise

The performance of all linear feature space transformation approaches is degraded when applied to noise corrupted speech. This issue is particularly prominent for manifold learning based approaches. This may be because the acoustic noise alters the manifold based neighborhood distributions of the feature vectors. The shapes and sizes of the local manifold structures are also affected by the choice of the scale parameter of the Gaussian kernel that is used to map the feature data onto the manifold space. The selection of this parameter has a crucial effect on the behavior of kernel and consequently on the performance of the features [14], [21]. Since the neighborhood structure is also affected by the presence of noise, there exists an increased interplay between the Gaussian kernel scale factor and ASR performance of manifold based approaches in the presence of acoustic noise. This issue is further investigated in Chapter 5.

3.4.6 Computational Complexity

It is well known that all manifold learning approaches to feature space transformation have extremely high computational complexity when compared to other discriminant feature space transformations [20], [72]. The complexity primarily arises from computing the affinity matrices $\mathbf{\Omega}_{int}$ and $\mathbf{\Omega}_{pen}$. The Aurora-2 task described in Section 2.8 involves 180 states and a training corpus of 1.4 million 117-dimensional feature vectors, and the Aurora-4 task involves about 6 million 117-dimensional feature vectors. This is a far larger task than those addressed in other application domains [8], [20], [72]. This represents a definite disadvantage of the DML algorithms when applied to the generally large speech processing tasks. A number of algorithms exist for faster but approximate nearest neighbors search that can be used for reducing the computational complexity of these algorithms. One such mechanism is LSH [26], [105]. LSH enables fast nearest neighbor search in high-dimensional spaces, thus allowing for fast computation of affinity matrices $\mathbf{\Omega}_{int}$ and $\mathbf{\Omega}_{pen}$ [25], [26]. The efficiency and effectiveness of LSH, in the context of DML algorithms, is evaluated in Chapter 4.

3.5 Conclusion

This chapter has presented a DML framework for locality preserving feature space transformations as applied to ASR. The proposed approaches attempt to preserve the within class manifold based local relationships while at the same time maximize the separability between classes. This is achieved by embedding the feature vectors into the intrinsic and penalty graphs using nonlinear kernels and by preserving or penalizing the local structure of the graphs. Two approaches have been presented which rely on two different kernels based on Euclidean and cosine-correlation distance measures. The performance of the proposed techniques has been evaluated on two different speech in noise tasks. When compared to well known approaches such as LDA and LPP, the DML algorithms have demonstrated up to 30% reduction in WER. It has also been shown that the use of the cosine-correlation based distance measure is more robust than the one based on Euclidean distances when speech is corrupted by noise. Furthermore, it is shown that these performance gains generalize across task domains and speaker populations.

Although the DML techniques provide significant gains in ASR accuracy, they suffer from two major issues, namely sensitivity to noise and high computational complexity. The next two chapters present detailed analysis of these two issues, along with empirical evaluations of techniques that can be used to address these problems.

Chapter 4

Locality Sensitive Hashing for Manifold Learning

In the previous chapter, a discriminative manifold learning framework has been presented that introduced class based discrimination to manifold learning techniques. Potential applications of these techniques include feature space transformation and dimensionality reduction in ASR. The discriminative manifold learning approaches lead to significant improvements in ASR WERs for the Aurora-2 and Aurora-4 speech in noise tasks as compared to well known feature space transformation techniques such as LDA [47], [49] and LPP [14], [19]. Despite the advantages, a major criticism against the application of manifold learning techniques to speech processing has been the very high computational complexity of these methods [18]–[20]. The computational complexity of manifold learning techniques originates from the need to construct nearest neighborhood based relationships represented by the intrinsic and penalty affinity matrices, $\mathbf{\Omega}_{int}$ and $\mathbf{\Omega}_{pen}$, discussed in Chapter 3.

If a given training set consists of T feature vectors of dimensionality D, it would take $O(DT^2)$ computations in order to construct the nearest neighbors based affinity graphs. For large amounts of speech data, where each corpus can have up to hundreds of millions of feature vectors each having thousands of dimensions, $O(DT^2)$ is a formidable computational requirement. A number of algorithms exist for faster but approximate nearest neighbors search such as kd-trees; however, many of these algorithms reach the complexity of linear search as the dimensionality of feature vectors increases [46].

[0]Parts of this chapter have been published in [25], [26].

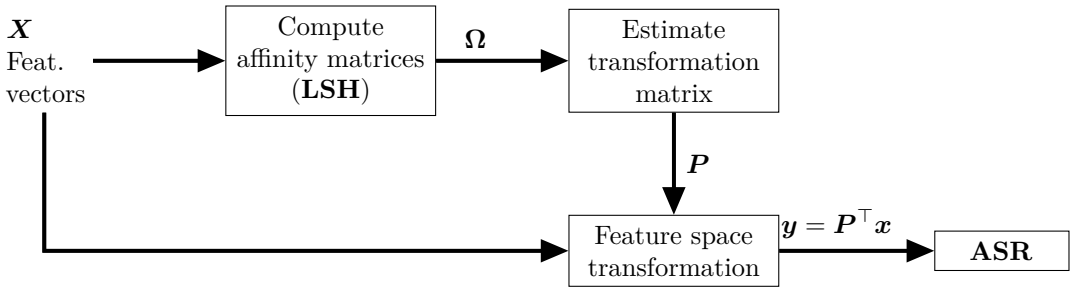

Fig. 4.1 The discriminative manifold learning framework with LSH based neighborhood computations.

This chapter investigates an approximate algorithm, known as LSH [22]–[24]. LSH is used for fast computation of the neighborhood graphs for manifold learning based feature space transformations in ASR as illustrated in Figure 4.1. LSH is particularly well suited for finding nearest neighbors in high-dimensional speech feature spaces. LSH creates hashed signatures of vectors in order to distribute them into a number of discrete buckets such that vectors close to each other are more likely to fall into the same bucket. For a given query point, the nearest neighbors search is restricted to the data points belonging to the bucket that the query point is hashed to. Therefore, LSH can drastically reduce the computational time at the cost of a small probability of failing to find the absolute closest match. In this chapter, the LSH algorithm is evaluated in the context of the DML framework introduced in Chapter 3. The LSH scheme is incorporated within the DML framework for fast computation of neighborhood graphs with the expectation of achieving high computational efficiency with minimum impact on ASR performance. LSH can be applied to other manifold learning algorithms in a similar fashion. Experimental results are presented that show that LSH can provide an order of magnitude speedup without significant impact on the ASR performance. These reductions in computational complexity should enable application of manifold based approaches to large speech datasets.

4.1 Locality Sensitive Hashing

LSH is a class of randomized algorithms that promise fast nearest neighborhood search with a high degree of accuracy [22]–[24]. A number of different implementations exist for these schemes [106]. Two of such schemes are discussed in this work, namely exact Euclidean LSH (E2LSH) and cosine distance based LSH (Cos-LSH). E2LSH attempts to find the

nearest neighbors in the Euclidean space using random projections derived from a p-stable distribution [23]. Cos-LSH on the other hand attempts to find the approximate nearest neighbors in a cosine correlation space [107], [108]. A general description of the E2LSH algorithm is presented in Section 4.1.1 and that of Cos-LSH is given in the Section 4.1.2. Though meant for nearest neighbors search in Euclidean space, in this work, E2LSH is also used for finding nearest neighbors within the cosine-correlation based CPDA. Arguments in favor of this choice are presented in Section 4.2. The details specific to the implementation and incorporation of the LSH schemes within LPDA and CPDA are given in Section 4.1.3.

4.1.1 Exact Euclidean LSH (E2LSH)

E2LSH attempts to hash a given set of feature vectors to a number of buckets on the real line in a "locality sensitive" manner. If two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are close then they should hash to the same bucket with a high probability. If two vectors are far then they should hash to different buckets with a high probability, in other words, the probability of collision of their hashed values should be very small. This is achieved by projecting each vector $\boldsymbol{x}_i$ onto real line by a family of hash functions $\mathcal{H} = \{h : \mathbb{R}^D \rightarrow \mathbb{N}\}$. The hash function used in this work is given by,

$$h(\boldsymbol{x}) = \left\lfloor \frac{< \bar{\boldsymbol{a}}, \boldsymbol{x} > + \bar{b}}{\varphi} \right\rfloor, \tag{4.1}$$

where $\bar{\boldsymbol{a}}$ is a D-dimensional random vector whose entries are chosen from a p-stable distribution, φ is the width of each segment or bucket on the real-line and acts as a quantization factor, and the bias, $\bar{b}$, is a uniform random number taken from $[0, \varphi]$. The projection of all the vectors in this manner results in a chain or table of hash buckets each having pointers to one or more vectors.

The hash function given in Eq. (4.1) is locality sensitive in Euclidean space because of the following property of p-stable distributions. If $\bar{a}_i$ for $i \in \{1 \dots D\}$ are independently and identically distributed random variables that follow a p-stable distribution, then their p^{th} -order-rooted linear combination, $(\bar{a}_1^p + \bar{a}_2^p + \dots)^{1/p}$, also follows the same distribution. For example, a Gaussian distribution is a 2-stable distribution. Thus, it can be inferred that for two arbitrary feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and a random vector $\bar{\boldsymbol{a}}$, whose elements have been independently sourced from a 2-stable distribution, the distance between their inner-products $< \bar{\boldsymbol{a}}, \boldsymbol{x}_i >$ and $< \bar{\boldsymbol{a}}, \boldsymbol{x}_j >$ is distributed as $||\boldsymbol{x}_i - \boldsymbol{x}_j||_2 \bar{\boldsymbol{z}}$, where $\bar{\boldsymbol{z}}$ is a

random variable that follows the same 2-stable distribution as the elements of $\overline{\boldsymbol{a}}$ [23], [109]. This property *guarantees* that if two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are close together in the original space then they should have a high probability of collision or hashing to the same bucket: $Pr[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)] \geq P_1$. If the two points are far apart then the collision probability should be small: $Pr[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)] \leq P_2$, where $P_1/P_2 > 1$.

For optimal performance, the difference between P_1 and P_2 should be large. To this end, k different random projections are used to create a family of composite hash functions $\mathbb{G} = \{g : \mathbb{R}^D \rightarrow \mathbb{N}^k\}$ such that $g(\boldsymbol{x}) = [h_1(\boldsymbol{x}), \ldots, h_k(\boldsymbol{x})]$, where $h_i(\boldsymbol{x}) \in \mathcal{H}$. Increasing the dimensionality k of the hash functions improves the hashing discriminative power as $(P_1/P_2)^k > P_1/P_2$. Effectively, a large k might result in a higher number of buckets each having fewer points and in turn, a smaller probability that the query and the nearest neighbors fall in the same bucket in all k projections. To reduce the impact of such unfortunate hashing, L independent hash tables are created for which hash functions, $g_1, \ldots, g_L$, are uniformly chosen from $\mathbb{G}$. This is motivated by the fact that a true nearest neighbor will be unlikely to be unfortunate in all the projections. By choosing the optimal values of φ, k and L, one can find the true neighbors with an arbitrarily high probability.

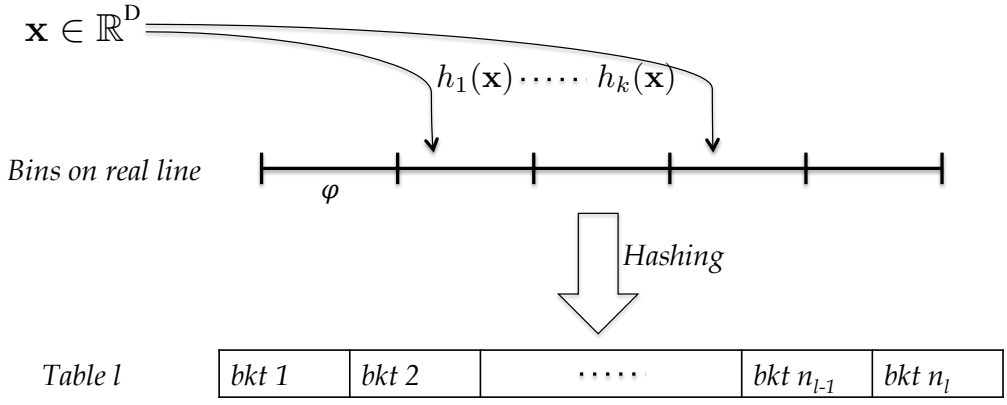

Fig. 4.2 An illustration of the two-state hashing using LSH

After hashing, each data point is represented by a k-dimensional hash signature. However, comparing these k-dimensional signatures to detect collisions may still be computationally expensive. To this end, a second level bucket hashing is implemented to store the k-dimensional signatures. The table size in the bucket hashing is chosen to be large enough to ensure that different signatures lead to different buckets with a high probability. Such a

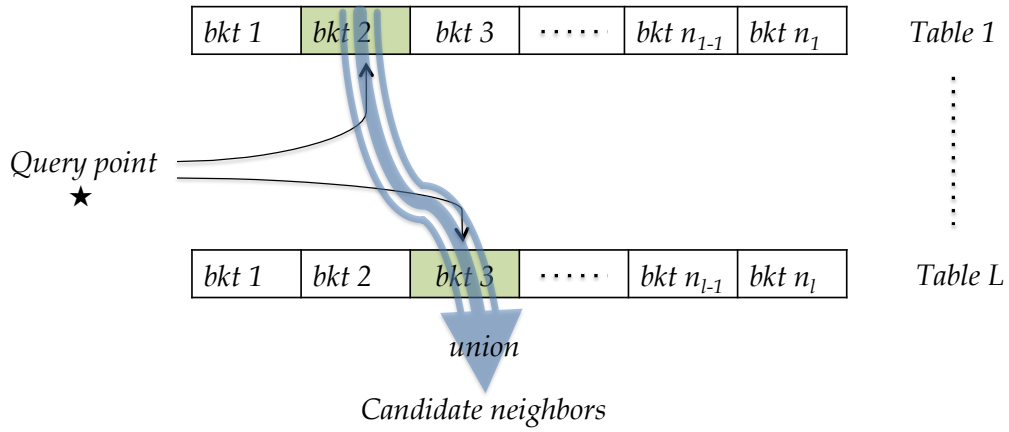

Fig. 4.3 An illustration of approximate neighbors search using LSH

secondary hashing further reduces the number of comparisons during collision detection and bucket lookup from $O(k)$ to $O(1)$. A generic schematic of LSH hashing is illustrated in Figure 4.2.

In summary, as a result of performing LSH, each feature vector, $\boldsymbol{x}_i$, is hashed into one of the buckets in each of the L tables. The number of buckets in different tables can vary because only non-empty buckets are stored. Ideally, two points that are close to each other should fall in the same buckets in all the tables. Once the hash tables are created, the search for the nearest neighbors of a given query $\boldsymbol{q}$, proceeds as follows. First, the query $\boldsymbol{q}$ is hashed to one of the buckets in each of the L tables. Then, candidate nearest neighbors to $\boldsymbol{q}$ are gathered by performing the union of these buckets from all the tables as illustrated in Figure 4.3. Finally, the nearest neighbors are searched from these candidates.

4.1.2 Cosine-correlation based LSH (Cos-LSH)

The E2LSH scheme discussed in the previous section is particularly well suited for finding nearest neighbors in Euclidean space, such as for the LPDA or LPP transformations. The CPDA algorithm on the other hand utilizes a cosine-correlation based distance measure as discussed in Section 3.2. A number of LSH schemes have been proposed in the literature that are specifically suited to a cosine-correlation space [107], [108], [110]. One such scheme is the Cos-LSH scheme [108].

Similar to E2LSH, the first step of Cos-LSH is to obtain a k-dimensional hash signature for each feature vector. However, unlike E2LSH, Cos-LSH hashes each vector, $\boldsymbol{x}_i$, using a binary representation, $\{0,1\}^k$, by projection with a unit normal random vector $\boldsymbol{r} \in \{\boldsymbol{r}_1 \dots \boldsymbol{r}_k\}$,

$$h(\boldsymbol{x}_i) = \begin{cases} 1 & ; \quad \langle \boldsymbol{r}, \boldsymbol{x}_i \rangle \geq 0 \\ 0 & ; \quad \langle \boldsymbol{r}, \boldsymbol{x}_i \rangle < 0 \end{cases} . \tag{4.2}$$

The cosine distance is then approximated by Hamming distance in the bit vector space,

$$cos(\boldsymbol{x}_i, \boldsymbol{x}_j) \approx cos(\frac{H(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j))\pi}{k}), \tag{4.3}$$

where $H(\cdot,\cdot)$ denotes Hamming distance between two bit-vectors, and k refers to the number of random projections per vector, *i.e.*, the dimensionality of the resultant bit vectors. Similar to E2LSH, multiple hash tables with different random projection vectors are created for high accuracy.

4.1.3 Implementation Details

The LSH algorithms are ideally targeted at applications with a small query set that is separate from the training set on which LSH tables are generated. In such cases, the nearest neighbors are found by iterating over the query points to identify the target hashing bucket and candidate neighbors of each. However, it is often necessary to search for nearest neighbors for all the points given in the training set. For example, for the DML algorithms described in Chapter 3, the nearest neighbors need to be calculated for all the feature vectors in the training set in order to populate the affinity matrices $\boldsymbol{\Omega}_{int}$ and $\boldsymbol{\Omega}_{pen}$. This is true for all manifold learning algorithms. In such cases, it is not feasible by both time and computational resources to re-iterate over the entire training set in order to find the nearest neighbors for each query.

This work has implemented a modified version of the LSH algorithms to avoid the aforementioned issue. For each vector multiple candidate neighborhood structures are created by calculating its pairwise distances with all vectors falling into the same bucket for each table. If the LSH structure contains L hash tables l_j, $j \in \{1 \dots L\}$, each containing several buckets B_i^j, $i \in \{1 \dots ..\}$, a nearest neighbor structure is created by calculating $T_{B_i^j}(T_{B_i^j} - 1)/2$ (where $T_{B_i^j}$ = number of points hashed in the bucket B_i^j) distances between

all the points in each bucket B_i^j. These candidate neighborhood structures are concatenated (take union) to create two separate graphs for within-class and inter-class distances with reference to given class labels. Final nearest neighbors are selected from these candidates.

4.2 The choice of LSH scheme

The CPDA algorithm is based on finding nearest neighbors on a unit hyperspace, where affinity between feature vectors is measured using an exponentially decaying cosine correlation kernel, *i.e.*, $\omega_{ij} = \exp\left(\frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - 1}{\rho}\right)$. It is trivial to show that, for two unit vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and $p = 2$, the p-stable property of E2LSH is also valid in this desired kernel space:

$$\begin{aligned} ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 &= \boldsymbol{x}_i^\top \boldsymbol{x}_i + \boldsymbol{x}_j^\top \boldsymbol{x}_j - 2\boldsymbol{x}_i^\top \boldsymbol{x}_j \\ &= 1 + 1 - 2\boldsymbol{x}_i^\top \boldsymbol{x}_j \\ &= 2(1 - \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle). \end{aligned}$$

Thus, extending the p-stability based arguments of E2LSH, it can be said that for two unit vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, their projection by a vector with elements chosen from a p-stable distribution will vary as $(1 - \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)\bar{\boldsymbol{z}}$. Therefore, the E2LSH hashing scheme is also locally sensitive in a normalized cosine-correlation space.

The Cos-LSH scheme suffers from many limitations. The scheme requires on the order of hundreds independent tables for acceptable accuracy [108], [110]. Furthermore, the approximation of Hamming distance between bit vectors to that of cosine distance between the original feature vectors only holds when high dimensional bit representations ($k \approx 1000$) are used [107], [110]. The approximation approaches equality when k goes to infinity. Though finding Hamming distance between two bit vectors is a fast and memory efficient task, the advantages diminish as the dimensionality of the bit vectors increases. The high dimensionality of target bit representations also increases the cost of the random projections. E2LSH, in comparison, is more well-behaved. The scheme is supported by the property of p-stable distributions, and provides a high degree of accuracy with an easy to understand dependence on the parameters (see Section 4.5) [23], [105]. The optimal range of dimensions of hash functions varies between 1 and 5, and the number of tables is less than 10. These claims are supported by the comparison of E2LSH and Cos-LSH in terms of their ability to find the true nearest neighbors in the next section.

E2LSH ($\varphi = 1, L = 6$)			Cos-LSH ($L = 50$)		
k	% Acc.	Speedup	k	% Acc.	Speedup
3	98.3	8.0	5000	65.9	2.1
4	86.9	9.7	1000	45.6	7.1
5	85.1	15.6	500	23.7	10.2
6	57.1	19.3	100	17.6	13.8
7	49.9	27.9	3	2.0	16.4

Table 4.1: Comparison of E2LSH and Cos-LSH schemes for their ability to find true nearest neighbors.

4.2.1 Performance Comparison of E2LSH and Cos-LSH

The arguments presented in the previous section in favor of E2LSH over Cos-LSH are supported by the performance comparisons given in Table 4.1. The table compares the two LSH schemes for their ability to provide computational performance gains and accuracy of finding true nearest neighbors by varying the number of dimensions of the projected hash signatures k. The speedup and %-accuracy are given with reference to the linear neighborhood search. For each hashing scheme, optimal value of other parameters – φ and L for E2LSH, and the number of permutations, L for Cos-LSH – are empirically chosen. Further details about the Cos-LSH scheme can be found in [107], [108], [110].

For this experiment 100,000 feature vectors are randomly selected from the Aurora-2 mixed-condition noisy training set described in Section 2.8. Two separate tables are shown, one for each LSH scheme. Note that both schemes operate in different parameter spaces, therefore a fair comparison can only be drawn with respect to the trade-off between the gained speedup and accuracy of finding true neighbors. It is evident from the comparisons in Table 4.1 that E2LSH provides much better search accuracy for the same amount of speedup. Therefore, E2LSH is used as the LSH scheme for both LPDA and CPDA transformations in this Chapter.

4.3 Experimental Results and Discussion

This section describes the experiments performed to evaluate the effectiveness of LSH for building neighborhood graphs when incorporated within discriminative manifold learning based LPDA and CPDA as applied to ASR feature space transformations. The effectiveness is measured in terms of reduction in time required to train the projection matrices, $\boldsymbol{P}_{lpda}$ and $\boldsymbol{P}_{cpda}$, and ASR WERs obtained using the transformed features. The experiments are performed on the Aurora-2 speech in noise task described in Section 2.8. All feature space transformations and HMMs are trained using the mixed-noise training set, which contains utterances corrupted by a mixture of noise conditions.

The ASR WERs are presented in Table 4.2. The table contains ASR WERs for seven different feature types for clean condition testing and noise levels ranging from 20dB to 5dB SNR. At each SNR level, the test results are averaged over a mix of utterances corrupted by three noise types, namely subway, exhibition hall and car. Each row of the table presents ASR WER results obtained using a particular feature type. The first row in the table displays the baseline ASR WER obtained when no feature space transformation is performed over the MFCC features. The second row, labeled "LDA", corresponds to application of the projection matrix obtained by LDA to the concatenated super vectors. The third row, labeled "LPP", corresponds to the ASR WER when the LPP projection matrix is applied to the concatenated feature vectors. The fourth row presents ASR WER results for features obtained by LPDA transformation. The fifth row, labeled "LPDA-LSH" refers to ASR results when LPDA transformation is obtained while using the fast E2LSH scheme for nearest neighbors calculations. Similarly, the sixth row, labeled "CPDA" refers to the features obtained as a results of using the CPDA transformation. The last row, labeled "CPDA-LSH" refers to ASR results when CPDA transformation is obtained by using the fast E2LSH scheme for nearest neighbors calculations. Note that for all but the baseline MFCC features STC transforms are estimated to minimize the impact of the data distributions resulting from the feature space transformations. For all feature space transformations, the projection matrices have dimensionality of 117×39.

A number of observations can be made when comparing ASR performances of different feature types in Table 4.2. By comparing the ASR performance of LPDA with LPDA-LSH, it can be seen that LPDA-LSH shows almost no impact on ASR performance as compared to LPDA in high SNR cases. However, ASR performance of these randomized algorithms seems

to be affected by the presence of noise. By comparing the ASR performance of LPDA-LSH with LDA it can be seen that LPDA-LSH produces improved ASR performance over LDA in most noise conditions. Similarly, it can be seen by comparing the performance of CPDA with CPDA-LSH that CPDA-LSH shows almost no impact on ASR performance as compared to CPDA without LSH for high SNR cases. However, similar to the case of LPDA-LSH, the performance is again affected by the presence of high noise. Another important observation is that the ASR performances of the fast LPDA-LSH and CPDA-LSH methods are still superior to that of LDA and LPP. Comparisons with LPP show that, similar to the results reported in Section 3.1, the DML based LPDA and CPDA algorithms gives lower ASR WERs even with LSH approximations than the unsupervised non-discriminative manifold learning based LPP technique. It can be concluded from the results that LSH can solve the inherent high computational complexity problem of the DML algorithms.

Features	SNR (dB)				
	Clean	20	15	10	5
MFCC	1.88	3.03	3.73	6.10	12.31
LDA	1.98	2.57	3.25	5.88	14.18
LPP	1.77	2.53	3.78	6.78	15.05
LPDA	1.44	2.23	3.23	5.71	12.77
LPDA-LSH	1.45	2.20	3.28	5.67	14.28
CPDA	1.57	2.37	3.13	5.20	12.69
CPDA-LSH	1.46	2.36	3.21	5.35	14.29

Table 4.2: Effect of using LSH on ASR performance. Avg. WERs for mixed noise training and noisy testing on Aurora-2 speech corpus for MFCC, LDA, LPP, LPDA and LPDA-LSH, and CPDA and CPDA-LSH are given.

4.4 Computational Complexity Analysis

LSH reduces the computational requirements for neighborhood search from $O(DT^2)$ to $O(DkTL) + O(DT_B^2)$, where D is the dimensionality of feature vectors, T is total number of feature vectors, T_B is average number of points in each bucket, k is dimension of hash

signatures $g(\boldsymbol{x})$, and L is number of hash tables. This is a significant improvement as typically T_B is several orders of magnitude smaller than T; for example, in this work, compared to $N = 1.4$ million, T_B had a value in the range of $50 - 70$ ($\varphi = 5, L = 6$ and $k = 3$) for LPDA and in the range of $250 - 350$ ($\varphi = 1$, $L = 6$ and $k = 3$) for CPDA.

The ASR results presented in the Section 4.3 are obtained using the Aurora-2 mixed conditions training set that contains 1.4 million vectors of dimensionality 117 extracted from 8440 utterances. The execution time for different feature space transformation techniques on the same computing system is: LDA – 90 seconds, LPP – 28 hours, LPDA without LSH – 26 hours, LPDA-LSH – 2.5 hours, CPDA without LSH – 36 hours, and CPDA-LSH – 4 hours, respectively. Thus, LSH provides a factor of 10 speedup to LPDA and a factor 9 speedup to CPDA. This is remarkable speedup that should enable the application of manifold learning based feature space transformation techniques to generally large speech databases.

4.5 LSH Parameterization vs Performance

The choice of the parameters of a LSH scheme can affect the computational performance as well as ASR word recognition accuracy. This section discusses some of these trade-offs in the context of the LPDA algorithm.

There are three main parameters that affect the performance of LSH, the quantization factor, φ, the number of projections or dimensions of the hash signatures, k, and the number of tables, L. The parameter φ controls the width of the buckets and hence the probability of collision for any two points. A large φ results in large buckets, thus an increase in the false collisions and computational complexity. It has been observed in other domains that a small positive value of φ suffices to achieve optimal LSH performance and larger values do not have a huge impact on accuracy [24]. In this work, $\varphi = 5$ is found to provide good LSH performance for the LPDA algorithm, however, $\varphi = 1$ is chosen for CPDA. This difference in choice of parameter φ for LPDA and CPDA is primarily due to the effect of normalizing the feature vectors in CPDA.

Increasing the dimensionality of the hash signatures, k, improves the hashing discriminative power, hence effectively decreasing the probability of collision of two points. In this work, values of k are searched in the range of 1 to 10. The dimensionality of the hash signatures represents a trade-off between the time spent in computing hash values and time

System	k	WER Clean (%-Rel.)	Training Time (Hours)	Speed-up
Baseline	–	1.88 (00.00)	–	–
LDA	–	1.98 (-05.32)	–	–
LPP	–	1.77 (05.85)	28	–
LPDA	–	1.41 (25.00)	26	1
LPDA-LSH	3	1.45 (22.87)	2.5	10.40
LPDA-LSH	4	1.66 (11.70)	0.92	28.26
LPDA-LSH	5	1.76 (06.38)	0.82	31.71

Table 4.3: LSH Parameterization vs. Performance for the Aurora-2 mixed condition training set and clean testing.

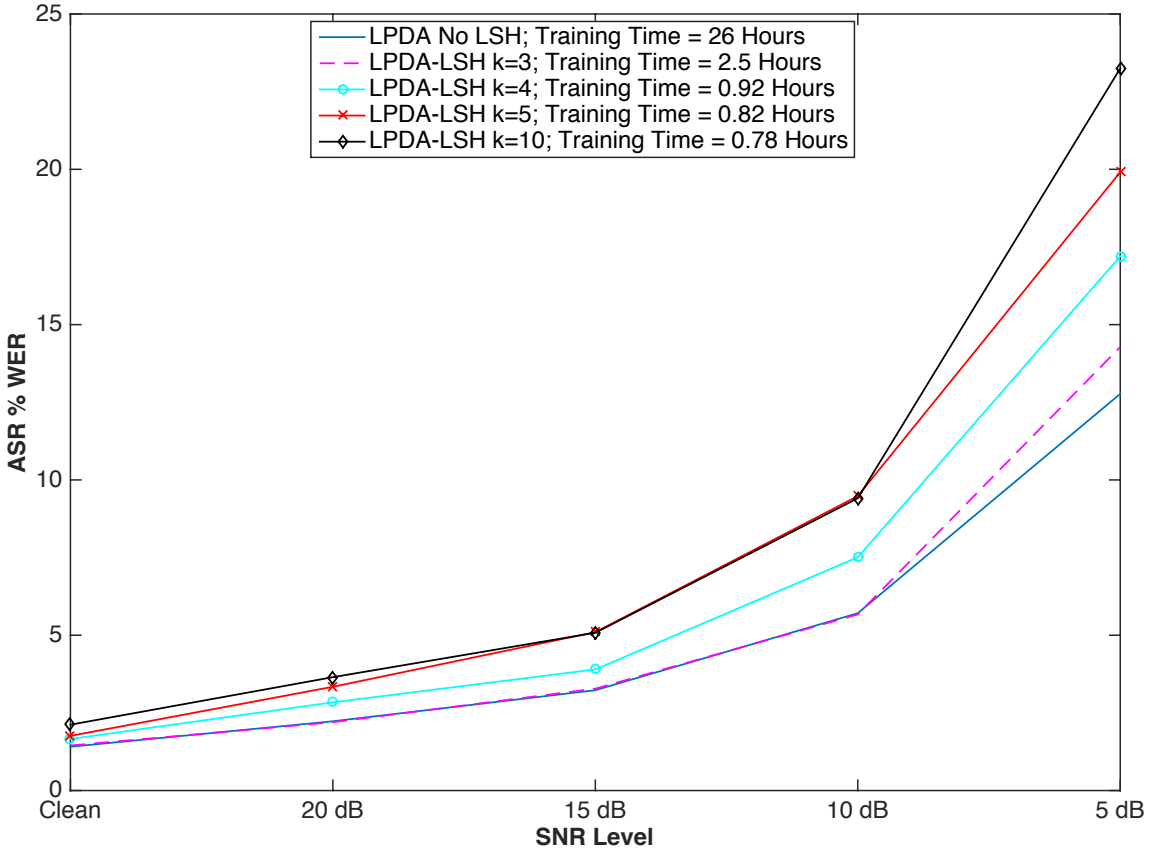

Fig. 4.4 Impact of varying the dimension, k, of hash functions on ASR performance and LPDA training time. For these experiments $\varphi = 5$ and $L = 6$ were fixed.

spent in pruning candidate neighbors to find the nearest neighbors from a bucket. This behavior is evident in the results presented in Table 4.3 and Figure 4.4. The table provides ASR WER performance of resultant LPDA features for different choices of k as evaluated on the clean test subsets of the Subway, Car, and Exhibition noise types in the Aurora-2 corpus when mixed-conditions training is used. For comparison, the table also contains ASR WER performance for the LDA and LPP features. Figure 4.4 provides comparisons for a range of SNR values and choice of the parameter k. The ASR WERs are reported for the mixed-conditions training set and are obtained as an average over utterances corrupted by the Subway, Car, and Exhibition noise types in the Aurora-2 task. See Section 2.8 for more details about the Aurora-2 task. These results show the affect of choice of k as a trade-off between the time required for training the LPDA projection matrix and ASR performance. The number of keys, k, controls the discrimination power of the hashing and the accuracy of finding true nearest neighbors; a higher value of k leads to more buckets with fewer points per bucket and lower computational complexity for searching for neighbors within each bucket. Therefore, a higher k might reduce the accuracy of finding true neighbors. It can be observed from the table that the best performance is observed for $k = 3$ with significant reduction in training time. Using value of $k < 3$ did not provide worthwhile gains in computational complexity. Another observation from the Figure 4.4 is that a low-accurate hashing (higher k) severely affects the ASR performance of the manifold learning technique. Therefore, it is important for these methods to have a close to truth representation of near neighbors.

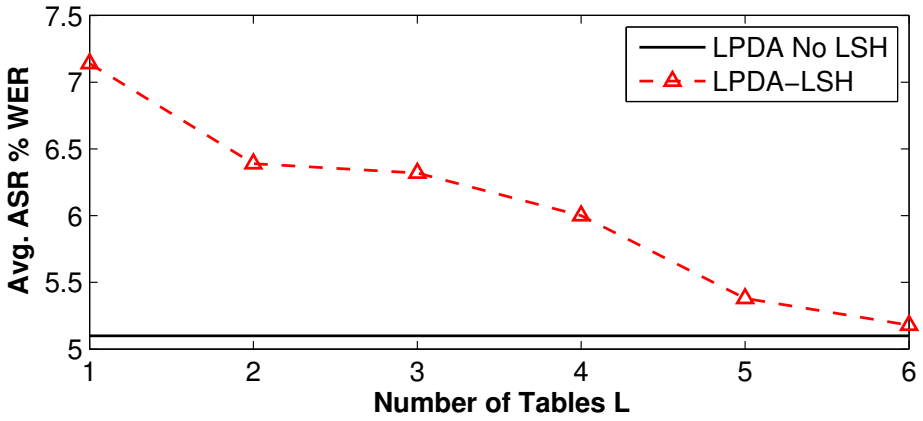

Fig. 4.5 Impact of varying the number of tables, L, of LSH on ASR performance. For these experiments $\varphi = 5$ and $k = 3$ were fixed.

Increasing L should increase the probability of finding accurate nearest neighbors, however, computational complexity also increases. This is because more tables mean more projections to perform and more buckets to scan. In this work, a suitable value of L is searched in the range of 1 to 6. Figure 4.5 presents a graph of average ASR WER versus the number of tables for the Aurora-2 mixed-conditions set described in Section 2.8 with reference to the WER from LPDA without LSH. For these experiments, the mixed conditions training set is used and the ASR WER average over all SNR levels of the Subway, Car and Exhibition noise types are provided. The training time of LPDA-LSH increases from 0.6 to 2.5 hours with the increase in L. $L = 6$ provides near-optimal ASR performance, and increasing L further does not lead to significant gains in ASR performance. The speedup decreases with increases in L.

4.6 Conclusion

This chapter has investigated the application of a fast approximate nearest neighbor search algorithm, known as LSH, in conjunction to the DML algorithms. LSH is used for populating the affinity matrices in manifold learning based feature space transformations. The discriminative manifold learning based LPDA and CPDA algorithms are used as examples of manifold learning techniques. ASR WER and execution times are reported for the algorithms with and without using LSH. Performance comparisons are also made between these approaches and the more well-known LDA and LPP. It is demonstrated that LSH provides a speedup for 10 for LPDA and 9 for CPDA with no to minimal impact on their ASR WER performance.

LPDA uses a Euclidean distance based measure to characterize the manifold domain relationships among feature vectors, whereas CPDA uses a cosine-correlation based distance measure. For the CPDA algorithm, a cosine adaptation of E2LSH scheme is chosen for hashing. The E2LSH scheme is compared to another commonly used cosine distance based LSH scheme, Cos-LSH, in terms of the trade-off between computational gains and nearest neighbor search accuracy. The results have shown that the E2LSH scheme provides better search accuracy for the same amount of speed up.

Finally, the results presented in this chapter can also be used to analyze the sensitivity of manifold learning based methods to thefinding true near neighbors. It is shown that a less accurate near neighbors estimation can lead to significant degradation in ASR performance.

Chapter 5

Noise aware Manifold Learning

The discriminative manifold learning approaches presented in Chapter 3 has shown significant improvements in ASR WERs as compared to well known feature space transformation techniques such as LDA [47], [49] and LPP [14], [19]. However, it is evident from the results presented in Tables 3.1 and 3.3 that the performance of all linear feature space transformation approaches is degraded when applied to noise corrupted speech. This chapter addresses this issue for manifold learning based feature space transformation approaches in particular.

Manifold learning based approaches work under the assumption that high dimensional data can be considered as a set of geometrically related points residing on the surface of a lower dimensional manifold. The aim is then to preserve non-linear relationships along this manifold in the transformed feature space. However, additive noise in the linear spectrum domain alters the norm of cepstrum domain feature vectors [97]. The unseen test features may not obey the structure of the manifold learned from the training data resulting in performance degradation for noise corrupted speech. Therefore, the interaction between acoustic noise conditions and the structure of local neighborhoods makes manifold learning based methods highly sensitive to noise. To support this argument, empirical evidence demonstrating the impact of noise on manifold learning techniques and the importance of environmental compensation for such techniques is presented in Section 5.1.

As discussed in Chapter 3, a Gaussian kernel is used to map the speech feature vectors onto the manifold space. The shape and size of the local manifold structures are affected by

[0]Parts of this chapter have been published in [15], [27].

the choice of the kernel scale parameter. The selection of this parameter has a crucial effect on the behavior of kernel, and consequently the performance of the features [21]. Since the neighborhood structure is also affected by the presence of noise, there exists a significant interplay between the Gaussian kernel scale factor and acoustic noise. The analysis of the effect of noise level on the optimal choice of Gaussian kernel scale factor for manifold learning techniques is presented in Section 5.2.

Section 5.3 proposes a heuristic NaML method to address the interaction between acoustic noise conditions and the structure of local neighborhoods. NaML is described as an approach for exploiting estimated background characteristics to define the size of the local neighborhoods used for LPDA feature space transformations. It is shown that NaML significantly reduces the speech recognition WER in a noisy speech recognition task over LPDA, particularly at low signal-to-noise ratios. In this chapter, the discriminative manifold learning LPDA is chosen as an example manifold learning technique primarily because of the good performance obtained using LPDA as presented in Chapter 3 and [16]. However, the NaML framework can be extended to any manifold learning algorithm.

5.1 Characterizing the impact of noise on manifold learning

This section analyzes the impact of mismatched environmental conditions on the discriminative manifold learning based LPDA approach for estimating ASR feature space transformations. It is argued in Chapter 3 that manifold learning techniques benefit from the assumption that there is a structural relationship amongst data vectors which can be maintained by preserving the local relationships among the transformed data vectors. This suggests that if the presence of noise distorts these local relationships, the effectiveness of these techniques will be affected. In this section, this phenomenon is examined under highly mismatched acoustic conditions by estimating LPDA transforms and training CDHMM models under clean conditions and evaluating ASR WER under a number of noisy conditions.

Two set of experiments are carried out to test the hypothesis that environmental mismatch may affect manifold learning techniques to a greater extent than other feature types such as unaltered MFCC features. First, the effect of noise on ASR WER is evaluated when both MFCC features and LPDA features are applied to the Aurora-2 speech in noise task described in Section 2.8. It is observed that the WER is far higher for the LPDA transformed features than that observed when no feature space transformation is performed.

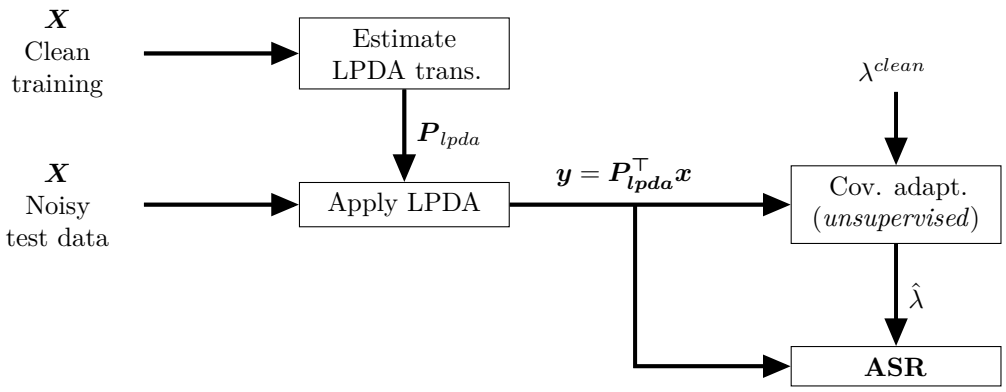

Fig. 5.1 Architecture of the system used to analyze the impact of noise on manifold learning based algorithms.

Second, the effects of noise are measured again after adapting the HMM covariance matrices to reduce the acoustic model mismatch with respect to the noisy test data. This procedure is illustrated in Figure 5.1. Interestingly, it is found that, after transforming the model parameters, the gain in ASR performance observed for the LPDA transformed features is significantly higher than that for the MFCC features.

The experimental results supporting these observations are displayed in Table 5.1. The table contains ASR WERs for test conditions corresponding to a range of SNRs in the subway noise condition. The two major rows in Table 5.1 represent the two different types of features being evaluated, namely untransformed MFCC features and the LPDA transformed features. For each feature set, the percent WER is displayed with respect to SNR level when no acoustic adaptation is performed, referred to in Table 5.1 as "None", and when unsupervised regression based covariance adaptation is performed on the HMM model during recognition, referred to in the table as "Cov.".

It is clear from observing the "None" labeled rows of Table 5.1 that there is a significant increase in WER for both the MFCC and LPDA transformed features as the testing conditions become increasingly mismatched with respect to the clean training conditions. However, it is also clear that the increase in WER is far greater for the case of the LPDA features. This suggests that imposing the structural constraints associated with manifold learning is actually increasing the confusability of the data when corrupted by additive noise.

It is well known that the presence of noise introduces distortions in the covariance structure of the data [111]. These distortions result in changes in the probability densities

of noisy speech features resulting in a mismatch with respect to model distributions trained under clean conditions. For the particular case of manifold learning, it is also true that the unseen test features may not obey the structure of the manifold learned from clean training features during LPDA estimation. This results in a higher degradation in ASR performance when manifold learning based feature space transformation techniques are used.

The observations concerning the impact of noise on the covariance of the data and the impact of the mismatched covariance on the assumed underlying manifold structure of the data suggests that some form of environmental compensation should reduce the effects of noise on the LPDA transformed features. The rows of Table 5.1 labeled as "Cov." display the WERs obtained when applying unsupervised regression based covariance adaptation to transforming Gaussian mixture covariance matrices in the CDHMM model. A multiple pass adaptation scenario is used where maximum likelihood linear regression (MLLR) based covariance transforms are estimated from all test utterances corresponding to a given noise level [112]. While this scenario is not consistent with the one used to obtain the "no adaptation" results shown in Table 5.1, it serves to demonstrate the added impact mismatched conditions have on the LPDA manifold learning approach.

Features	Adapt.	20 dB	15 dB	10 dB	5 dB
MFCC	None	2.52	6.97	24.01	54.19
	Cov.	2.67	5.59	16.49	44.83
LPDA	None	7.80	18.94	40.71	61.20
	Cov.	1.96	4.30	13.75	39.89

Table 5.1: ASR %-WER for clean training and subway noise testing on the Aurora-2 speech corpus for MFCC and LPDA features with and without environmental compensation.

It is clear from the "Cov." results in the table that WERs for both LPDA transformed and untransformed features are significantly reduced at almost all SNR levels. Part of this reduction in both cases is due to the fact that all utterances from a given SNR level are used to estimate the regression based adaptation matrix. This is an expected and well known phenomenon. However, the WER reductions for the case of LPDA transformed features are remarkable. In fact, while LPDA WERs are considerably higher than the WERs for the untransformed features with uncompensated models, the LPDA WERs are considerably lower than those obtained for the untransformed features when covariance normalization is

applied.

It can be concluded from the results presented in Table 5.1 that the direct impact of noise on the DML techniques described in Chapter 3 occurs through distortions in the local neighborhoods for the manifold learning algorithm. These local neighborhoods are defined by the affinity matrices and the associated Gaussian kernels. The NaML approach presented in Section 5.3 directly deals with this issues by considering the relationship between the size of the Gaussian kernels and the SNR levels.

5.2 Role of the Gaussian kernel scale factor

An important issue with the manifold learning approaches is the choice of the shape and size of the neighborhoods. Typically, a nonlinear kernel is used to map the feature data onto the manifold space (see Section 2.6.2 and Chapter 3). The geometrical compactness of this mapping is determined by the kernel scale parameter, ρ, as described in Eq. (3.1) and Eq. (3.8). The selection of this parameter has a crucial effect on the behavior of the kernel and consequently on the performance of the features [21], [113]. Most often this parameter is heuristically tuned to the given dataset. Using a value of the scale parameter which is too large would tend to flatten the Gaussian kernel leading to a graph where all data pairs are considered equally important. On the other hand, using a value which is too small would result in a graph which lacks sufficient smoothing of the manifold, thus resulting in a kernel which is overly sensitive to noise. These observation are supported by experimental results presented in this section. For these experiments a multi-noise mixed training dataset is used to minimize the environmental mismatch between the training and testing conditions.

Table 5.2 demonstrates the dependence of LPDA approach on ρ values. ASR performance results using multi-noise mixed CDHMM training on the LPDA transformed features corresponding to two different values of ρ, $\rho_1 = 800, \rho_2 = 1000$, are given for three different noise types (Sub.=subway, Exh.=exhibition hall, and car). The next five columns of the table display the ASR %-WER performance for five different SNR levels (clean, 20 dB, 15 dB, 10 dB, and 5 dB). In all these cases, a STC transform is estimated to minimize the impact of the data correlations resulting from the LPDA projections.

It can be observed from the results in Table 5.2 that $\rho = 800$ gives better performance in case of clean speech and high SNR compared to the case when $\rho = 1000$ is used. However, using a kernel with $\rho = 1000$ produces better performance for low SNR conditions. This

Noise	ρ	Clean	20 dB	15 dB	10 dB	5 dB
Sub.	800	**1.69**	**2.27**	3.65	6.02	13.11
	1000	1.83	2.43	**3.29**	**5.25**	**11.82**
Exh.	800	**1.08**	**2.56**	**3.61**	6.79	16.17
	1000	1.38	2.56	3.72	**6.08**	**14.04**
Car	800	**1.73**	2.74	3.40	6.83	15.99
	1000	2.19	**2.27**	**3.02**	**5.04**	**15.33**

Table 5.2: Comparison of LPDA ASR performance in terms of %-WER for two different values of ρ, *viz.,* $\rho_1 = 800, \rho_2 = 1000$. The best of the two cases have been highlighted in bold.

trend is visible for all noise types. These results establishes that a fatter kernel gives better performance with noise, while keeping mind that a too high ρ will result in loss of non-linear manifold based relationships.

To conclude, an important finding that can be derived form these results is that the optimal value of the scale parameter is heavily influenced by the level of noise corruption in speech. Such dependence of the optimal choice of kernel scale factor on SNR level can be handled by multiple scale factors, each specific to a noise condition. An automatic scheme to achieve this is discussed next.

5.3 Noise aware Manifold Learning (NaML)

As a demonstration of how the relationship between ρ and environmental noise can be exploited, an approach for noise-aware manifold learning (NaML) is investigated. As an example of NaML, noise-aware LPDA (N-LPDA) transformations in ASR is implemented. The approach is carried out in three steps as shown in Figure 5.2. First, an ensemble of projection matrices are trained based on a range of values of ρ. Second, a heuristic technique is used to identify a value of ρ – and hence a specific LPDA projection – that maximizes ASR performance for a given SNR level. Note that an intermediate step of estimating the SNR for each speech utterance is involved here. Recent research has produced a number of algorithms for blind estimation of SNR[1] for noisy speech data [114], [115]. However, in the

[1]It is called blind estimation if corresponding clean speech data is not available.

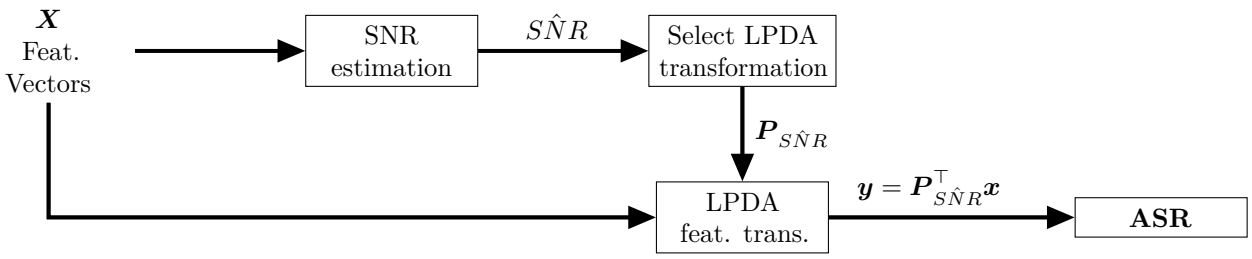

Fig. 5.2 A schematic for the N-LPDA system

absence of the corresponding clean data, it is often hard to accurately estimate the SNR of a given speech waveform. In fact, different techniques can provide very different SNR estimates for the same noisy utterance. In this thesis, a hybrid SNR estimation algorithm[2] based on [114] and [115] is used to automatically estimate the SNRs for the noise corrupted utterances in the Aurora-2 corpus. The hybrid approach has been able to correctly estimate the SNR levels for 85% of the utterances in Aurora-2 test set[3]. Finally, separate CDHMM models are trained from the features obtained by using this ensemble of projection matrices. During recognition, SNR is estimated for each test utterance, feature space transformation is performed using the projection matrix associated with the corresponding SNR level, and finally the corresponding CDHMM model is used for recognition.

In this work, an ensemble of LPDA transformations are trained using affinity and penalty matrices relying on five different sets of kernel scale parameters: $\{800, 800|900, 900, 1000, 1000|3000\}$. These values are empirically chosen based on ASR performance obtained on a development set across a range of SNRs. Here, the values in the format '$a|b$' refer to the two different scaling parameters used for the intrinsic and penalty graph kernels, respectively. The results of this approach are shown in Table 5.3 for the various noise conditions described earlier. For each noise type, ASR %-WERs are compared for LPDA and N-LPDA. Note that in the case of LPDA the configuration corresponding to $\rho = 1000|3000$ as discussed in Table 3.1 is used. The last column in the table lists ASR WER averaged over all listed SNR levels for each noise condition. An additional fourth table, labeled "Avg.", is shown with ASR WER for LPDA and N-LPDA averaged across

[2]Final SNR = 0.6*NIST [115] + 0.4*WADA [114]

[3]For the Aurora-2 test set, the actual SNR levels of the speech data is known. However, in a real-world scenario, SNR of a noisy speech utterance would not be known. Therefore, in this work, a blind estimate of SNR is used to choose the appropriate projection matrix.

Noise	LPDA (ρ)	SNR (dB)				
		Clean	20	15	10	5
Sub.	800	1.69	2.27	3.65	6.02	13.11
	1000	1.83	2.43	3.29	**5.25**	11.82
	1000|3000	1.57	**2.18**	3.29	5.28	11.73
	N-LPDA	**1.44**	**2.18**	**3.25**	**5.25**	**11.44**
Exh.	800	**1.08**	2.56	3.61	6.79	16.17
	1000	1.38	2.56	3.72	**6.08**	14.04
	1000|3000	1.23	**2.22**	3.64	6.66	**13.85**
	N-LPDA	1.14	2.28	**3.36**	**6.08**	**13.85**
Car	800	1.73	2.74	3.40	6.83	15.99
	1000	2.19	**2.27**	3.02	**5.04**	15.33
	1000|3000	**1.52**	2.30	**2.77**	5.19	12.73
	N-LPDA	1.67	2.36	2.92	**5.04**	**12.60**

Table 5.3: ASR %-WERs for mixed noise training and testing on the Aurora-2 speech corpus for N-LPDA for different heat-kernel values.

the different noise types.

It is apparent from the results in Table 5.3 that N-LPDA produces slightly better average ASR performance for most conditions as compared to any single ρ choice. This result confirms the dependence of the kernel scale factor on environmental conditions. It also suggests that choosing an optimal value of ρ by selecting from an ensemble of alternative transforms may be a plausible approach for reducing the impact of this dependence. Though, the suggested open-loop scheme might not be the most parsimonious approach in terms of computational resources, the homogeneous performance gain in terms of ASR WER across all noise conditions justifies the increase in computational complexity. Considering the difficulties with an open-loop approach, it might be worthwhile to investigate the use of a noise removal technique with manifold learning based approaches.

Noise	Feat.	SNR (dB)				
		Clean	20	15	10	5
Sub.	MFCC	1.76	2.99	4.0	6.21	11.89
	LDA	1.82	2.25	**2.93**	5.29	12.32
	LPP	1.66	2.33	3.50	5.71	13.26
	N-LPDA	**1.44**	**2.18**	3.25	**5.25**	**11.44**
Exh.	MFCC	1.89	3.34	3.83	6.64	**12.72**
	LDA	1.83	2.63	3.37	6.67	14.29
	LPP	1.76	2.56	4.23	8.55	16.91
	N-LPDA	**1.14**	**2.38**	**3.36**	**6.08**	13.85
Car	MFCC	1.99	2.77	3.36	5.45	**12.31**
	LDA	2.29	2.83	3.45	5.69	15.92
	LPP	1.88	2.71	3.61	6.08	14.97
	N-LPDA	**1.67**	**2.36**	**2.92**	**5.04**	12.60

Table 5.4: ASR %-WER for mixed noise training and testing on Aurora-2 speech corpus for LDA, LPP and N-LPDA. The best performance has been highlighted for each noise condition.

5.4 Conclusion

This chapter has investigated the effect of acoustic noise conditions on manifold learning approaches for feature space transformations in CDHMM based ASR. It is found that the structural constraints associated with manifold learning approaches result in transformed features that are more sensitive to mismatch in acoustic conditions than untransformed MFCC features. It has also been shown that environment dependent performance degradation can be traced to the choice of the size of the Gaussian kernel scale factor used for defining local affinity matrices in manifold learning. These observations led to a multi-model approach for improving the noise robustness of manifold learning based feature space transformations, referred to here as NaML. The NaML approach is shown to provide reduced WER across a range of acoustic conditions with respect to the LPDA transformation implemented without requiring explicit knowledge of background acoustic conditions.

The work in this chapter has only focused on the effect of additive background noise on the speech manifold. It has, however, not analyzed the impact of channel distortion on the manifold. It would be interesting to extend the studies presented in this chapter to the case of channel distortion. Furthermore, it would also be interesting to see the effect of channel distortion on manifold when some channel distortion removal technique, such as cepstrum mean normalization (CMN), is applied.

Chapter 6

Manifold Regularized Deep Neural Networks

This chapter presents an approach for applying manifold based constraints to the problem of regularizing training algorithms for DNN based acoustic models in ASR. The goal of this research is to develop methods that can enhance the ability of DNNs to provide efficient and distributed model representations by using manifold learning to impose local structural constraints. To this end, the optimization criterion in DNN training is redefined to incorporate the DML based constraints as discussed in Chapter 3 for emphasizing local neighborhood relationships among acoustic feature vectors. This approach involves redefining. The resultant training mechanism is referred to here as MRDNN training and is described in Section 6.2. ASR performance of the proposed technique is evaluated on the Aurora-2 and Aurora-4 speech-in-noise corpora discussed in Section 2.8.

This work is primarily motivated by two set of studies. The first is studies have discussed the impact of the local structure of the feature space on EBP optimization, including the features provided to the input of the DNN as well as features produced at the output of hidden layers of the DNN. It has been suggested that input features with strong local structure lead to improved learning for DNNs [57]. The second is the studies suggesting that deep auto-encoder networks perform well because they are able to implicitly learn a low-dimensional manifold based representation of the training data [58], [65]. Considering that manifold based constraints emphasize the underlying local structure of speech features,

[0]Parts of this chapter have been published in [28], [29].

manifold learning in DNN framework has the potential to address several important learning issues.

Previous work on optimizing deep network training includes approaches for pre-training of network parameters such as restricted Boltzmann machine (RBM) based generative pre-training [54], [60], [61], layer-by-layer discriminative pre-training [64], and stacked auto-encoder based pre-training [65], [66]. Other techniques, like dropout with the use of ReLUs as nonlinear elements in place of sigmoid units in the hidden layers of the network are also thought to have the effect of regularization on DNN training [95], [116].

Similar to the shallow NNs discussed in Section 2.7.1, DNNs have been applied to ASR tasks in both *tandem* and *hybrid* configurations [96]. In tandem configurations, DNNs are usually trained with a low dimensional *bottleneck* hidden layer [30], and the activations obtained from the bottleneck layer are used as input features to a CDHMM based ASR system [31], [32]. In hybrid configuration, DNNs are used for discriminative estimation of HMM state posterior probabilities. This is referred to as a hybrid DNN/HMM configuration [33], [55], [56], [117]. These state posteriors are converted into HMM state observation probabilities by normalizing the posteriors using prior state probabilities [55]. The manifold regularization techniques described in this chapter are applied to estimate parameters for a tandem DNN with a low-dimensional bottleneck hidden layer. The discriminative features obtained from the bottleneck layer are input to a GMM-HMM speech recognizer [31], [96]. One such implementation is illustrated in Fig. 6.1, where the concatenated feature vectors are fed into a four layers deep network, and the outputs are used as features for a GMM-HMM ASR system. Because of the diagonal covariance requirements of HMM-GMM ASR system, it is also helpful to orthogonalize the bottleneck features using PCA.

An important impact of the proposed technique is the well-behaved internal feature representations associated with MRDNN trained networks. It is observed that the modified objective criterion results in a feature space in the internal network layers such that the local neighborhood relationships between feature vectors are preserved. That is, if two input vectors, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, lie within the same local neighborhood in the input space, then their corresponding mappings, $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, in the internal layers will also be neighbors. This property can be characterized by means of an objective measure referred to as the contraction ratio [70], which describes the relationship between the sizes of the local neighborhoods in the input and the mapped feature spaces. The performance of MRDNN training in

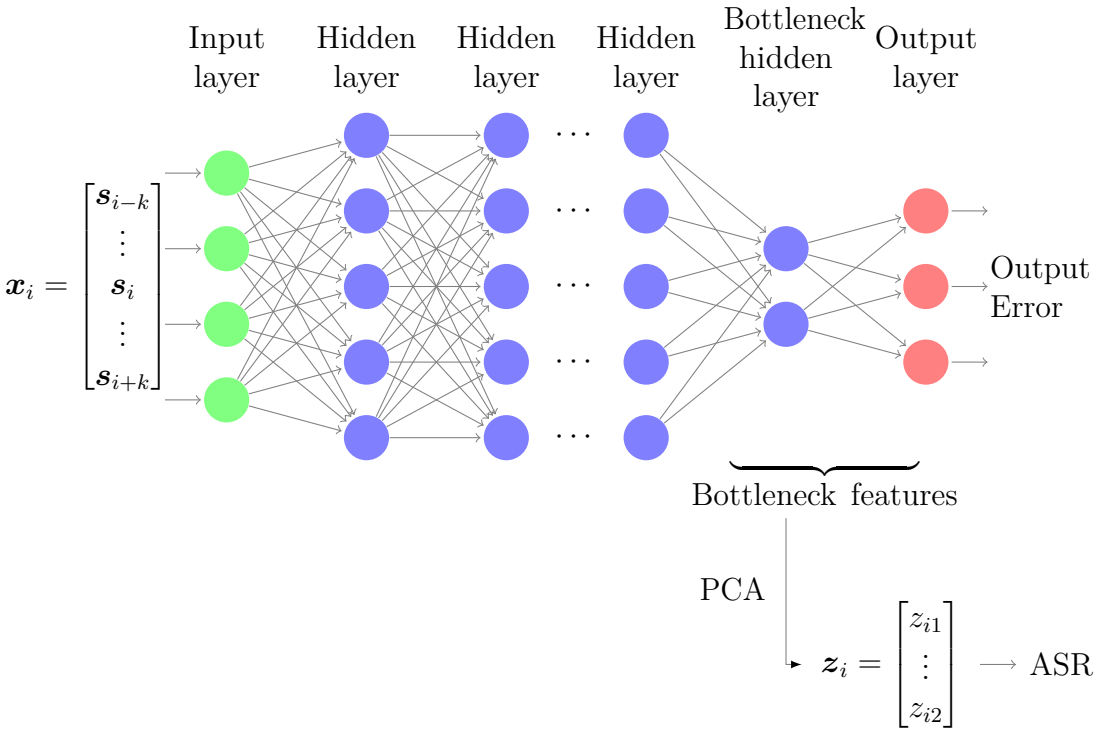

Fig. 6.1 An example of a bottleneck DNN based feature extractor in tandem configuration. The network takes multiple concatenated frames of speech features, $\boldsymbol{s}_i$, at the input and provides phonetic probabilities at the output. After training, the outputs from the bottleneck layer, $\boldsymbol{z}_i$, are used as features in a GMM-HMM speech recognizer.

producing mappings that preserve these local neighborhood relationships is presented in terms of the measured contraction ratio in Section 6.2.1. In addition, it is discussed in Section 6.2.2 that the inclusion of manifold based locality preservation constraints in the objective criterion leads to a robust gradient estimation during EBP training, which helps in reducing the impact of rapid changes on the classification function.

6.1 Deep Neural Networks

A DNN is simply a feed-forward artificial neural network that has multiple hidden layers between the input and output layers. The basic concepts of a deep network are sames as those discussed for the shallow NNs in Section 2.7. It has been long believed in the machine learning community that deep architectures are well suited for complex tasks that can be coded into a layered architecture involving several sub-tasks. The ASR process is a natural fit to this model.

Typically, a deep network produces a mapping $f_{dnn} : \boldsymbol{x} \rightarrow \boldsymbol{z}$ from the inputs, $\boldsymbol{x}_i \in \boldsymbol{X}_1^T$, to the output activations, $\boldsymbol{z}_i, i = 1 \ldots T$. This is achieved by finding an optimal set of weights, $\boldsymbol{W}$, to minimize a global error or loss measure, $V(\boldsymbol{x}_i, \boldsymbol{c}_i, f_{dnn})$, defined between the outputs of the network and the targets, $\boldsymbol{c}_i, i = 1 \ldots T$. In this work, $L2$-norm regularized DNNs are used as baseline DNN models. The objective criterion for such a model is defined as

$$\mathcal{F}(\boldsymbol{W}) = \frac{1}{T} \sum_{i=1}^{T} V(\boldsymbol{x}_i, \boldsymbol{c}_i, f_{dnn}) + \gamma_1 ||\boldsymbol{W}||^2, \tag{6.1}$$

where the second term refers to the $L2$-norm regularization applied to the weights of the network, and γ_1 is the regularization coefficient that controls the relative contributions of the two terms.

The weights of the network are updated for several epochs over the training set using mini-batch gradient descent based EBP,

$$w_{m,n}^l \longleftarrow w_{m,n}^l + \eta \nabla_{w_{m,n}^l} \mathcal{F}(\boldsymbol{W}), \tag{6.2}$$

where $w_{m,n}^l$ refers to the weight on the input to the n^{th} unit in the l^{th} layer of the network from the m^{th} unit in the preceding layer. The parameter η corresponds to the gradient learning rate. The gradient of the global error with respect to $w_{m,n}^l$ is given as

$$\nabla_{w_{m,n}^l} \mathcal{F}(\boldsymbol{W}) = -\Delta_{m,n}^l - 2\gamma_1 w_{m,n}^l, \tag{6.3}$$

where $\Delta^{l}_{m,n}$ is the error signal in the l^{th} layer and its form depends on both the error function and the activation function. Typically, a soft-max nonlinearity is used at the output layer along with the cross-entropy objective criterion. Assuming that the input to p^{th} unit in the final layer is calculated as $net^{L}_{p} = \sum_{n} w^{L}_{n,p} z^{L-1}_{p}$, the error signal for this case is given as $\Delta^{L}_{n,p} = (z_p - t_p) z^{L-1}_{n}$ [118], [119].

6.2 Manifold Regularized Deep Neural Networks

Manifold regularization is a data dependent regularization framework. An introduction to manifold regularization with the examples of LapRLS and LapSVM is provided in Section 2.6.2.2. There has been several recent efforts to apply some form of manifold based learning to a variety of machine learning tasks. Authors in [10] have applied the techniques presented in manifold regularization framework to feature space transformations for a phone recognition task.

The graph manifold regularization framework has also been applied to neural networks to some extent. In [120], authors have investigated manifold regularization in single hidden layer multilayer perceptrons for a phone classification task. Various aspects of manifold learning based semi-supervised embedding have been applied to deep learning for a hand-written character recognition task in [121]. Manifold regularized shallow neural networks have been used for image classification for remote sensing in [122]. In other somewhat related work, deep recurrent neural networks based methods have been explored to preserve temporal correlation structure in the speech sequence data [123].

While the manifold based regularization approach presented in this chapter is related to the efforts mentioned earlier, the methods presented here rely on the DML framework. This choice is motivated by the results discussed in Chapter 3 that demonstrate the effectiveness of DML for maintaining separability between classes of speech feature vectors while preserving the local manifold based relationships when applied to feature space transformations in ASR. The work presented in this chapter is also the first to apply this class of techniques to training deep neural networks for ASR.

The algorithm formulation for manifold regularized training of deep networks is provided in Section 6.2.1. The architecture and training procedure of MRDNNs is discussed in Section 6.2.2. The proposed training procedure emphasizes local relationships among speech feature vectors along a low dimensional manifold while optimizing network parameters. To support

this claim, empirical evidence is provided in Section 6.2.3 that a network trained with manifold constraints has a higher capability of preserving the local relationships between the feature vectors than one trained without these constraints.

6.2.1 Algorithm Formulation

This work incorporates locality and geometrical relationships preserving manifold constraints as a regularization term in the objective criterion of a deep network. These constraints are derived from a graph characterizing the underlying manifold of speech feature vectors in the input space. The objective criterion for a MRDNN network producing a mapping $f_{mrdnn} : \boldsymbol{x} \rightarrow \boldsymbol{z}$ is given as follows,

$$\mathcal{F}(\boldsymbol{W}; \boldsymbol{Z}) = \sum_{i=1}^{T} \left\{ \frac{1}{T} V(\boldsymbol{x}_i, \boldsymbol{c}_i, f_{mrdnn}) + \gamma_1 ||\boldsymbol{W}||^2 + \gamma_2 \frac{1}{k^2} \sum_{j=1}^{k} ||\boldsymbol{z}_i - \boldsymbol{z}_j||^2 \omega_{ij}^{int} \right\}, \tag{6.4}$$

where $V(\boldsymbol{x}_i, \boldsymbol{c}_i, f_{mrdnn})$ is the loss between a target vector $\boldsymbol{c}_i$ and output vector $\boldsymbol{z}_i$ given an input vector $\boldsymbol{x}_i$; V$(\cdot)$ is taken to be the cross-entropy loss in this work. $\boldsymbol{W}$ is the matrix representing the weights of the network. The second term in Eq. (6.4) is the $L2$-norm regularization penalty on the network weights; this helps in maintaining smoothness for the assumed continuity of the source space. The effect of this regularizer is controlled by the multiplier γ_1. The third term in Eq. (6.4) represents manifold learning based locality preservation constraints as defined in Chapter 3. Note that only the constraints modeled by the intrinsic graph, $\mathcal{G}_{int} = \{\boldsymbol{X}, \boldsymbol{\Omega}_{int}\}$, are included, and the constraints modeled by the penalty graph, $\mathcal{G}_{pen} = \{\boldsymbol{X}, \boldsymbol{\Omega}_{pen}\}$, are ignored. The reasoning behind this choice is discussed in Section 6.4.3. The scalar k denotes the number of nearest neighbors connected to each feature vector, and ω_{ij}^{int} refers to the weights on the edges of the intrinsic graph as defined in Eq. (3.1a). The relative importance of the mapping along the manifold is controlled by the regularization coefficient γ_2.

This framework assumes that the data distribution is supported on a low dimensional manifold and corresponds to a data-dependent regularization that exploits the underlying manifold domain geometry of the input distribution. By imposing manifold based locality preserving constraints on the network outputs in Eq. (6.4), this procedure encourages a mapping where relationships along the manifold are preserved and different speech classes

are well separated. The manifold regularization term penalizes the objective criterion for vectors that belong to the same neighborhood in the input space but have become separated in the output space after projection.

The objective criterion given in Eq. (6.4) has a very similar form to that of a standard DNN given in Eq. (6.1). The weights of the network are optimized using EBP and gradient descent,

$$\nabla_{\boldsymbol{W}} \mathcal{F}(\boldsymbol{W}; \boldsymbol{Z}) = \sum_{i}^{T} \frac{\partial \mathcal{F}(\boldsymbol{W}; \boldsymbol{Z})}{\partial \boldsymbol{z}_i} \frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{W}}, \tag{6.5}$$

where $\nabla_{\boldsymbol{W}} \mathcal{F}(\boldsymbol{W}; \boldsymbol{Z})$ is the gradient of the objective criterion with respect to the DNN weight matrix $\boldsymbol{W}$. Using the same nomenclature as defined in Eq. (6.3), the gradient with respect to the weights in the last layer is calculated as

$$\begin{aligned} \nabla_{w^L_{n,p}} \mathcal{F}(\boldsymbol{W}; \boldsymbol{Z}) &= -\Delta^L_{n,p} - 2\gamma_1 w^L_{n,p} \\ &- \frac{2\gamma_2}{k^2} \sum_{j=1}^{k} \omega_{ij} (z_{(i),p} - z_{(j),p}) \left(\frac{\partial z_{(i),p}}{\partial w^L_{n,p}} - \frac{\partial z_{(j),p}}{\partial w^L_{n,p}} \right), \end{aligned} \tag{6.6}$$

where $z_{(i),p}$ refers to the activation of the p^{th} unit in the output layer when the input vector is $\boldsymbol{x}_i$. The error signal, $\Delta^L_{n,p}$, is the same as the one specified in Eq. (6.3).

6.2.2 Architecture and Implementation

The computation of the gradient in Eq. (6.6) depends not only on the input vector, $\boldsymbol{x}_i$, but also its neighbors, $\boldsymbol{x}_j, j = 1, \ldots, k$, that belong to the same class. Therefore, MRDNN training can be broken down into two components. The first is the standard DNN component that minimizes a cross-entropy based error with respect to given targets. The second is the manifold regularization based component that focuses on penalizing a criterion related to preservation of neighborhood relationships.

An architecture for manifold regularized DNN training is shown in Figure 6.2. For each input feature vector, $\boldsymbol{x}_i$, k of its nearest neighbors, $\boldsymbol{x}_j, j = 1, \ldots, k$, belonging to the same class are selected. These $k + 1$ vectors are forward propagated through the network. This can be visualized as making $k + 1$ separate copies of the DNN one for the input vector and the remaining k for its neighbors. The DNN corresponding to the input vector, $\boldsymbol{x}_i$, is trained to minimize cross-entropy error with respect to a given target, c_i. Each copy-DNN corresponding to one of the selected neighbors, $\boldsymbol{x}_j$, is trained to minimize a function of the distance between its output, $\boldsymbol{z}_j$, and that corresponding to the input vector, $\boldsymbol{z}_i$. The

weights of all these copies are shared, and only an average gradient is used for weight-update. This algorithm is then extended for use in mini-batch training. Note that for all practical purposes, there is only one neural network; the copy-DNNs are only used for visualization for easy understanding. After training, only one set of weights are kept for inference and there is no additional computational complexity requirement.

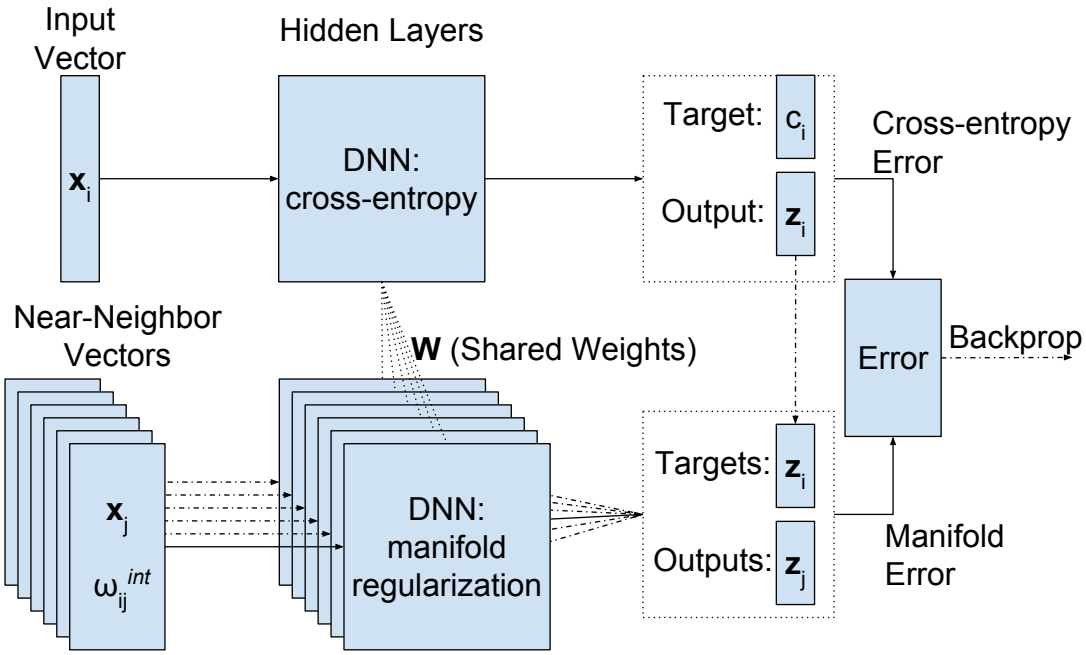

Fig. 6.2 Illustration of MRDNN architecture. For each input vector, $\boldsymbol{x}_i$, k of its neighbors, $\boldsymbol{x}_j$, belonging to the same class are selected and forward propagated through the network. For the neighbors, the target is set to be the output vector corresponding to $\boldsymbol{x}_i$. The scalar ω_{ij}^{int} represents the intrinsic affinity weights as defined in Eq. (3.1a).

It should be clear from this discussion that the relationships between a given vector and its neighbors play an important role in the weight update using EBP. The gradient updates in MRDNN depend not only on the current vector but also k of its neighbors. The objective criterion is penalized if the network maps a feature vector far from its neighbors in the source domain. This also leads to a more robust computation of the weight updates: if the current feature vectors, $\boldsymbol{x}_i$, is corrupted due to noise or is an outlier, ω_{ij} would be small for most of its neighbors, $j's$. For such a vector, the manifold error at the output of the neural network –given by the second term in Eq. (6.4)– would be very small. Therefore, such a data point would have less impact on the optimization of the neural network.

6.2.3 Preserving Neighborhood Relationships

This section describes a study conducted to characterize the effect of applying manifold based constraints on the behavior of a deep network. This is an attempt to quantify how neighborhood relationships between feature vectors are preserved within the hidden layers of a manifold regularized DNN. To this end, a measure referred to as the contraction ratio is investigated. A form of this measure is presented in [70].

In this work, the contraction ratio is defined as the ratio of distance between two feature vectors at the output of the first hidden layer to that at the input of the network. The average contraction ratio between a feature vector and its neighbors can be seen as a measure of locality preservation and compactness of its neighborhood. Thus, the evolution of the average contraction ratio for a set of vectors as a function of distances between them can be used to characterize the overall locality preserving behavior of a network. To this end, a subset of feature vectors not seen during the training are selected. The distribution of pair-wise distances for the selected vectors is used to identify a number of bins. The edges of these bins are treated as a range of radii around the feature vectors. An average contraction ratio is computed as a function of radius in a range $r_1 < r \leq r_2$ over all the selected vectors and their neighbors falling in that range as

$$CR(r_1, r_2) = \frac{1}{T} \sum_{i=1}^{T} \sum_{j \in \Phi} \frac{1}{k_\Phi} \cdot \frac{||\boldsymbol{z^1}_i - \boldsymbol{z^1}_j||^2}{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}, \tag{6.7}$$

where for a given vector $\boldsymbol{x}_i$, Φ represents a set of vectors $\boldsymbol{x}_j$ such that $r_1^2 < ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \leq r_2^2$, and k_Φ is the number of such vectors. $\boldsymbol{z^1}_i$ represents the output of the first layer corresponding to vector $\boldsymbol{x}_i$ at the input. It follows that a smaller contraction ratio represents a more compact neighborhood.

Figure 6.3 displays the contraction ratio of the output to input neighborhood sizes relative to the radius of the neighborhood in the input space for the DNN and MRDNN systems. The average contraction ratios between the input and the first layer's output features are plotted as functions of the median radius of the bins. It can be seen from the plots that the features obtained from a MRDNN represent a more compact neighborhood than those obtained from a DNN. Therefore it can be concluded that the hidden layers of a MRDNN are able to learn the manifold based local geometrical representation of the feature space. It should also be noted that the contraction ratio increases with the radius indicating

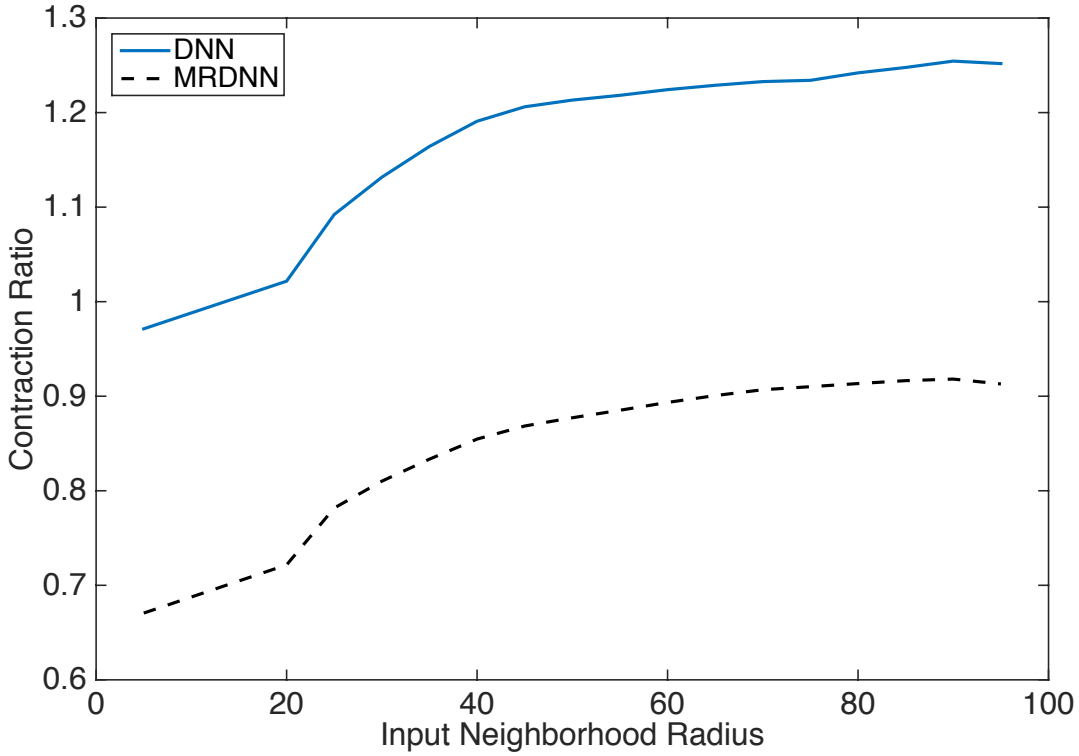

Fig. 6.3 Contraction ratio of the output to input neighborhood sizes as a function of input neighborhood radius.

that the effect of manifold preservation diminishes as one moves farther from a given vector. This is in agreement with the local invariance assumption over low-dimensional manifolds.

The auto-encoder based manifold learning methods, such as given in [70], can be seen as applying contraction in the directions perpendicular to the manifold (as these methods are trying to map the feature-vectors onto the manifold). In comparison, the presented graph based manifold regularization applies contraction (within each class) along the manifold.

6.3 Experimental Study

This section presents the experimental study conducted to evaluate the effectiveness of the proposed MRDNN framework in terms of ASR WER. The ASR performance of a MRDNN is presented and compared with that of a DNN without manifold regularization and the

traditional GMM-HMM systems. The experiments in this work are done on the Aurora-2 and the Aurora-4 tasks discussed in Section 2.8. The system setup are described in Section 6.3.1. The results of the experiments on the Aurora-2 task are presented in Section 6.3.2. The results and comparative performance of the proposed technique on the Aurora-4 task are presented in Section 6.3.3.

6.3.1 System Configuration

As a first step of DNN training, GMM-HMM systems are used for generating the context-dependent state alignments. These alignments are then used as the target labels for training the deep networks[1]. Both DNN and MRDNN systems include $L2$-norm regularization applied to the weights of the networks. The ASR performance of these models is evaluated in a tandem setup where a bottleneck deep network is used as feature extractor for a GMM-HMM system. For the deep networks, the class labels or targets at the output are set to be the CDHMM states obtained by a single pass force-alignment using the baseline GMM-HMM system. Both DNN and MRDNN systems are trained with $L2$ weight-decay with the value of the regularization coefficient γ_1 set to 0.0001. Additionally, in the MRDNN setup, the number of nearest neighbors, k, is set to 10, and γ_2 is set to 0.001.

For the Aurora-2 experiments, the deep networks have five hidden layers. The first four hidden layers have 1024 hidden units each, and the fifth layer is bottleneck layer with 40 hidden units. The deep networks take 117-dimensional input vectors that are created by concatenating 11 context frames of 12-dimensional MFCC features augmented with log energy. The number of units in the output layer is 181, which is equal to the number of CDHMM monophone states. The hidden units use either sigmoid or ReLUs as activation functions. For the Aurora-4 experiments, larger networks are used. There are seven hidden layers. The first six hidden layers have 2048 hidden units each, and the seventh layer is bottleneck layer with 40 hidden units. The deep networks take 429-dimensional input vectors that are created by concatenating 11 context frames of 12-dimensional MFCC features augmented with log energy, and first and second order difference. This larger network for Aurora-4 is in-line with other recent work [124], [125]. Keeping in-line with other work, the baseline GMM-HMM system is also modified to include higher number of states and wider pruning beam during HMM decoding [34]. This is the reason why the ASR WER results in

[1]Collectively, the term “deep network" is used to refer to both DNN and MRDNN configurations.

Table 6.3 are somewhat different than those presented in Table 3.3 in Chapter 3. Compared to the baseline Aurora-4 system used in Chapter 3, the system used here contains 3202 senones instead of 1206, and a beam width of 350 instead of 250[2].

For both Aurora-2 and Aurora-4, the soft-max nonlinearity is used in the output layer with the cross-entropy loss between the outputs and targets of the network as the error [118], [119]. After the EBP training, the 40-dimensional output features are taken from the bottleneck layer and de-correlated using PCA. Only features corresponding to the top 39 components are kept during the PCA transformation to match the dimensionality of the baseline system. The resultant features are used to train a GMM-HMM ASR system using a maximum-likelihood criterion. Although some might argue that PCA is not necessary after the compressed bottleneck output, in this work, a performance gain of 2-3% absolute is observed with PCA on the same bottleneck features for different noise and channel conditions on the Aurora-4 dataset. No difference in performance is observed for the clean data.

6.3.2 Results for the Aurora-2 Connected Digits Corpus

The ASR WER for the Aurora-2 speech-in-noise task are presented in Table 6.1. The models are trained on the mixed-noise set. The test results are presented in four separate tables each corresponding to a different noisy subset of the Aurora-2 test set. Each noisy subset is further divided into four subsets corresponding to noise corrupted utterances with 20 to 5dB SNRs. For each noise condition, the ASR results for features obtained from three different techniques are compared. The first row of each table, labeled 'GMM-HMM', contains the ASR WER for the baseline GMM-HMM system trained using MFCC features appended with first and second order differences. The second row, labeled 'DNN', displays results for the bottleneck features taken from the DNN. The final row, labeled 'MRDNN', presents the ASR WER results for the bottleneck features obtained from the MRDNN described in Section 6.2.1. For all the cases, the GMM-HMM and DNN configurations described in Section 6.3.1 are used. The initial learning rates for both systems are set to 0.001 and decreased exponentially with each epoch. Each system is trained for 40 epochs over the training set.

[2]If beamwidth is set to δ, it means that only models whose maximum log probability token falls within δ below the maximum for all models are kept for search, and the rest are pruned/deactivated. For more details see HTK documentation in [34].

Noise	Technique	SNR (dB)				
		Clean	20	15	10	5
Subway	GMM-HMM	1.76	2.99	4.00	6.21	11.89
	DNN	0.94	1.19	1.69	2.95	6.02
	MRDNN	**0.74**	**0.91**	**1.60**	**2.39**	**5.67**
Exhibition	GMM-HMM	1.89	3.34	3.83	6.64	12.72
	DNN	0.88	1.23	1.54	3.30	7.87
	MRDNN	**0.54**	**0.96**	**1.44**	**2.48**	**7.24**
Car	GMM-HMM	1.99	2.77	3.36	5.45	12.31
	DNN	0.96	1.05	1.85	2.98	6.92
	MRDNN	**0.78**	**0.84**	**1.37**	**2.59**	**6.38**
Average	GMM-HMM	1.88	3.03	3.73	6.10	12.31
	DNN	0.93	1.16	1.69	3.08	6.94
	MRDNN	**0.69**	**0.90**	**1.47**	**2.49**	**6.43**

Table 6.1: ASR performance of MRDNN system for mixed conditions training on the Aurora-2 corpus. WERs for GMM-HMM, DNN and MRDNN systems are given.

Two main observations can be made from the results presented in Table 6.1. First, both DNN and MRDNN provide significant reductions in ASR WER over the GMM-HMM system. The second observation can be made by comparing the ASR performance of DNN and MRDNN systems. It is evident from the results presented that the features derived from a manifold regularized network provide consistent gains over those derived from a network that is regularized only with the $L2$ weight-decay. The maximum relative WER gain obtained by using MRDNN over DNN is 38.64%.

In addition to the experiments on the mixed-conditions noisy datasets discussed in Table 6.1, experiments are also conducted to evaluate the performance of MRDNN for matched clean training and testing scenarios for the Aurora-2 set. To this end, the clean training sets of the Aurora-2 corpus is used for training the deep networks as well as building manifold based neighborhood graphs [90]. The results are presented in Table 6.2. It can be observed from the results presented in the table that MRDNN provides 21.05% relative improvements in WER over DNN for the Aurora-2 set.

	GMM-HMM	DNN	MRDNN	% Imp.
Aurora-2	0.93	0.57	0.45	21.05

Table 6.2: WERs for clean training and clean testing on the Aurora-2 speech corporus for GMM-HMM, DNN, and MRDNN models. The last column lists % WER improvements of MRDNN relative to DNN.

6.3.3 Results for the Aurora-4 Read News Corpus

The ASR WERs for the Aurora-4 task are given in Table 6.3. All three acoustic models in the table are trained on the Aurora-4 mixed-conditions set. The table lists ASR WERs for four sets, namely clean, additive noise (noise), channel distortion (channel), and additive noise combined with channel distortion (noise + channel). These sets are obtained by grouping together the fourteen subsets described in Section 2.8. The first row of the table, labeled 'GMM-HMM', provides results for the traditional GMM-HMM system trained using MFCC features appended with first and second order differences. The second row, labeled 'DNN', presents WERs corresponding to the features derived from a $L2$ regularized DNN system. The third row, labeled 'MRDNN', displays the WERs for features obtained from a MRDNN. Similar to the case for Aurora-2, L2 regularization with the coefficient γ_1 set to 0.0001 is used for training both the DNN and MRDNN. The initial learning rate is set to 0.003 and reduced exponentially for 40 epochs when the training is stopped.

	Clean	Noise	Channel	Noise + Channel
GMM-HMM	13.02	18.62	20.27	30.11
DNN	5.91	10.32	11.35	22.78
MRDNN	**5.30**	**9.50**	**10.11**	**21.90**

Table 6.3: ASR performance of MRDNN system for mixed conditions training on the Aurora-4 corpus. WERs for GMM-HMM, DNN, and MRDNN systems are given.

The results in Table 6.3 compare the ASR WER performance of MRDNN and DNN models for this corpus. Similar to the results for the Aurora-2 corpus, two main observations can be made by comparing the performance of the presented systems. First, both DNN and MRDNN provide very large reductions in WERs over the GMM-HMM for all conditions. Second, the trend of the WER reductions by using MRDNN over DNN is visible here as well.

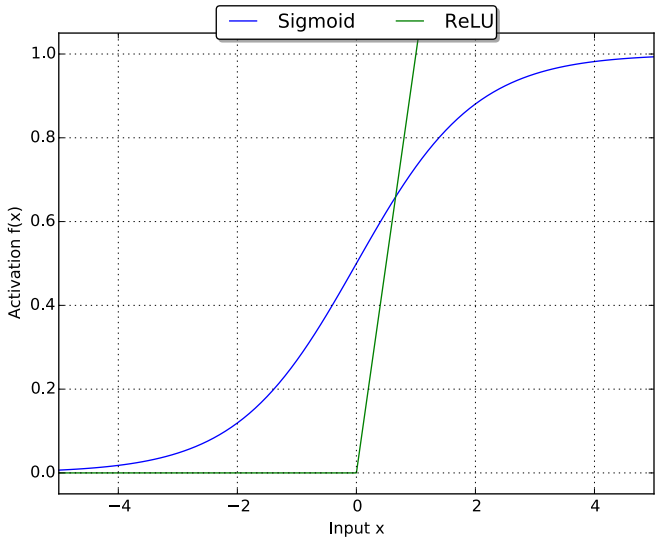

Fig. 6.4 Comparison of Sigmoid and ReLU activation functions.

The maximum relative WER reduction obtained by using MRDNN over DNN is 10.3%.

6.4 Discussion and Issues

There are a number of factors that can have an impact on the performance and application of DNN and MRDNN to ASR tasks. This section highlights some of the factors and issues affecting these techniques.

6.4.1 Sigmoid vs ReLU Hidden Units

There are a number of different nonlinearities that can be used as the activation functions in the hidden units of a neural network. Some of these include:

sigmoid:

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

tanh:

$$f(x) = tanh(x), \text{and}$$

ReLU:

$$f(x) = max(0, x),$$

where x is the input to the hidden unit. In this thesis work, the sigmoid and ReLU nonlinearities have been investigated. The two functions are graphically compared in Figure 6.4. Therefore, sigmoid is well suited to model probability whereas ReLU is better suited to model positive real numbers. The sigmoid function has range between 0 and 1, whereas the ReLU function is unbounded at the top. A number of recent studies have compared the benefits of using ReLU units over sigmoids for DNNs [69], [95], [116]. There are a number of reasons that make using ReLU units advantageous over using sigmoid units:

- ReLUs are piecewise linear, as shown in Figure 6.4. This leads to a simpler optimization with reduced computational complexity as they do not require division or exponentiation.

- Using ReLUs results in sparse and regularized networks. While sigmoid units produces small positive values when input to a unit is less than zero, ReLU units output exact zeros. This increased sparsity can also lead to a more generalizable network [95].

- Sigmoid units suffer from the vanishing gradient problem, *i.e.*, the gradient of the function vanishes for the input, x, close to -1 or 1. In contrast, the gradient for the ReLU units is 0 for $x < 0$ and 1 for $x > 0$. This constant gradient of ReLUs results in faster learning.

Because of the aforementioned advantages, increasingly more recent works in deep networks has adopted ReLU hidden units. In this thesis as well, ReLU units are found to be superior to sigmoid units when used with the DNN or MRDNN systems. Similar to other work, networks with ReLU units produces smaller errors and converge faster. Figure 6.5 compares the ReLU and sigmoid units for decrease in training error with epochs for the Aurora-2 corpus. It can be seen that the network with ReLU units shows faster drop in error and over-all smaller error.

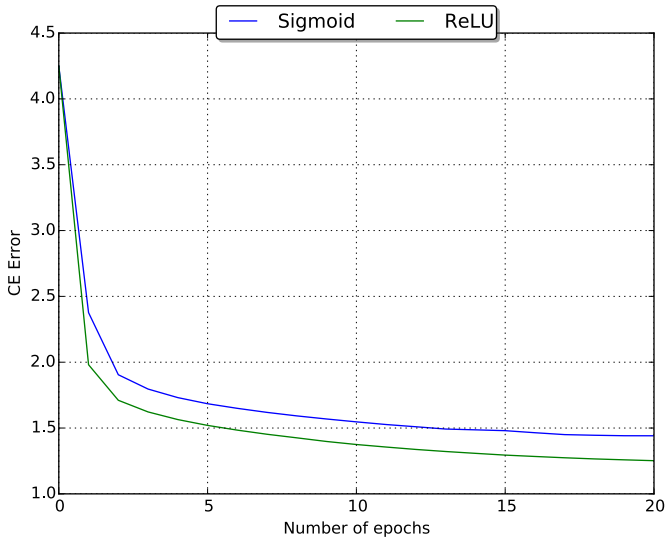

Fig. 6.5 Comparison of Sigmoid and ReLU hidden units for drop in CE error with epochs over the training data for DNN training on the Aurora-2 speech corpus.

Similarly, networks with ReLU units also lead to slightly improved ASR performance. The ASR WERs for the Aurora-2 corpus are compared in Table 6.4 when training the DNN and MRDNN systems with sigmoid and ReLU units. Two main observations can be made by comparing the WERs in the Table. The first is that using ReLU units provide much lower WERs for both the DNN and MRDNN system. The second observation is that MRDNN provides consistent reductions in WERs over DNN irrespective of the choice of unit activations. It is due to these evident benefits of ReLUs that most of the experiments in this thesis have used ReLU units.

Noise	Technique	SNR (dB)				
		Clean	20	15	10	5
Sigmoid	DNN	0.99	1.15	1.97	3.27	7.55
	MRDNN	0.74	0.97	1.60	2.68	6.59
ReLU	DNN	0.93	1.16	1.69	3.08	6.94
	MRDNN	0.69	0.90	1.47	2.49	6.43

Table 6.4: Comparison of Sigmoid and ReLU hidden units on the Aurora-2 speech corpus for DNN and MRDNN systems.

6.4.2 Computational Complexity

Though the inclusion of a manifold regularization factor in DNN training leads to encouraging gains in WER, it does so at the cost of additional computational complexity for parameter estimation. This additional cost for MRDNN training is two-fold. The first is the cost associated with calculating the pair-wise distances for populating the intrinsic affinity matrix, Ω_{int}. As discussed in Chapter 4, this computation complexity can be managed by using locality sensitive hashing for approximate nearest neighbor search without sacrificing ASR performance.

The second source of this additional complexity is the inclusion of k neighbors for each feature vector during forward and back propagation. This results in an increase in the computational cost by a factor of k. This cost can be managed by various parallelization techniques [56] and becomes negligible in the light of the massively parallel architectures of the modern processing units. This added computational cost is only relvent during the training of the networks and has no impact during the test phase or when the data is transformed using a trained network.

The networks in this work are training on Nvidia K20 graphics boards using tools developed on python based frameworks such as numpy, gnumpy and cudamat [35], [36]. For DNN training, each epoch over the Aurora-2 set took 240 seconds and each epoch over the Aurora-4 dataset took 480 seconds. In comparison, MRDNN training took 1220 seconds for each epoch over the Aurora-2 set and 3000 seconds for each epoch over the Aurora-4 set.

6.4.3 Exclusion of the Penalty Graph Term

In the applications of DML framework for feature space transformations, inclusion of both the intrinsic and penalty graphs terms, as discussed in Chapter 3, is found to be important [15], [16], [27]. For this reason, experiments in the previous work included both these terms [28]. However, in experiments performed on the Aurora-2 and Aurora-4 corpora described in Section 2.8, the gains achieved by including the penalty graph are found to be inconsistent across noise conditions and task domains. Results for MRDNN with and without including the penalty graph for the Aurora-2 and Aurora-4 corpora are given in Tables 6.5 and 6.6. For the Aurora-2 set, an average of the three test noises given in Table 6.1 are given. For the Aurora-4 set, same grouping for the test sets as given in Table 6.3 is used. The first rows of the tables, labeled 'MRDNN (only int.)', show the WERs when only the intrinsic

graph is used for manifold constraints. The second rows of the tables, labeled 'MRDNN (int. and pen.)', show WERs when both the intrinsic and penalty graphs are used for manifold constraints. It is apparent by comparing the two rows that the two results are very similar, and including the penalty graph does not provide additional consistent WER improvements. This may be because adding an additional term for discriminating between classes of speech feature vectors might not always impact the performance of DNNs since DNNs are inherently powerful discriminative models. However, including the penalty graph does add to the computational complexity. Therefore, the penalty graph based term is not included in any of the experiments presented in this work, and the manifold learning based expression given in Eq. (6.4) consists of the intrinsic component only.

	SNR (dB)				
	Clean	20	15	10	5
MRDNN (only int.)	0.69	0.90	1.47	2.49	6.43
MRDNN (int. and pen.)	0.70	0.92	1.40	2.53	6.42

Table 6.5: Effect of including the penalty graph term in MRDNN objective criterion. WERs for mixed noise training on the Aurora-2 speech corpus for MRDNN with and without penalty graph term when ReLU hidden units are used.

	Clean	Noise	Channel	Noise + Channel
MRDNN (only int.)	5.30	9.50	10.11	21.90
MRDNN (int. and pen.)	5.32	9.80	10.31	22.10

Table 6.6: Effect of including the penalty graph term in MRDNN objective criterion. WERs for mixed noise training on the Aurora-4 speech corpus for MRDNN with and without penalty graph term when ReLU hidden units are used.

6.4.4 Effect of Noise

In previous work, the authors have demonstrated that manifold learning techniques are very sensitive to the presence of noise [27]. This sensitivity can be traced to the Gaussian kernel scale factor, ρ, used for defining the local affinity matrices in Eq. (3.1a). This argument might apply to MRDNN training as well. Therefore, the performance of a MRDNN might be affected by the presence of noise. This is visible to some extent in the results presented

in Table 6.1, where the WER gains by using MRDNN over DNN vary with SNR level. On the other hand, the Aurora-4 test corpus contains a mixture of noise corrupted utterances at different SNR levels for each noise type. Therefore, only an average over all SNR levels per noise type is presented.

There is a need to conduct extensive experiments to investigate this effect as was done in [27]. However, if these models are affected by the presence of noise in a similar manner, additional gains could be derived by designing a method for building the intrinsic graphs separately for different noise conditions. During the DNN training, an automated algorithm could select a graph that best matches the estimated noise conditions associated with a given utterance. Training in this way could result in a MRDNN that is able to provide further gains in ASR performance in various noise conditions. An alternative approach would be to employ a noise-removal technique in the MRDNN training pipeline. Such a technique could be used either as a pre-processing step (for example: stacked denoising auto-encoders [65]) or in a multi-task learning fashion [126].

6.5 Alternating Manifold Regularized Training

Section 6.3 has demonstrated reductions in ASR WER by forcing the output feature vectors to conform to the local neighborhood relationships present in the input data. This is achieved by applying the underlying input manifold based constraints to the DNN outputs throughout the training. There have also been studies in literature where manifold regularization is used only for first few iterations of model training. For instance, authors in [127] have applied manifold regularization to multi-task learning. The authors have argued that optimizing deep networks by alternating between training with and without manifold based constraints can increase the generalization capacity of the model.

Motivated by these efforts, this section investigates a scenario in which the manifold regularization based constraints are used only for the first few epochs of the training. All layers of the networks are randomly initialized and then trained with the manifold based constraints for first 10 epochs. The resultant network is then further trained for 20 epochs using the standard EBP without manifold based regularization. Note that contrary to the pre-training approaches in deep learning, this is not a greedy layer-by-layer training.

ASR results for these experiments are given in Table 6.7 for the Aurora-2 dataset. For brevity, the table only presents the ASR WERs as an average over the four noise types

Noise	Technique	SNR (dB)				
		clean	20	15	10	5
Avg.	DNN	0.93	1.16	1.69	3.08	6.94
	MRDNN	0.69	0.90	1.47	2.49	6.43
	MRDNN_10	0.62	0.81	1.37	2.46	6.45

Table 6.7: Average WERs for mixed noise training and noisy testing on the Aurora-2 speech corpus for DNN, MRDNN and MRDNN_10 models.

described in Table 6.1 at each SNR level. In addition to the average WERs for the DNN and MRDNN, a row labeled 'MRDNN_10' is given. This row refers to the case where manifold regularization is used only for first 10 training epochs.

A number of observations can be made from the results presented in Table 6.7. First, it can be seen from the results in Table 6.7 that both MRDNN and MRDNN_10 training scenarios improve ASR WERs when compared to the standard DNN training. Second, MRDNN_10 training provides further reductions in ASR WERs for the Aurora-2 set. Experiments on the clean training and clean testing set of the Aurora-2 corpus also results in interesting comparison. In this case, the MRDNN_10 training improved ASR performance to 0.39% WER as compared to 0.45% for MRDNN and 0.57% for DNN training. That translates to 31.5% gain in ASR WER relative to DNNs.

The WER reductions achieved in these experiements are encouraging. Furthermore, this approach where manifold constraints are applied only for the first few epochs might lead to a more efficient manifold regularized training procedure because manifold regularized training has higher computational complexity than a DNN (as discussed in Section 6.4.2). However, unlike [127], this work only applied one cycle of manifold constrained-unconstrained training. It should be noted that these are preliminary experiments. Similar gains are not seen when these techniques are applied to the Aurora-4 dataset. Therefore, further investigation is required before making any substantial conclusions.

6.6 Conclusions and Future Work

This chapter has presented a framework for regularizing the training of DNNs using discriminative manifold learning based locality preserving constraints. The manifold based constraints are derived from a graph characterizing the underlying manifold of the speech

feature vectors. Empirical evidence has also been provided showing that the hidden layers of a MRDNN are able to learn the local neighborhood relationships between feature vectors. It has also been conjectured that the inclusion of a manifold regularization term in the objective criterion of DNNs results in a robust computation of the error gradient and weight updates. It has been shown through experimental results that the MRDNNs result in consistent reductions in ASR WERs that range up to 38.64% relative to the standard DNNs on the Aurora-2 corpus. For the mixed noise training on Aurora-4 corpus, the relative WER gains range up to 10%.

Chapter 7

Conclusions and Future Work

This thesis has investigated manifold learning based techniques and their applications to speech recognition. Discriminative manifold learning techniques for feature space transformations in speech recognition have been developed. The discriminative manifold learning based constraints allow preserving the local geometric relationships inherent in the speech feature vectors while increasing the discrimination between different classes of the vectors. It has been shown that the resultant features provide significant gains in ASR accuracy. Finally, the application of these manifold learning constraints to deep neural networks has also been explored. This chapter summarizes the work presented in this thesis and suggests potential topics for future research.

7.1 Summary of Contributions

The main contributions of this thesis work has been presented in Chapters 3 to 6. A summary of these contributions is presented as follows.

7.1.1 Discriminative Manifold Learning

Chapter 3 of this thesis has presented a family of DML based locality preserving feature space transformation techniques for ASR. The proposed approaches attempt to preserve the within class manifold based local relationships while at the same time maximize the separability between classes. This is achieved by embedding the feature vectors into undirected graphs by using nonlinear kernels and preserving or penalizing the local structure of the graphs.

Two approaches have been presented that rely on two different kernels utilizing Euclidean and cosine-correlation distance measures. When compared to well-known approaches such as LDA and LPP, the discriminative manifold learning algorithms demonstrated up to 30% reduction in WER. It has also been shown that the use of the cosine-correlation based distance measures is more robust than those based on Euclidean distances when speech is corrupted by noise. Furthermore, these performance gains have been shown to generalize across task domains and speaker populations.

7.1.2 Locality Sensitive Hashing

Despite the demonstrated effectiveness of the discriminative manifold learning based techniques, their application to speech and other application domains is hindered by their requirement of high computational complexity. Chapter 4 of this thesis has investigated the application of a fast approximate nearest neighbor search algorithm, known as LSH, in conjunction with the discriminative manifold learning techniques proposed in Chapter 3. ASR WER and execution times have been reported for the LPDA and CPDA methods with and without LSH. Performance comparisons have also been made between these approaches and the more widely used LDA. CPDA utilizes a cosine-correlation based distance measure to characterize the manifold domain relationships among feature vectors. For this reason, a cosine adaptation of E2LSH scheme has been chosen for hashing. It has been demonstrated that the use of LSH within the LPDA and CPDA frameworks leads to significant computational gains without adverse effect on ASR performance.

7.1.3 Noise aware Manifold Learning

Having addressed the issue of high computational complexity of manifold learning methods in Chapter 4, Chapter 5 of this thesis has investigated the effect of acoustic noise on the performance of manifold learning techniques for speech recognition. It has been found that the structural constraints associated with manifold learning approaches result in transformed features that are more sensitive to mismatch in acoustic conditions than untransformed MFCC features. It has also been shown that environment dependent performance degradation can be traced to the choice of the size of the local neighborhood used for defining local affinity matrices in manifold learning. These observations have led to a multi-model approach for improving the robustness of manifold learning based feature

space transformations, referred to here as noise-aware manifold learning. This involved automatic selection from a set of noise-matched LPDA transformations to find a transform that best matches the estimated noise conditions associated with a given utterance. The approach has been shown to reduce WER across a range of acoustic conditions with respect to LDA and LPP based feature space transformations.

7.1.4 Manifold Regularized Deep Neural Networks

Motivated by the effectiveness of DML methods when applied to feature space transformations for ASR, Chapter 6 of this thesis has explored the application of manifold learning to DNNs. This chapter has presented a framework for regularizing the training of DNNs using discriminative manifold learning based locality preserving constraints. The manifold based constraints have been applied to DNN training in order to preserve the underlying low-dimensional manifold based relationships between feature vectors while minimizing the cross-entropy loss between network outputs and targets. It has been shown that MRDNNs provide consistent reductions in ASR WERs that range up to 38.64% relative to DNNs trained without manifold regularization on the Aurora-2 corpus. For the Aurora-4 corpus, the WER gains range up to 10.3% relative on the mixed-noise training task. Empirical evidence has also been provided showing that the hidden layers of a MRDNN are able to learn the local neighborhood relationships between feature vectors. It has also been conjectured that the inclusion of a manifold regularization term in the objective criterion of DNNs results in a robust computation of the error gradient and weight updates.

7.2 Future Work

The future directions of the manifold learning framework presented in this thesis can be summarized in three general areas, namely the characterization of the local neighborhood, importance of supervision and interclass discrimination, and the implications of recent breakthroughs in speech recognition research. These are discussed separately in this section.

7.2.1 Characterization of the Local Neighborhood

Manifold learning based relationships between feature vectors are defined by their local neighborhoods. The structure of these local neighborhoods are characterized by the choice of distance metric, a non-linear kernel and its scaling. Chapter 3 of this thesis has discussed two separate kernels: one based on Euclidean distance and another based on cosine-correlation. It has been shown that, in the presence of noise, use of a cosine-correlation based kernel in estimating the manifold based relationships resulted in a higher recognition accuracy. Furthermore, Chapter 5 has shown that the shape and size of these neighborhoods are also affected by the presence of noise. Therefore, optimal choice of these parameters is dependent on the distribution of data and background conditions. While Chapter 6 of this thesis has demonstrated the effectiveness of manifold regularization for deep networks, a thorough study analyzing the effect of different kernels and distance metrics remains to be seen. Furthermore, it would also be interesting to extend the studies presented in the Chapter 5 of this thesis to characterize the impact of channel distortion on the application of manifold learning techniques to speech recognition.

7.2.2 Effective Graph Embedding

Another important open issue for future work is the computational complexity requirements for computing the neighborhood based relationships for graph-embedding. Chapter 4 of this thesis has shown successful use of LSH for reducing this complexity; however, application of these techniques to datasets worth thousands of hours of speech remains an arduous task. For efficient application of manifold learning techniques to speech recognition, new methods for defining the neighborhood relationships needs to be investigated.

7.2.3 Filter-bank features and application to large speech corpus

In recent years, two interesting findings have help further optimize the application of DNNs to ASR. The first is the use of filter-bank features (FBANK) instead of the conventional MFCC features. A number of researchers, starting with [57], have shown that when DCT is not applied to the log-compressed output of the Mel-filterbank, the resultant features lead to better behaving DNN models. These models also lead to better performance in ASR. In this thesis, only MFCC features have been used for training the DNN and MRDNN models.

While the author expect to obtain similar gains by using MRDNN over DNN with FBANK features, it would be interesting to see this verified by experiments.

The second finding is that typical regularization methods, such as weight-decay and dropout, might become unnecessary for DNNs, if thousands of hours of speech is used for training [68], [69]. Naturally, a question arises whether the proposed manifold based regularization methods will still maintain the observed gains over DNNs when trained on such large datasets. Owing to the optimization constraints based on strong local relationships between feature vectors, this should still be true, particularly in the presence of noise. However, such experiments have not been conducted in this thesis. One interesting experiment in this direction would be to compute the graph-embedding based affinity matrices on clean data, and use those to regularize deep net's training in the presence of noise. This work is left for future work.

7.2.4 Supervised vs Unsupervised Manifold Learning

The majority of the work in manifold learning literature has focused on unsupervised [2], [5], [8], [19], [128] or semi-supervised learning [120], [122], [129]. These techniques assume that by using manifold learning based constraints they can exploit the inherent relationships among the feature vectors or structure of the feature-space in order to separate naturally occurring clusters or classes in the data [1]. However, it has been shown in recent years that manifold learning based techniques can benefit from information about class distribution of the feature-space, either in a supervised or semi-supervised manner [4], [6], [12]. For example, authors in [14] and [16] have shown that features extracted using LPP lead to increased ASR accuracy when class based supervision is applied. Inspired by these studies, the work in this thesis has focused on supervised discriminative manifold learning; however, it raises a number of questions in relation to unsupervised and semi-supervised learning:

- the first question is how unlabeled data can be effectively exploited in order to reduce the performance gap between supervised and unsupervised approaches. This has been an open question in the research community for some time given the difficulty in labeling speech data [120]. In the context of deep networks, an obvious approach to this end is to simultaneously learn from large unlabeled datasets by applying manifold learning and from smaller labeled datasets by applying conventional back propagation. While previous approaches in this manner have not led to significant improvements

[130], many open issues remain. The issue most relevant to the work in this thesis is whether applying the proposed form of manifold regularization in deep learning on both labeled and unlabeled data can lead to worthwhile accuracy gains. An interesting approach would be to use multitask learning (see Section 7.2.5). Alternatively, the manifold regularization based techniques could be used for pre-training on a large unlabeled dataset, followed by fine-tuning on a smaller labeled set. This might lead to new possibilities for low-resource scenarios. Another open issue is whether additional unlabeled data can lead to further improvements in the performance when these techniques are applied in a semi-supervised manner.

- A second question is whether manifold learning can be used to minimize the impact of condition mismatch between training and usage scenarios on the performance of ASR systems. Chapter 5 of this thesis has shown promising results towards using the supervised discriminative manifold learning based feature space transformation techniques for increased noise robustness compared to similar unsupervised approaches. This should be investigated for the presently ubiquitous deep learning approaches in ASR. Furthermore, as the underlying manifold based relationships are best represented by clean data, it would also be interesting to investigate the noise robustness effects of these techniques when manifold based graph structures or relationships are computed on clean data and applied to noisy data.

7.2.5 Manifold Learning and recent breakthroughs in Speech Recognition

The technological landscape for ASR has seen a dramatic shift with the advent of deep learning. A number of new breakthroughs have been made in past few years and unforeseen ASR accuracy have been achieved. In some cases, the conventional GMM-HMM based speech recognition pipeline has been completely replaced with new end-to-end systems. This section discusses the potential of manifold learning in the context of these innovations.

Most popular of these is the use of long short term memory (LSTM) units based recurrent neural network (RNN) models in conjunction with connectionist temporal classification (CTC) for end-to-end speech recognition [131]–[135]. RNNs employ a feedback structure that enables them to take into account previous states of the network when estimating the current activations. In additon, RNNs with LSTM units are able to model the long term context that exists between speech frames. The gated and memory nodes based architecture

of LSTM units provides increased control over the memory and temporal context to RNNs [136]. As it is not clear how to define the manifold learning based neighborhood constraints for these sequential approaches, the application of the proposed manifold regularization based techniques in the context of RNNs remains a challenging yet interesting topic for future research.

In related work, speech recognition research has shown progress in the direction of feature free implementations. In these methods, the initial signal processing based features extraction steps are no longer needed, and the filter-banks are directly learned from raw speech waveforms of training data by a neural network based system [137]–[140]. These methods have the potential of discovering the hidden feature patterns in speech that the conventional signal processing based methods might have missed. Therefore, these methods might result in speech recognition systems with simpler implementations and a higher degree of recognition accuracy. Similar to the RNN work, it is not clear how to apply manifold learning to these methods, and whether this application would lead to gains in their recognition performance.

Another recent wave of work includes the application of multitask learning, in particular, to improving the behavior of deep learning algorithms for ASR acoustic models [126], [141]. Multitask learning adds additional related tasks to the main task and attempts to take advantage of interactions between them. The argument is that sharing information among related tasks can help the model in learning to perform those tasks more efficiently. The authors in [142] have argued that it may be better for a neural network to learn two or more related tasks together rather than to learn each separately. This, in a way, is motivated by learning in humans [143]. When applied to ASR, these learning algorithms regulate parameter estimation in deep networks by combining the loss function associated with phone or state classification with various secondary task dependent loss functions. The manifold regularization framework presented in this thesis is related to mutitask learning, where a DNN is trained to minimize the cross-entropy error between its input and targets as the primary task and to minimize a manifold learning based loss as the secondary task. However, further research needs to be done to fully establish manifold regularization in the multitask learning framework. Some of the issues to be investigated are the application of manifold regularization constraints at different layers and the number of shared layers between the tasks. Authors in [143] have also shown that incorporating two additional tasks can be more beneficial than incorporating one additional task. In this regard, it would be

interesting to see the effect of combining the manifold regularization task with recent work on noise robustness in a multitask learning framework [141]. This is particularly interesting given the noise robust behavior of discriminative manifold learning as discussed in Chapters 3 and 5 of this thesis.

Bibliography

[1] L. Saul and S. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *The Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003 (cit. on pp. iii, v, 2, 59, 113).

[2] J. Tenenbaum, "Mapping a manifold of perceptual observations," *Advances in neural information processing systems*, 1998 (cit. on pp. iii, v, 113).

[3] L. V. D. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning*, vol. 9, pp. 2579–2605, 2008 (cit. on pp. iii, v).

[4] X. Zhu, "Semi-supervised learning with graphs," PhD thesis, Carnegie Mellon University, 2005 (cit. on pp. iii, v, 113).

[5] Q. Gao, J. Liu, K. Cui, H. Zhang, and X. Wang, "Stable locality sensitive discriminant analysis for image recognition," *Neural Networks*, Mar. 2014. DOI: 10.1016/j.neunet.2014.02.009 (cit. on pp. iii, v, 113).

[6] T. Zhang, Y. Yan Tang, C. Philip Chen, Z. Shang, and B. Fang, "An approximate closed-form solution to correlation similarity discriminant analysis," *Neurocomputing*, pp. 1–15, Jan. 2014. DOI: 10.1016/j.neucom.2013.12.015 (cit. on pp. iii, v, 113).

[7] S. Gerber, T. Tasdizen, and R. Whitaker, "Dimensionality reduction and principal surfaces via kernel map manifolds," *2009 IEEE 12th International Conference on Computer Vision*, pp. 529–536, Sep. 2009. DOI: 10.1109/ICCV.2009.5459193 (cit. on pp. iii, v).

[8] D. Cai, X. He, K. Zhou, and J. Han, "Locality sensitive discriminant analysis," in *International Joint Conferences on Artificial Intelligence*, 2007, pp. 708–713 (cit. on pp. iii, v, 2, 9, 61, 113).

[9] H. Tang, S. M. Chu, and T. S. Huang, "Spherical discriminant analysis in semi-supervised speaker clustering," *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers on - NAACL '09*, no. June, p. 57, 2009. DOI: 10.3115/1620853.1620871 (cit. on pp. iii, v, 9, 46, 49).

[10] A. Jansen, S. Thomas, and H. Hermansky, "Intrinsic spectral analysis for zero and high resource speech recognition," in *Interspeech*, 2012 (cit. on pp. iii, v, 6, 38, 45, 91).

[11] A. Jansen and P. Niyogi, "Intrinsic spectral analysis," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1698–1710, Apr. 2013. DOI: 10.1109/TSP.2013.2238931 (cit. on pp. iii, v).

[12] ——, "Semi-supervised learning of speech sounds," *Proceedings of INTERSPEECH 2007*, 2007 (cit. on pp. iii, v, 113).

[13] ——, "Intrinsic fourier analysis on the manifold of speech sounds," in *ICASSP: IEEE International Conference on Acoustics Speech and Signal Processing*, 2006 (cit. on pp. iii, v, 2, 5, 6, 45).

[14] Y. Tang and R. Rose, "A study of using locality preserving projections for feature extraction in speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, NV: IEEE, Mar. 2008, pp. 1569–1572. DOI: 10.1109/ICASSP.2008.4517923 (cit. on pp. iii, v, 5, 35, 36, 45, 55, 56, 61, 63, 77, 113).

[15] V. S. Tomar and R. C. Rose, "A family of discriminative manifold learning algorithms and their application to speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language processing*, vol. 22, no. 1, pp. 161–171, 2014. DOI: 10.1109/TASLP.2013.2286906 (cit. on pp. iii, v, 8, 9, 45, 46, 77, 104).

[16] ——, "Application of a locality preserving discriminant analysis approach to asr," in *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, Montreal, QC, Canada: IEEE, Jul. 2012, pp. 103–107. DOI: 10.1109/ISSPA.2012.6310443 (cit. on pp. iii, v, 8, 9, 45, 46, 51, 53, 78, 104, 113).

[17] ——, "A correlational discriminant approach to feature extraction for robust speech recognition," in *Interspeech*, Portland, OR, USA, 2012 (cit. on pp. iii, v, 9, 45, 46).

[18] S. Yan, D. Xu, B. Zhang, *et al.*, "Graph embedding and extensions: a general framework for dimensionality reduction.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 40–51, Jan. 2007. DOI: 10.1109/TPAMI.2007.12 (cit. on pp. iv, vi, 8, 9, 35, 45, 46, 51, 63).

[19] X. He and P. Niyogi, "Locality preserving projections," in *Neural Information Processing Systems (NIPS)*, 2002 (cit. on pp. iv, vi, 2, 5, 6, 32, 36, 45, 63, 77, 113).

[20] Y. Ma, S. Lao, E. Takikawa, and M. Kawade, "Discriminant analysis in correlation similarity measure space," *Proceedings of the 24th international conference on Machine learning - ICML '07*, no. 1, pp. 577–584, 2007. DOI: 10.1145/1273496.1273569 (cit. on pp. iv, vi, 2, 6, 9, 45, 46, 49, 59, 61, 63).

[21] H. Xiong, M. N. S. Swamy, and M. O. Ahmad, "Optimizing the kernel in the empirical feature space.," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 16, no. 2, pp. 460–74, Mar. 2005. DOI: 10.1109/TNN.2004.841784 (cit. on pp. iv, vi, 61, 78, 81).

[22] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth Annual ACM Symposium on Theory of Computing*, 1998, pp. 604–613 (cit. on pp. iv, vi, 9, 64).

[23] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," *Proceedings of the twentieth annual symposium on Computational geometry - SCG '04*, p. 253, 2004. DOI: 10.1145/997817.997857 (cit. on pp. iv, vi, 9, 64–66, 69).

[24] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing using stable distributions," in *Nearest-neighbor methods in learning and vision: Theory and practice*, G. Shakhnarovich, T. Darrell, and P. Indyk, Eds., 1st ed., MIT Press, 2006, ch. 3, pp. 55–67 (cit. on pp. iv, vi, 9, 64, 73).

[25] V. S. Tomar and R. C. Rose, "Locality sensitive hashing for fast computation of correlational manifold learning based feature space transformations," in *Interspeech*, 2013, pp. 2–6 (cit. on pp. iv, vi, 61, 63).

[26] ——, "Efficient manifold learning for speech recognition using locality sensitive hashing," in *ICASSP: IEEE International Conference on Acoustics Speech and Signal Processing*, Vancouver, BC, Canada, 2013 (cit. on pp. iv, vi, 61, 63).

[27] ——, "Noise aware manifold learning for robust speech recognition," in *ICASSP: IEEE International Conference on Acoustics Speech and Signal Processing*, Vancouver, BC, Canada, 2013 (cit. on pp. iv, vi, 9, 77, 104–106).

[28] ——, "Manifold regularized deep neural networks," in *Interspeech*, 2014 (cit. on pp. iv, vi, 87, 104).

[29] ——, "Manifold regularized deep neural networks for automatic speech recognition," in *ASRU*, 2015 (cit. on pp. iv, vi, 87).

[30] F. Grézl and M. Karafiát, "Probabilistic and bottle-neck features for lvcsr of meetings," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007 (cit. on pp. iv, vi, 88).

[31] H. Hermansky, D. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional hmm systems," in *ICASSP*, 2000, pp. 1–4 (cit. on pp. iv, vi, 42, 88).

[32] A. Faria and N. Morgan, "Corrected tandem features for acoustic model training," in *IEEE International Conference on Acoustics Speech and Signal Processing*, 2008 (cit. on pp. iv, vi, 42, 88).

[33] G. Hinton, L. Deng, and D. Yu, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, pp. 1–27, 2012 (cit. on pp. iv, vi, 19, 31, 42, 88).

[34] University of Cambridge and Microsoft Research, *HTK speech recognition toolkit, http://htk.eng.cam.ac.uk/* (cit. on pp. xviii, 97, 98).

[35] V. Mnih, "Cudamat : a cuda-based matrix class for python," Department of Computer Science, University of Toronto, Tech. Rep., 2009 (cit. on pp. xviii, 104).

[36] T. Tieleman, "Gnumpy : an easy way to use gpu boards in python," Department of Computer Science, University of Toronto, Tech. Rep., 2010 (cit. on pp. xviii, 104).

[37] L. Tenbosch and K. Kirchhoff, "Bridging the gap between human and automatic speech recognition," *Speech Communication*, vol. 49, no. 5, pp. 331–335, May 2007. DOI: 10.1016/j.specom.2007.03.001 (cit. on p. 2).

[38] S. M. Chu, H. Tang, T. S. Huang, I. B. M. T. J. Watson, and Y. Heights, "Locality preserving speaker clustering," pp. 494–497, 2009 (cit. on p. 2).

[39] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization : a geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning*, vol. 1, pp. 1–36, 2006 (cit. on pp. 2, 36–38).

[40] R. Lippmann, "Speech recognition by machines and humans," *Speech communication*, 1997 (cit. on p. 3).

[41] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*, 1st. Prentice Hall PTR, 2001 (cit. on pp. 3, 14, 23, 32).

[42] D. O'Shaughnessy, *Speech Communication: Human and Machine*, 2nd. IEEE, 2002 (cit. on pp. 3, 13).

[43] A. Ljolje, "The importance of cepstral parameter correlations in speech recognition," *Computer Speech & Language*, vol. 8, no. 3, pp. 223–232, Jul. 1994. DOI: 10.1006/csla.1994.1011 (cit. on pp. 4, 17).

[44] S. Furui, "Speaker independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 52–59, Feb. 1986 (cit. on pp. 4, 17).

[45] J. Hernando, "Maximum likelihood weighting of dynamic speech features for cdhmm speech recognition," in *ICASSP: IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 1997, pp. 1267–1270 (cit. on p. 4).

[46] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st. Springer, 2006 (cit. on pp. 4, 26, 63).

[47] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd. Wiley Interscience, 2000 (cit. on pp. 5, 32–34, 59, 63, 77).

[48] R. Haeb-Umbach and H. Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition," in *ICASSP: IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992, 13–16 vol. 1. DOI: 10.1109/ICASSP.1992.225984 (cit. on p. 5).

[49] K. Beulen, L. Welling, and H. Ney, "Experiments with linear feature extraction in speech recognition," in *European Conference on Speech Communication and Technology*, 1995 (cit. on pp. 5, 32, 63, 77).

[50] N. Kumar, "Investigation of silicon-auditory models and generalization of linear discriminant analysis for improved speech recognition," PhD thesis, Johns Hopkins University, Baltimore, MD, 1997 (cit. on pp. 5, 33).

[51] K. N. Stevens, *Acoustic Phoenetics*. Cambridge, MA, USA: MIT Press, 1998 (cit. on pp. 5, 45).

[52] H. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*. Kluwer Academic Publishers, 1994 (cit. on pp. 6, 8, 19, 39, 41).

[53] N. Morgan and H. Bourlard, "An introduction to hybrid hmm/connectionist continuous speech recognition," *IEEE Signal Processing Magazine*, 1995 (cit. on pp. 6, 19, 39).

[54] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language processing*, pp. 1–13, 2012 (cit. on pp. 6, 42, 88).

[55] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "An application of pretrained deep neural networks to large vocabulary conversational speech recognition," in *Interspeech*, 2012, pp. 3–6 (cit. on pp. 6, 88).

[56] T. N. Sainath, B. Kingsbury, B. Ramabhadran, *et al.*, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *IEEE Workshop on Automatic Speech Recognition & Understanding*, Dec. 2011, pp. 30–35. DOI: 10.1109/ASRU.2011.6163900 (cit. on pp. 6, 88, 104).

[57] A. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," *Neural Networks*, pp. 6–9, 2012 (cit. on pp. 6, 7, 87, 112).

[58] D. Erhan, P. Manzagol, and Y. Bengio, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR Workshop and Conference Procedings*, vol. 5, 2009, pp. 153–160 (cit. on pp. 7, 87).

[59] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *JMLR Workshop: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, 2010, pp. 249–256 (cit. on p. 7).

[60] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, 2006 (cit. on pp. 7, 88).

[61] D. Yu, L. Deng, and G. E. Dahl, "Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition," in *NIPS workshop on Deep Learning and Unsupervised Feature Learning*, 2010 (cit. on pp. 7, 88).

[62] G. Hinton, "A practical guide to training restricted boltzmann machines," Univ. of Toronto, Tech. Rep., 2010 (cit. on p. 7).

[63] Y. Bengio and P. Lamblin, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2007 (cit. on p. 7).

[64] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *IEEE Workshop on Automatic Speech Recognition & Understanding*, Dec. 2011, pp. 24–29. DOI: 10.1109/ASRU.2011.6163899 (cit. on pp. 7, 88).

[65] P. Vincent, H. Larochelle, and I. Lajoie, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010 (cit. on pp. 7, 87, 88, 106).

[66] H. Larochelle and Y. Bengio, "Exploring strategies for training deep neural networks," *The Journal of Machine Learning Research*, vol. 1, pp. 1–40, 2009 (cit. on pp. 7, 88).

[67] C. Plahl and T. Sainath, "Improved pre-training of deep belief networks using sparse encoding sysmmertric machines," in *ICASSP: IEEE International Conference on Acoustics Speech and Signal Processing*, 2012, pp. 4165–4168 (cit. on p. 7).

[68] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *Proc. ICASSP*, 2013 (cit. on pp. 7, 113).

[69] L. Deng, J. Li, J. Huang, K. Yao, and D. Yu, "Recent advances in deep learning for speech research at microsoft," *ICASSP 2013*, pp. –4, 2013 (cit. on pp. 7, 102, 113).

[70] S. Rifai, P. Vincent, X. Muller, G. Xavier, and Y. Bengio, "Contractive auto-encoders : explicit invariance during feature extraction," in *International Conference on Machine Learning*, vol. 85, 2011, pp. 833–840 (cit. on pp. 7, 88, 95, 96).

[71] S. Rifai, G. Mesnil, P. Vincent, *et al.*, "Higher order contractive auto-encoder," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2011 (cit. on p. 7).

[72] Y. Fu and T. Huang, "Correlation embedding analysis," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, IEEE, 2008, pp. 1696–1699 (cit. on pp. 9, 46, 49, 59, 61).

[73] S. Young, "A review of large-vocabulary continuous-speech recognition," *IEEE Signal Processing Magazine*, 1996 (cit. on pp. 12, 13).

[74] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989. DOI: 10.1109/5.18626 (cit. on pp. 12, 19, 20).

[75] B. X. Huang, J. Baker, and R. Reddy, "A historical perspective of speech recognition," *Communications of The ACM*, vol. 57, no. 1, pp. 94–103, 2014 (cit. on p. 14).

[76] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuous spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, pp. 357–366, Aug 1980 (cit. on p. 15).

[77] P. Mermelstein, "Distrance measures for speech recognition – psychological and instrumental," in *Joint Workshop on Pattern Recognition and Artifical Intelligence*, 1976 (cit. on p. 16).

[78] S. K. Scott and I. S. Johnsrude, "The neuroanatomical and functional organization of speech perception.," *Trends in neurosciences*, vol. 26, no. 2, pp. 100–7, Feb. 2003. DOI: 10.1016/S0166-2236(02)00037-1 (cit. on p. 16).

[79] H. Sakoe and S. Chiba, "Dynamic programming optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978 (cit. on p. 19).

[80] G. M. White and R. B. Neely, "Speech recognition experiments with linear prediction, bandpass filtering, and dynamic programming," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, pp. 183–188, 1976 (cit. on p. 19).

[81] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probablistic functions of markov process and to a model for ecology," *Bull. Amer. Math. Soc.*, vol. 73, no. 2, pp. 360–363, 1966 (cit. on pp. 19, 22).

[82] F. Jelinek, *Statistical Methods for Speech Recognition*. The MIT Press: London, 1999 (cit. on p. 19).

[83] J. K. Baker, "The dragon system-an overview," *IEEE Transactions on Speech and Audio Processing*, vol. 23, no. 1, pp. 24–29, 1975 (cit. on p. 19).

[84] F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, vol. 64, no. 4, pp. 532–556, 1976. DOI: 10.1109/PROC.1976.10159 (cit. on p. 19).

[85] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, Feb. 1970. DOI: 10.1214/aoms/1177697196 (cit. on p. 22).

[86] L. R. Welch, "Hidden markov models and the baum-welch algorithm," *IEEE Information Theory Society Newsletter*, vol. 53, no. 4, pp. 1, 10–13, 2003 (cit. on p. 22).

[87] A. P. Dempster, A. P. Dempster, N. M. Laird, *et al.*, "Maximum likelihood from incomplete data via the em algorithm," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, no. 1, pp. 1–38, 1977. DOI: 10.1.1.133.4884 (cit. on p. 23).

[88] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the em algorithm," *Society for Industrial and Applied Mathematics*, vol. 1984, no. April 1984, pp. 195–239, 1982. DOI: 10.1137/1026034 (cit. on p. 23).

[89] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967. DOI: 10.1109/TIT.1967.1054010 (cit. on p. 28).

[90] H. Hirsch and D. Pearce, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Automatic Speech Recognition: Challenges for the next millenium*, 2000 (cit. on pp. 30, 42, 43, 53, 99).

[91] N. Parihar and J. Picone, "Aurora working group : dsr front end lvcsr evaluation," European Telecommunications Standards Institute, Tech. Rep., 2002 (cit. on pp. 30, 42–44).

[92] M. J. F. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 272–281, May 1999. DOI: 10.1109/89.759034 (cit. on pp. 32, 33, 38, 39, 52).

[93] ——, "Maximum likelihood multiple subspace projections for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 2, pp. 37–47, 2002 (cit. on p. 33).

[94] G. Saon and M. Padmanabhan, "Maximum likelihood discriminant feature spaces," in *ICASSP: IEEE International Conference on Acoustics Speech and Signal Processing*, 2000, pp. 1129–1132 (cit. on pp. 33, 38, 53).

[95] M. Zeiler, M. Ranzato, R. Monga, and M. Mao, "On rectified linear units for speech processing," pp. 3–7, 2013 (cit. on pp. 40, 88, 102).

[96] Z. Tüske, M. Sundermeyer, R. Schlüter, and H. Ney, "Context-dependent mlps for lvcsr: tandem, hybrid or both," in *Interspeech*, Portland, OR, USA, 2012 (cit. on pp. 42, 88).

[97] D. Mansour and B. Juang, "A family of distortion measures based upon projection operation for robust speech recognition," *Speech and Signal Processing, IEEE*, vol. 37, no. 11, pp. 4–7, 1989. DOI: 10.1109/29.46548 (cit. on pp. 48, 59, 77).

[98] B. A. Carlson and M. A. Clements, "A projection-based likelihood measure for speech recognition in noise," *Audio*, vol. 2, no. 1, 1994 (cit. on pp. 49, 59).

[99] J.-T. Chien, H.-C. Wang, and L.-m. Lee, "A novel projection-based likelihood measure for noisy speech recognition," *Speech Communication*, vol. 24, no. 4, pp. 287–297, Jul. 1998. DOI: 10.1016/S0167-6393(98)00024-7 (cit. on p. 49).

[100] Q. Zhu and A. Alwan, "Non-linear feature extraction for robust speech recognition in stationary and non-stationary noise," *Computer speech & language*, vol. 17, pp. 381–402, 2003. DOI: 10.1016/S0885-2308(03)00026-3 (cit. on p. 53).

[101] C. Breslin, "Generation and combination of complementary systems for automatic speech recognition," PhD thesis, Cambridge University, 2008 (cit. on p. 53).

[102] Y. Shekofteh, "Comparison of linear based feature transformations to improve speech recognition performance," in *Electrical Engineering (ICEE), 19th Iranian Conference on*, 2011 (cit. on p. 53).

[103] G. Saon, J. Huerta, and E. Jan, "Robust digit recognition in noisy environments: the ibm aurora 2 system," in *INTERSPEECH*, 2001, pp. –3 (cit. on p. 55).

[104] S. Cox, "The gillick test: a method for comparing two speech recognisers tested on the same data," NASA STI/Recon Technical Report N, Tech. Rep., 1988 (cit. on pp. 55, 57).

[105] S. Har-Peled, P. Indyk, and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," *Theory of Computing*, vol. 8, no. 1, pp. 321–350, 2012. DOI: 10.4086/toc.2012.v008a014 (cit. on pp. 61, 69).

[106] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality sensitive hashing: a comparison of hash function types and querying mechanisms," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1348–1358, Aug. 2010. DOI: 10.1016/j.patrec.2010.04.004 (cit. on p. 64).

[107] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing - STOC '02*, p. 380, 2002. DOI: 10.1145/509961.509965 (cit. on pp. 65, 67, 69, 70).

[108] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, Dec. 2011, pp. 401–406. DOI: 10.1109/ASRU.2011.6163965 (cit. on pp. 65, 67, 69, 70).

[109] V. M. Zolotarev, "One-dimensional stable distributions," in *Vol. 65 of Translations of Mathematical Monographs*, American Mathematical Society, 1986 (cit. on p. 66).

[110] D. Ravichandran, P. Pantel, E. Hovy, M. Rey, and I. S. I. Edu, "Randomized algorithms and nlp : using locality sensitive hash functions for high speed noun clustering" (cit. on pp. 67, 69, 70).

[111] A. D. L. Torre, J. C. Segura, C. Benitez, and J. Ramirez, "Speech recognition under noise conditions: compensation methods," in *Robust Speech Recognition and Understanding*, June, M. Grimm and K. Kroschel, Eds., InTech Education and Publishing, 2007, ch. 25, pp. 439–460 (cit. on p. 79).

[112] M. J. F. Gales and P. C. Woodland, "Mean and variance adaptation within the mllr framework," *Computer Speech and Language*, pp. 249–264, 1996 (cit. on p. 80).

[113] J. Wang, H. Lu, K. Plataniotis, and J. Lu, "Gaussian kernel optimization for pattern classification," *Pattern Recognition*, vol. 42, no. 7, pp. 1237–1247, Jul. 2009. DOI: 10.1016/j.patcog.2008.11.024 (cit. on p. 81).

[114] C. Kim and R. M. Stern, "Robust signal-to-noise ratio estimation based on waveform amplitude distribution analysis," *In Practice*, pp. 2598–2601, 2008 (cit. on pp. 82, 83).

[115] M. Vondrasek and P. Pollak, "Methods for speech snr estimation: evaluation tool and analysis of vad dependency," *Radioengineering*, vol. 14, no. 1, p. 7, 2005 (cit. on pp. 82, 83).

[116] G. Dahl, T. Sainath, and G. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," *ICASSP*, 2013 (cit. on pp. 88, 102).

[117] Y. Bengio, "Deep learning of representations: looking forward," *Statistical Language and Speech Processing*, pp. 1–37, 2013 (cit. on p. 88).

[118] R. Dunne and N. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," in *Proc. 8th Aust. Conf. on the Neural Networks*, 1997, pp. 1–5 (cit. on pp. 91, 98).

[119] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: a theoretical and experimental comparison.," in *Interspeech*, 2013, pp. 2–6 (cit. on pp. 91, 98).

[120] A. Subramanya and J. Bilmes, "The semi-supervised switchboard transcription project," *Interspeech*, 2009 (cit. on pp. 91, 113).

[121] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," *Proceedings of the 25th international conference on Machine learning - ICML '08*, pp. 1168–1175, 2008. DOI: 10.1145/1390156.1390303 (cit. on p. 91).

[122] F. Ratle, G. Camps-valls, S. Member, and J. Weston, "Semi-supervised neural networks for efficient hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–12, 2009 (cit. on pp. 91, 113).

[123] J. Chen and L. Deng, "A primal-dual method for training recurrent neural networks constrained by the echo-state property," *Proc. ICLR*, pp. 1–17, 2014 (cit. on p. 91).

[124] R. Su, X. Xie, X. Liu, and L. Wang, "Efficient use of dnn bottleneck features in generalized variable parameter hmms for noise robust speech recognition," in *Interspeech*, 2015 (cit. on p. 97).

[125] M. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," *Proc. ICASSP*, pp. 7398–7402, 2013 (cit. on p. 97).

[126] Z. Chen, S. Watanabe, H. Erdogan, and J. R. Hershey, "Speech enhancement and recognition using multi-task learning of long short-term memory recurrent neural networks," in *Interspeech*, 2015, pp. 3274–3278 (cit. on pp. 106, 115).

[127] A. Agarwal, S. Gerber, and H. Daume, "Learning multiple tasks using manifold regularization," in *Advances in neural information processing systems*, 2010, pp. 1–9 (cit. on pp. 106, 107).

[128] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *Proceedings of the 25th international conference on Machine learning - ICML '08*, pp. 1096–1103, 2008. DOI: 10.1145/1390156.1390294 (cit. on p. 113).

[129] R. Hsiao, T. Ng, and F. Grezl, "Discriminative semi-supervised training for keyword search in low resource languages," *ASRU*, pp. 440–445, 2013 (cit. on p. 113).

[130] J. Malkin, A. Subramanya, and J. Bilmes, "A semi-supervised learning algorithm for multi-layered perceptrons," Tech. Rep. 206, 2009 (cit. on p. 114).

[131] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification," *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pp. 369–376, 2006. DOI: 10.1145/1143844.1143891 (cit. on p. 114).

[132] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," *ICML 2014*, vol. 32, no. 1, pp. 1764–1772, 2014 (cit. on p. 114).

[133] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *ArXiv preprint*, 2015. arXiv: 1507.06947 (cit. on p. 114).

[134] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *ArXiv preprint*, no. Cd, 2014. arXiv: arXiv:1402.1128v1 (cit. on p. 114).

[135] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: end-to-end speech recognition using deep rnn models and wfst-based decoding," in *ArXiv preprint*, 2015. arXiv: 1507.08240 (cit. on p. 114).

[136] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1–32, 1997. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 115).

[137] D. Palaz, M. Magimai, and R. Collobert, "Analysis of cnn-based speech recognition system using raw speech as input," in *Interspeech*, 2014 (cit. on p. 115).

[138] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, *et al.*, "Improvements to filterbank and delta learning within a deep neural network framework," in *ICASSP*, 2014, pp. 6889–6893 (cit. on p. 115).

[139] M. Bhargava and R. Rose, "Architectures for deep neural network based acoustic models defined over windowed speech waveforms," in *Interspeech*, 2015 (cit. on p. 115).

[140] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Interspeech*, 2015 (cit. on p. 115).

[141] Z. Huang, J. Li, S. M. Siniscalchi, *et al.*, "Rapid adaptation for deep neural networks through multi-task learning," in *Interspeech*, 2015, pp. 3625–3629 (cit. on pp. 115, 116).

[142] R. Caruana, "Multitask learning," in *Machine Learning*, 1, vol. 28, 1997, pp. 41–75. DOI: 10.1023/A:1007379606734 (cit. on p. 115).

[143] Y. Lu, F. Lu, S. Sehgal, *et al.*, "Multitask learning in connectionist speech recognition," in *Australian International conference on Speech Science and Technology*, 2004 (cit. on p. 115).