Subspace and Graph Methods to Leverage Auxiliary Data for Limited Target Data Multi-Class Classification, Applied to Speaker Verification

by

Zahi Nadim Karam

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

Author . . . .
Department of Electrical Engineering and Computer Science
May 20, 2011

Certified by . . . .
William M. Campbell
Member of Technical Staff at MIT Lincoln Laboratory
Thesis Supervisor

Certified by . . . .
Alan V. Oppenheim
Ford Professor of Engineering
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Subspace and Graph Methods to Leverage Auxiliary Data for Limited Target Data Multi-Class Classification, Applied to Speaker Verification

by

Zahi Nadim Karam

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering

Abstract

Multi-class classification can be adversely affected by the absence of sufficient target (in-class) instances for training. Such cases arise in face recognition, speaker verification, and document classification, among others. Auxiliary data-sets, which contain a diverse sampling of non-target instances, are leveraged in this thesis using subspace and graph methods to improve classification where target data is limited.

The auxiliary data is used to define a compact representation that maps instances into a vector space where inner products quantify class similarity. Within this space, an estimate of the subspace that constitutes within-class variability (e.g. the recording channel in speaker verification or the illumination conditions in face recognition) can be obtained using class-labeled auxiliary data. This thesis proposes a way to incorporate this estimate into the SVM framework to perform nuisance compensation, thus improving classification performance. Another contribution is a framework that combines mapping and compensation into a single linear comparison, which motivates computationally inexpensive and accurate comparison functions.

A key aspect of the work takes advantage of efficient pairwise comparisons between the training, test, and auxiliary instances to characterize their interaction within the vector space, and exploits it for improved classification in three ways. The first uses the local variability around the train and test instances to reduce false-alarms. The second assumes the instances lie on a low-dimensional manifold and uses the distances along the manifold. The third extracts relational features from a similarity graph where nodes correspond to the training, test and auxiliary instances.

To quantify the merit of the proposed techniques, results of experiments in speaker verification are presented where only a single target recording is provided to train the classifier. Experiments are performed on standard NIST corpora and methods are compared using standard evaluation metrics: detection error trade-off curves, minimum decision costs, and equal error rates.

Thesis Supervisor: William M. Campbell
Title: Member of Technical Staff at MIT Lincoln Laboratory

Thesis Supervisor: Alan V. Oppenheim
Title: Ford Professor of Engineering

Acknowledgments

I would first like to thank my advisers Bill and Al. Bill, it has been a great pleasure working with you and a tremendous learning experience. It was a long and challenging journey, yet your guidance, mentorship, patience and light-heartedness made it an enjoyable and memorable one. Al, thank you for welcoming me into the DSPG family and looking out for me from my first day at MIT. Without your help and guidance over these years I would have surely been lost. Also, thank you for encouraging me to always look at the big picture. Bill and Al, I hope this thesis is the start of a life-long friendship.

Next, I'd like to thank my thesis committee Doug and Tommi. I truly appreciated your feedback and comments.

I'd also like to thank MIT for the experience and the opportunity.

DSPG, you have been my home at MIT, and I could not imagine a better group to belong to. Thank you for the engaging group meetings and friendly group atmosphere. Particularly: Tom, it's been great sharing an office with you, we had some good times. Dennis and Jon-Paul thanks for all the great discussions. Petros and Sourav, thanks for your mentorship and friendship. Melanie thanks for making the group feel like a family.

LL Group 62, thank you for the friendly and engaging atmosphere, it was great working there. I'd like to specifically thank: Cliff and Joe for looking out for me over the years. Scott, for always willing to help. DougS, DougR, Fred, Alan, Wade for all the feedback and help getting the research done. DougS for setting up Ducati for me, without her I would have never gotten my thesis done. Tom and Nancy, you really made working at LL a fun and memorable experience, I'll never forget our heated discussions.

I'd also like to thank the administrative assistants at Group 62 and DSPG: Eric, Kathryn, Nancy, Shannon, and Anna. Thank you for your patience and for always helping me in the last minute situations.

Najim, thank you for your friendship, feedback on my research and for providing me with the TV system.

I would have not survived MIT, had I not been surrounded by the greatest of friends. Amer, you encouraged me to start this adventure and have always been there over the years. Al, we've been friends from day one, we've been through so much together and going through it with you was an honor and a pleasure. Demba, thanks for always looking out, even when it's an hour before a submission deadline! Ali, thank you for your energy and optimism. PatrickZ, thanks for always reminding me there is more to life than just work. Nadim and PatrickM for all the great adventures, and PatrickM thanks for all the help with Tigger! Akram and Victor, it was great sharing 24B with you guys, we sure had some good times. Viktorija, thank you for being a great friend, always there always encouraging. Dana, thank you for your love, tenderness, and patience you've really brightened up my last years here, you always know how to make me smile.

Randa and Nawfal, thank you for always checking up on us and always welcoming us in your home.

I'd also like to thank my home-town friends, George, Pierre, Nayla, Woody, Loucy, and Yousef thank you for always welcoming me back and including me when I visit. Yousef, it's also been great spending all this time with you in Cambridge.

I now would like to thank my family: I'd first like to thank the aunts and uncles, nieces and nephews (the older ones and the new additions!). Aunts Salwa and Najwa, thanks for taking care of us and always welcoming us with love and support. Mamy Alieh, thank you for your unconditional love. Teta and Jiddo, thank you for your love, wisdom and support. Aunt Linda, thank you for looking over and praying for us all, and for your endless love. Samer, you've always supported me and encouraged me and always looked out for Hanan and I, thank you and I love you. Hanan, you made Cambridge feel like a home with your warmth and love, I'm grateful we had this opportunity to live together and support each other over these years, I love you.

And most importantly I'd like to thank my parents. Mamy, you've always been there for me with unconditional love, encouragement, support and warmth, I love you and could not have done it without you. Papy, thank you for your love, wisdom, mentorship, and encouragement over the years, I love you and could not have done it without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis explores the problem of training one-versus-all multi-class classifiers when limited target instances are available. We propose graph and subspace methods that leverage auxiliary data (class-labeled non-target or unlabeled instances) to mitigate the adverse effects of limited target instances for training.

The aim of one-versus-all classification is to separate instances of one class from those of all others. While instances may belong to any of a multitude of classes, only a select subset is of interest. We refer to these select classes as targets, and for each, train a one-versus-all classifier to separate instances into target or non-target. An important aspect of training accurate classifiers is the availability of a large number of instances from the target and non-target classes, as demonstrated by the example shown in Figure 1-1. The figure shows three classifiers trained to preform one-versus-all classification. The classifiers, represented by the solid and dashed lines, are trained to distinguish between the $+s$ and $-s$, target and non-target instances respectively, in R^2. Once the classifiers are trained, slope and intersect of the lines are set, an instance that lies above the lines is classified as a target and below as a non-target. These decision boundaries vary according to how many instances are available for training: The “1 train” decision boundary assumes only the circled target instance is available along with all the non-targets, “3 train” extends the training data by including the squared target instances, and “All train” uses all the target instances. It is worth noting that though “1 train” separates the circled target from the non-targets, it fails to properly classify 7 of the $+$ instances, while “3 train” erroneously classifies 3 $+$ instances. Thus, the accuracy of the classifier is adversely affected by the limited availability of target instances for training. This is of concern since the collection of sufficient target training data for accurate classification is sometimes prohibitively costly or simply not possible. In such cases, a large volume of auxiliary data, which can be exploited to mitigate the effects of this deficiency, may be available or cheap to collect.

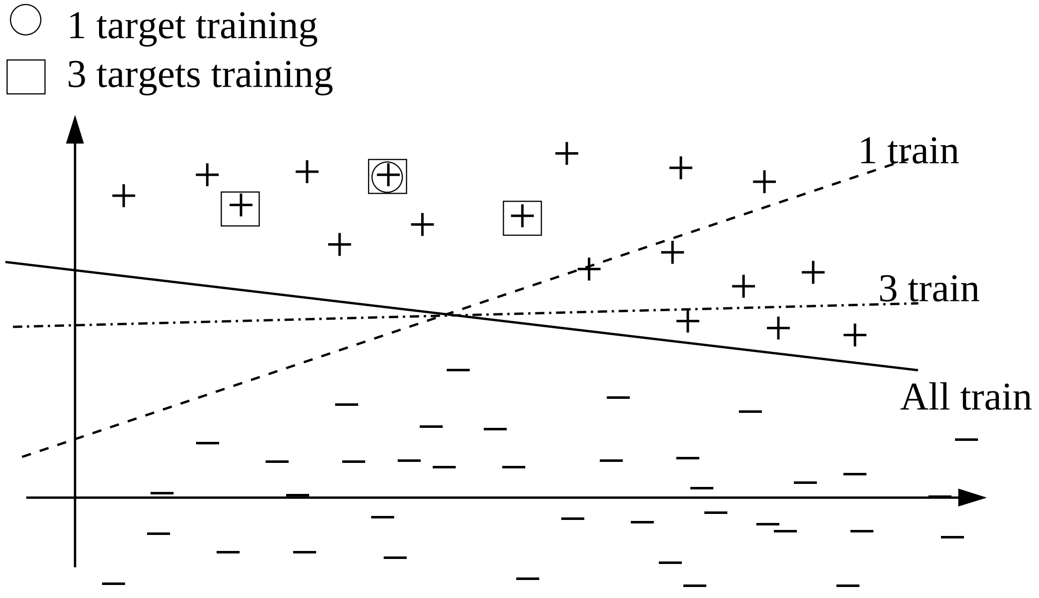

Figure 1-1: Effect of limited target data.

The problem of limited target data arises in several tasks, including face recognition, audio mining, author recognition of documents and speaker verification. In face recognition, the instances are face images. Several target instances may be provided to build a classifier, but these do not span the full set of lighting conditions, backgrounds and capture angles needed to fully specify the variability of the target. Auxiliary data, such as person-labeled face images under different capture conditions as well as a large collection of unlabeled instances, may be used as additional training instances for the classifier.

In speaker verification, instances are recordings of speech and targets are particular speakers of interest. Only a small number of target speaker recordings may be available, as in the core tasks in the NIST speaker verification evaluations [1], which provide only one target recording to train the classifier. In this case, a large amount of auxiliary instances, some labeled, are available for use in training. These are chosen to contain a diverse set of speakers and recording conditions and provide information about how recordings differ across speakers, as well as the variability within the same speaker's recordings.

In these scenarios, auxiliary data can be used for better modeling and representation of the target, as an impostor set in discriminant classifiers, for variability compensation and for inducing graph-relational features:

- At the modeling level, the limited target data does not allow for training a rich representative model of the target. However, auxiliary data may be used to train a generalized model with a large number of parameters that captures the aggregate behavior of the different classes. The parameters of the generalized model can then be fit to the available target data to provide a target model [2].

- The parameters of the generalized model can also be fit to individual instances, and the space of adapted parameters used as a vector space in which to represent the instances [3].

- If the auxiliary data is known to not contain any target data, then it can be used as an impostor data-set to define the decision boundary in a discriminative classifier, such as a support vector machine [3, 4, 5].

- Class-labeled instances can be used to estimate subspaces that capture the within and between class variability. Once estimated they can be used to improve modeling, or incorporated into the classifier [6, 7, 8].

- Assuming a comparison module that measures similarity between two instances is provided, one can perform thousands of comparisons between the target and auxiliary instances to form a similarity matrix. The matrix can then be summarized in a relational graph where nodes represent instances and edges exist between two similar instances, where similarity is defined by the summarization method. The relational graph has been used for manifold discovery and embedding [9], and in semi-supervised algorithms to exploit available unlabeled data for training [10].

- Once a classifier has been trained, it can be scored against a set of impostor instances and the mean and standard deviation of these scores can be used to calibrate the classifier [11, 12, 13].

This thesis focuses particularly on machine learning techniques that leverage auxiliary data to improve classication when limited labeled target data is available. The next section gives an overview of the thesis without considering specific applications. The rest of the thesis, however, focuses on the speaker verification problem and discusses and evaluates the proposed methods within that framework.

Chapter 2 will describe the speaker verification problem, present an overview of the literature, and present the thesis contributions. Chapters 3-9 will each present a contribution in detail. Appendix A briefly describes support vector machines, Gaussian mixture models and relevant adaptation techniques.

1.1 Thesis Overview

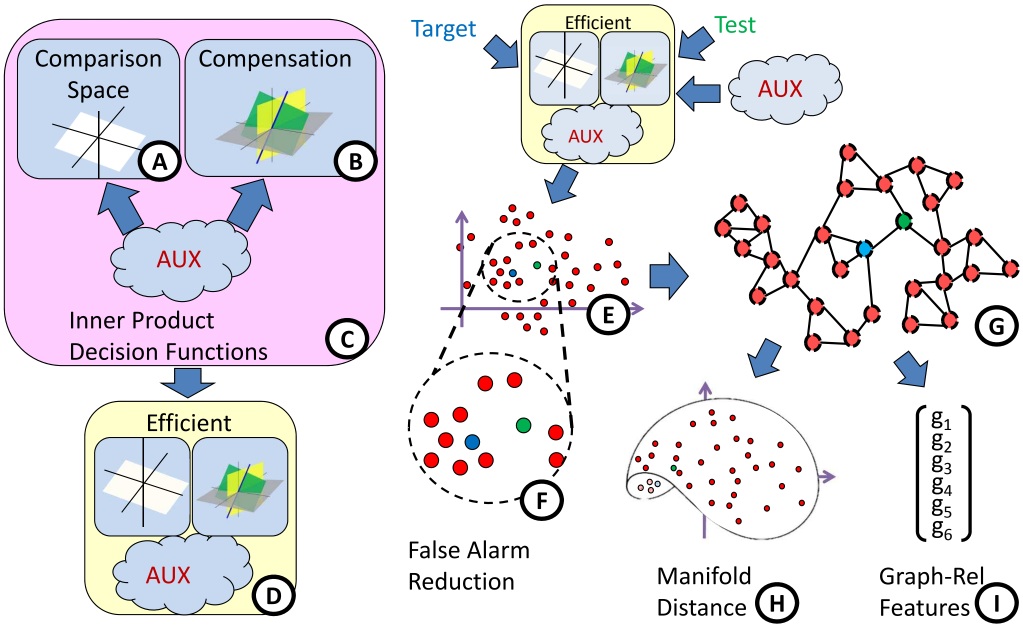

Figure 1-2: High-level overview of the thesis

Figure 1-2 presents a high-level overview of the different ways this thesis leverages auxiliary data. First, we use the auxiliary data to define a vector space where vectors represent instances, as well as the corresponding metric used to compare them, as shown in the top left corner (A) of the figure. The auxiliary data is then used identify nuisance subspaces in the inner-product space, and we propose a technique to compensate for the nuisance (B). Next, we propose a linear framework that combines comparison and compensation (C), which motivates an efficient and accurate way to compare instances (D). Efficient comparisons can then be used to map target, test and auxiliary instances into a single vector space (E). We use this joint representation of the instances to explore the local region around the train and test instances for false-alarm reduction (F). The representation can also be used to perform graph-embedding of the instances (G) which we use in two ways: as a proxy to a manifold on which the data lies (H), and to extract graph-relational features which are useful for classication (I). The remainder of this chapter will describe each part of the thesis in greater detail and will refer to the different parts of Figure 1-2 by their corresponding letters so the reader does not lose sight of the high-level picture. A more detailed discussion of the contributions will be presented in Chapter 2.

1.1.1 GMM-MLLR Inner-Product Space (A)

Comparing and classifying instances is made more difficult by certain variations such as duration and content for speaker verification, image size and resolution in face recognition, and document length in author identification. It is therefore useful to first map instances into an inner-product space that offers invariance to these differences. This can be done by modeling aggregate behavior of features extracted from the instances, and having the space of model parameters be the vector space to which they are mapped. In speaker verification, for example, the choice of features could be local-frequency based [14, 15], while in document classification they could constitute word and N-gram counts [16, 17].

A rich probabilistic model with a large number of parameters, such as a Gaussian mixture model (GMM) (Appendix A.2) with hundreds of mixture components, is needed to properly model the class variability. However, the number of features extracted from each instance may, in general, not be enough to fully fit such a large number of parameters. Rather than train the full model, parameters of a universal model that captures the multi-class aggregate feature distribution can be adapted to fit an instance's features. With a probabilistic model representing each instance, two instances can be compared by comparing their respective models. This has been done, for example, using the Kullback-Leibler (KL) divergence [3].

In Chapter 3, we propose using a GMM with hundreds of mixture components for the universal background model. Maximum likelihood linear regression (MLLR) adaptation is used to adapt the means of the Gaussians, via an affine transformation shared among the mixture components, to fit the features of each instance. Starting with the KL divergence between the adapted models, we apply approximations and algebraic manipulations to derive a new distance metric which defines an inner product space whose dimensions are the parameters of the MLLR affine transform.

1.1.2 Variability-Compensated Support Vector Machines (B)

In classification there are two types of variability: the between-class (good) and the within-class (bad). The good, or signal, captures the between-class variations and enables classification, while the bad, or nuisance, encompasses all other variability that confuses the classifier. Assuming that the instances have been mapped into points in an inner-product space, auxiliary data can be used to estimate subspaces of interest, for example the one that contains the nuisance variability. To utilize these estimates, Chapter 4 proposes variability-compensated support vector machines (VCSVM), which incorporate the subspaces of interest into the SVM formulation, thus combining training the decision function and the variability compensation into one optimization.

1.1.3 Inner Product Decision Functions (C,D)

In this thesis, we propose a particular inner-product space and a specific manner in which to compensate that space (Chapters 3 & 4). There are, however, other linear comparison and compensation techniques in the literature [6, 7, 8, 3]. An unrealized effort is to compare these to one another and to understand how the interaction between the choice of inner-product space and compensation affects classification performance. We therefore propose in Chapter 5 the inner product decision function (IPDF) framework that combines the two and encompasses the majority of the techniques in the literature. This unified framework allows for direct contrasting between these compensated inner-product spaces, leading to a better understanding of what crucial components are needed to represent instances well. We then use this understanding to propose a new efficient metric and compensation that match the existing in accuracy with reduced computational cost.

1.1.4 Leveraging Auxiliary Data with Fast Comparison Functions (E-I)

The efficient compensated inner product resultant from the IPDF framework can be used as a class-similarity score between two instances of interest. The efficiency further enables us to also compute a similarity matrix whose entries are pairwise comparisons between the two instances of interest and auxiliary instances. This matrix captures the interaction between instances and contains information that may be leveraged to obtain a more accurate class-similarity score. In this section, we present the different ways we use this interaction to improve classification.

In Chapter 6, we propose algorithms that use the interaction of the pair of instances under consideration with those most similar to them in the auxiliary set to reduce false alarms.

In Chapter 7, we explore how the similarity matrix can be summarized, by keeping only the entries corresponding to strongest similarity, and transformed into a relational graph. Each instance is represented by a node in the graph and nodes are connected by edges if they are deemed similar enough. These relational graphs are then used to explore whether the data lies on a low-dimensional manifold in the space, and distances along the manifold between instances are then used for classification. The relational graph can also be used for visualization and exploring large data-sets, as shown in Chapter 9.

In Chapter 8, we suggest that the graph interaction between the pair of interest and the auxiliary data can be used for classification. We do this by extracting from the graph several relational features, including the graph distance used in Chapter 7 and local neighborhood ones similar to those used to identify false-alarms in Chapter 6. We then use these graph-relational features in a classifier trained to determine whether or not a pair of instances belongs to the same class.

Chapter 2

Speaker Verification

The goal of text-independent speaker verification is to classify whether a given recording was spoken by the target speaker, regardless of the spoken words. This is typically approached as a one-versus-all classification problem where a binary classifier is trained to distinguish recordings of the target speaker from those of all others. This chapter begins by presenting how speaker verification algorithms are evaluated, followed by a brief overview of the relevant literature and finally the thesis contributions to the field.

2.1 Algorithm Evaluation

Speaker verification is an active field with a well established community and standardized evaluation opportunities provided by the NIST speaker recognition evaluations [1]. This section describes the NIST evaluation scope, data-sets and metrics.

2.1.1 The NIST Speaker Recognition Evaluation

The National Institute of Standards and Technology NIST evaluates the state of the art of speaker verification, typically every other year, in the Speaker Recognition Evaluation (SRE) [1]. The most recent three evaluations occurred in 2006 [18], 2008 [19], and 2010 [20] with each containing multiple tasks to evaluate system performance and robustness to variability such as: the length of the recordings, number of target recordings, language spoken and channel. Each task contains thousands of trials and each is considered independently of the others. A trial consists of target recordings to train the classifier and one recording to test. True or target trials are ones where the test recording corresponds to the target speaker, the remaining are false or non-target trials. The goal of speaker verification systems is, therefore, to label each trial correctly and they are evaluated based on the number of misses and false-alarms. A miss is when the classifier erroneously labels a true trial as false, and a false alarm is when it labels a false trial as true.

This thesis focuses on the core task which consists of male and female trials containing a single training and single test recording, around 5 minutes long, of telephony (cellular and land-line) speech. The 2006 and 2008 evaluations contain recordings in multiple languages and allow for a language mismatch between the training and testing recordings of a trial, while the 2010 contains only English speech.

Each of the thesis contributions is evaluated using a subset of the evaluations, which we will specify before presenting each set of experimental results.

2.1.2 Algorithm Evaluation Metrics

A standard evaluation tool in the speaker verification literature is the detection error trade-off (DET) curve. The curve fully characterizes the trade-off between the probability of false-alarm and probability of miss of the system as the decision threshold is varied. Figure 2-1 shows an example of a DET plot which overlays the performance of three systems. System A clearly outperforms systems B and C since the DET curve for A lies below the others over the full operating range. The curves of B and C, however, intersect, meaning that the better choice of system depends on the cost trade-off between false alarms and misses, with system B being preferred if false-alarms were more costly.

While the DET curve provides a broad overview of system performance, systems can also be evaluated at particular points on the DET curve. The two that are typically reported in the literature, are the equal error (EER) and minimum detection cost function (minDCF) points: the EER point is the location on the DET curve where the probability of miss is equal to the probability of false alarm, and the minDCF point is the location where the detection cost function (DCF) is minimized. The DCF is a function of the classification threshold and takes into consideration the prior on the target speaker recordings as well as the cost of a false alarm and a miss:

$$DCF(thld) = C_{Miss} P_{Miss|Target} P_{Target} + C_{FalseAlarm} P_{FalseAlarm|NonTarget} (1 - P_{Target}). \tag{2.1}$$

The choice of the costs, C_{Miss} and $C_{FalseAlarm}$, and target prior, P_{Target}, as set by the NIST evaluations are presented in Table 2.1 :

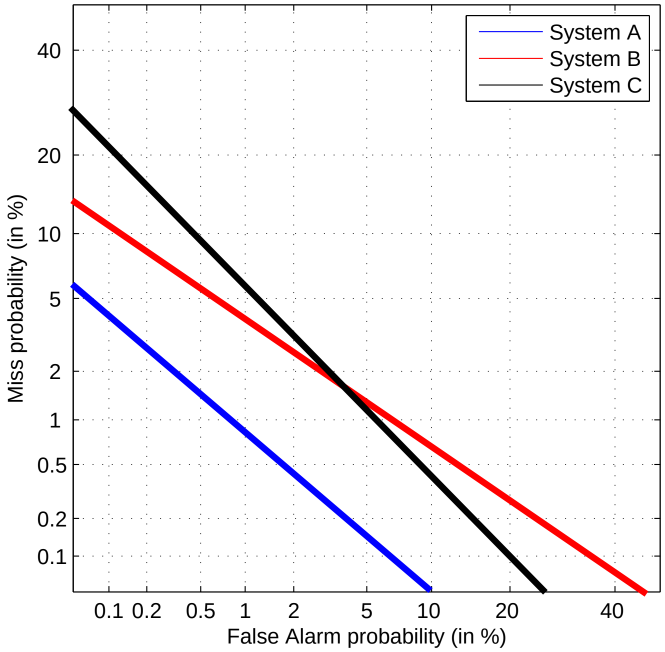

Figure 2-1: Example DET plot comparing three systems.

Table 2.1: Decision Cost Function Parameters

NIST SRE Year	Cost of Miss	Cost of False-Alarm	Probability of Target
2006	10	1	0.01
2008	10	1	0.01
2010	1	1	0.001

2.2 Literature Overview

This section presents some of the more recent and popular speaker verification techniques as well as those that are relevant to the thesis. The goal is to provide the reader with a broad overview of how the speaker verification problem is typically approached, and to set the stage for presenting the thesis contributions.

2.2.1 Features

As in any classification problem, the first step is to extract from each recording $\mathbf{R}$ features, $\{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N\}$, that capture individual speaker identity thus enabling classification. Short-time frequency based features, such as PLP [14] and mel-cepstral coefficient [15] features, have proven to contain this information. The most widely used in the recent literature, and those used in this thesis are the mel-cepstral features. These are extracted by sliding a short, typically 25ms, window across the speech recording, computing mel-cepstral coefficients for each window and complementing these with deltas and double deltas [21] extracted from consecutive windows. Typically RASTA [22], feature warping [23] and/or mean-variance normalization are applied to the features to help reduce channel effects. Figure 2-2 shows a block diagram of the feature extraction process.

Each chapter of this thesis uses a slight variation on these features, and we will present the specific configuration before providing any experimental results.

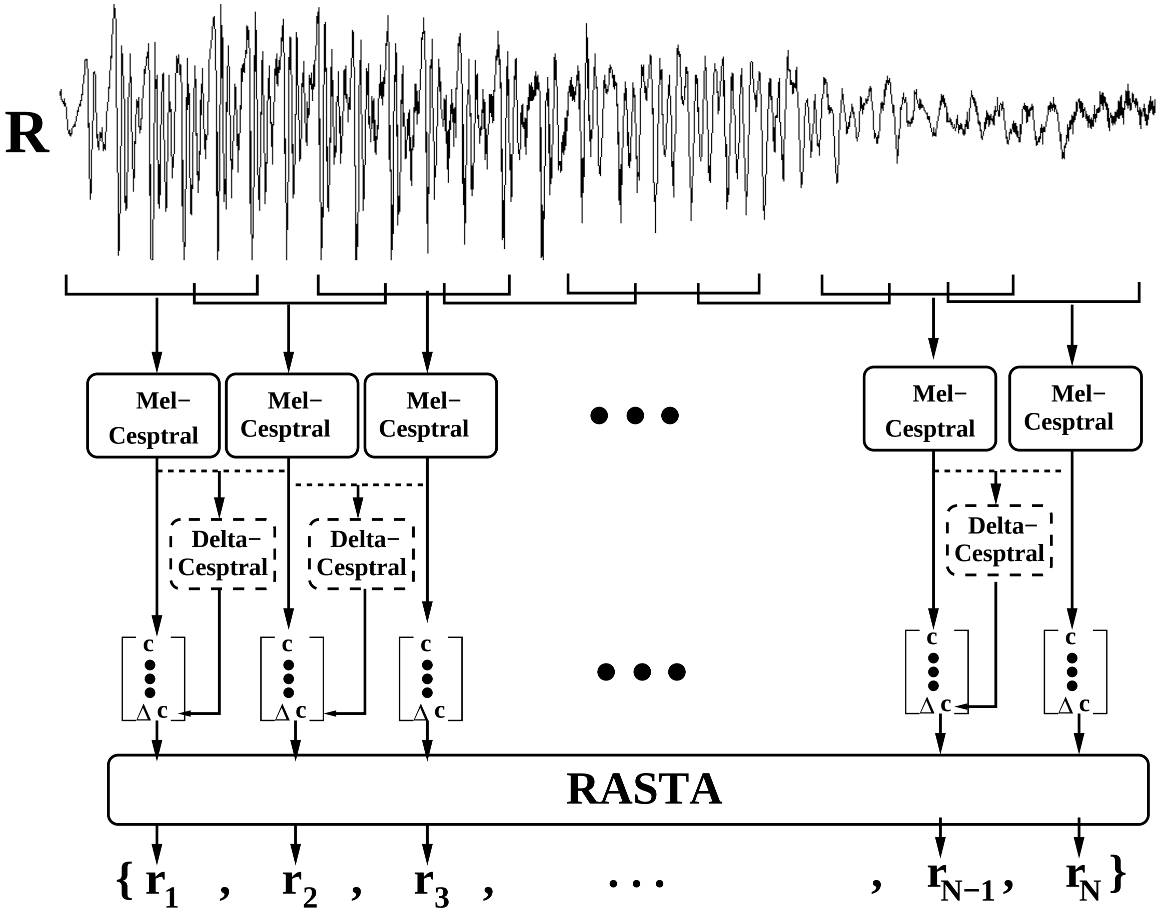

Figure 2-2: Sketch of the feature extraction

2.2.2 MAP Adapted GMM UBM

For each target speaker, a generative model, $g_{TGT}(\mathbf{r})$, can be trained to model the features extracted from the target training recordings. A universal background model (UBM), $g_{UBM}(\mathbf{r})$, can also be trained to model the features extracted from an auxiliary set of recordings representing the general population of speakers, thus resulting in a speaker-independent model. Given features $\{\mathbf{r}_1^{TST}, \mathbf{r}_2^{TST}, ..., \mathbf{r}_{N^{TST}}^{TST}\}$ extracted from the test recording $\mathbf{R}_{TST}$, the binary classification becomes a log-likelihood-ratio test that classifies the recording as target if the log-likelihood of the target is larger than the log-likelihood of the UBM given the test recording:

$$\log p(\{\mathbf{r}_1^{TST}, ..., \mathbf{r}_{N^{TST}}^{TST}\}|TGT) - \log p(\{\mathbf{r}_1^{TST}, ..., \mathbf{r}_{N^{TST}}^{TST}\}|UBM) > \tau, \tag{2.2}$$

where $\log p(\{\mathbf{r}_1^{TST}, ..., \mathbf{r}_{N^{TST}}^{TST}\}|.)$ is the log-likelihood of the model given the test features and τ is a threshold set based on the operating point of the DET curve.

In [2] the generative models used were Gaussian mixture models (GMMs) (Appendix A.2) with diagonal covariances and $M = 2048$ mixture components:

$$g_{UBM}(\mathbf{r}) = \sum_{i=1}^{M} \lambda_{UBM,i} \mathcal{N}(\mathbf{r}; \mathbf{m}_{UBM,i}, \Sigma_{UBM,i}). \tag{2.3}$$

Maximum likelihood (ML) estimation of the model parameters was used to train the UBM, via expectation maximization (EM), to fit the auxiliary data. The target model is trained by adapting, via maximum a posteriori (MAP) adaptation (Appendix A.3), the means of the UBM to fit the target data.

2.2.3 MAP Gaussian Supervector (GSV) Kernel

The binary classification aspect of speaker verification makes the problem especially suited for support vector machines (SVMs), refer to Appendix A.1. The challenge, however, is in defining a vector space and devising a kernel (Appendix A.1) to compare between two recordings, possibly of different lengths. The Gaussian supervector (GSV) kernel, introduced in [24], is one particular choice that has been widely used in the literature and is based on comparing GMMs that model each of the recordings:

As in the previous section, the kernel begins with a diagonal-covariance GMM for the speaker-independent UBM (2.3). The means of the UBM are MAP adapted to each recording. Thus for recordings $\mathbf{R}_\alpha$ & $\mathbf{R}_\beta$, this results in new GMMs that represent them:

$$g_\alpha(\mathbf{r}) = \sum_{i=1}^{M} \lambda_{UBM,i}\mathcal{N}(\mathbf{r}; \mathbf{m}_{\alpha,i}, \Sigma_{UBM,i}) \quad \& \quad g_\beta(\mathbf{r}) = \sum_{i=1}^{M} \lambda_{UBM,i}\mathcal{N}(\mathbf{r}; \mathbf{m}_{\beta,i}, \Sigma_{UBM,i}). \tag{2.4}$$

Since only the means differ between the two models it is reasonable to expect that the means would contain the information needed for classification. Thus a good choice for a vector space in which to represent recordings is that of the stacked means. Figure 2-3 sketches this process.

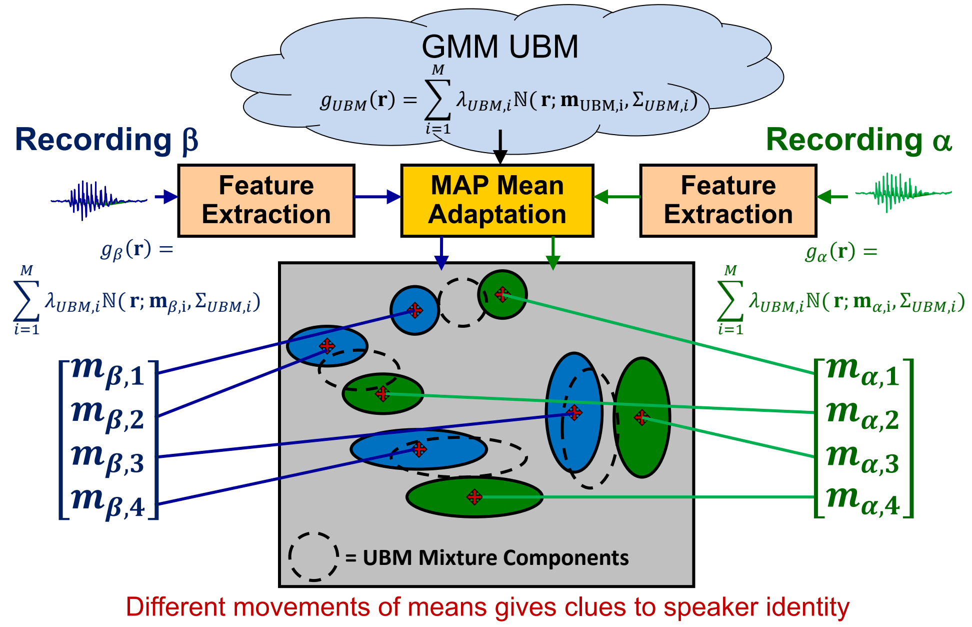
GMM UBM
$g_{UBM}(\mathbf{r}) = \sum_{i=1}^{M} \lambda_{UBM,i}\mathbb{N}(\mathbf{r}; \mathbf{m}_{\text{UBM,i}}, \Sigma_{UBM,i})$
Recording β
Recording α
Feature Extraction
MAP Mean Adaptation
Feature Extraction
$g_\beta(\mathbf{r}) = \sum_{i=1}^{M} \lambda_{UBM,i}\mathbb{N}(\mathbf{r}; \mathbf{m}_{\beta,\text{i}}, \Sigma_{UBM,i})$
$g_\alpha(\mathbf{r}) = \sum_{i=1}^{M} \lambda_{UBM,i}\mathbb{N}(\mathbf{r}; \mathbf{m}_{\alpha,\text{i}}, \Sigma_{UBM,i})$
$m_{\beta,1}$ $m_{\beta,2}$ $m_{\beta,3}$ $m_{\beta,4}$
$m_{\alpha,1}$ $m_{\alpha,2}$ $m_{\alpha,3}$ $m_{\alpha,4}$
= UBM Mixture Components
Different movements of means gives clues to speaker identity

Figure 2-3: Representing recordings by MAP adapted GMMs.

To formalize this choice of vector space and to define a metric (inner product) on it, one can begin by considering that each recording is represented by its own probability model. Thus, comparing recordings can be done by comparing their corresponding models. A good measure of the difference between two probability distributions is the KL divergence:

$$D(g_\alpha \| g_\beta) = \int_{R^n} g_\alpha(\mathbf{r}) \log\left(\frac{g_\alpha(\mathbf{r})}{g_\beta(\mathbf{r})}\right) d\mathbf{r}. \tag{2.5}$$

The KL divergence measures how much two probability densities differ, unfortunately it does not directly induce an inner product that satisfies the Mercer conditions [25] required for it to be used as an SVM kernel.

Instead of using the KL divergence directly, the log-sum inequality can be used to approximate it by its upper bound [24],

$$D(g_\alpha \| g_\beta) \leq \frac{1}{2} \sum_{i=1}^{M} \lambda_{UBM,i} (\mathbf{m}_{\alpha,i} - \mathbf{m}_{\beta,i})^T \mathbf{\Sigma}_{UBM,i}^{-1} (\mathbf{m}_{\alpha,i} - \mathbf{m}_{\beta,i}). \tag{2.6}$$

Note that this approximation is removing any inter-mixture dependency, i.e. it's a weighted (by the mixture weights) sum of distances between the i^{th} Gaussian component in $g_\alpha, \mathcal{N}(\mathbf{r}; \mathbf{m}_{\alpha,i}, \Sigma_{UBM,i})$, and its corresponding Gaussian in $g_\beta, \mathcal{N}(\mathbf{r}; \mathbf{m}_{\beta,i}, \Sigma_{UBM,i})$. The distance in (2.6) induces an inner product, which results in the Gaussian supervector (GSV) kernel between two recordings [24]:

$$K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) \quad = \quad \sum_{i=1}^{M} \lambda_{UBM,i} \mathbf{m}_{\alpha,i}^T \mathbf{\Sigma}_{UBM,i}^{-1} \mathbf{m}_{\beta,i} \tag{2.7}$$

Defining GMM supervectors to be vectors formed by stacking the means of the GMM,

$$\mathbf{m}_\alpha = \begin{bmatrix} \mathbf{m}_{\alpha,1} \\ \mathbf{m}_{\alpha,2} \\ ... \\ \mathbf{m}_{\alpha,M} \end{bmatrix} \qquad and \qquad \mathbf{m}_\beta = \begin{bmatrix} \mathbf{m}_{\beta,1} \\ \mathbf{m}_{\beta,2} \\ ... \\ \mathbf{m}_{\beta,M} \end{bmatrix}, \tag{2.8}$$

allows us to write the GSV kernel in terms of the supervectors:

$$K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \quad \mathbf{m}_\alpha^T \mathbf{\Delta}_{UBM} \mathbf{m}_\beta, \tag{2.9}$$

$$\mathbf{\Delta}_{UBM} = \begin{bmatrix} \lambda_{UBM,1} \mathbf{\Sigma}_{UBM,1}^{-1} & 0 & ... & 0 \\ 0 & \lambda_{UBM,2} \mathbf{\Sigma}_{UBM,2}^{-1} & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & \lambda_{UBM,M} \mathbf{\Sigma}_{UBM,M}^{-1} \end{bmatrix}. \tag{2.10}$$

where $\mathbf{\Delta}_{UBM} = diag(\lambda_{UBM,1} \mathbf{\Sigma}_{UBM,1}^{-1}, ..., \lambda_{UBM,M} \mathbf{\Sigma}_{UBM,M}^{-1})$ is a diagonal matrix, since each of the covariance matrices $\mathbf{\Sigma}_{UBM,i}$ are diagonal. Normalizing the supervectors by the square root of $\mathbf{\Delta}_{UBM}$ results in:

$$K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \mathbf{m}_\alpha^T \mathbf{\Delta}_{UBM}^{1/2} \mathbf{\Delta}_{UBM}^{1/2} \mathbf{m}_\beta = \tilde{\mathbf{m}}_\alpha^T \tilde{\mathbf{m}}_\beta, \tag{2.11}$$

where $\tilde{\mathbf{m}}$ represent the normalized supervectors. The space spanned by the supervectors and the normalized supervectors will be referred to as the GSV space and the normalized GSV space respectively.

SVM training then finds the hyperplane that maximally separates the normalized supervectors $\tilde{\mathbf{m}}_{TGT,t}$ representing the target recordings $\{\mathbf{R}_{TGT,1}, ..., \mathbf{R}_{TGT,T}\}$ from the normalized supervectors $\tilde{\mathbf{m}}_{IMP,i}$ representing the impostor ones $\{\mathbf{R}_{IMP,1}, ..., \mathbf{R}_{IMP,I}\}$ in the normalized GSV space. A test recording $\mathbf{R}_{TST}$ is then classified according to which side of the hyperplane its normalized supervector $\tilde{\mathbf{m}}_{TST}$ falls. The training and testing process is illustrated in Figure 2-4.

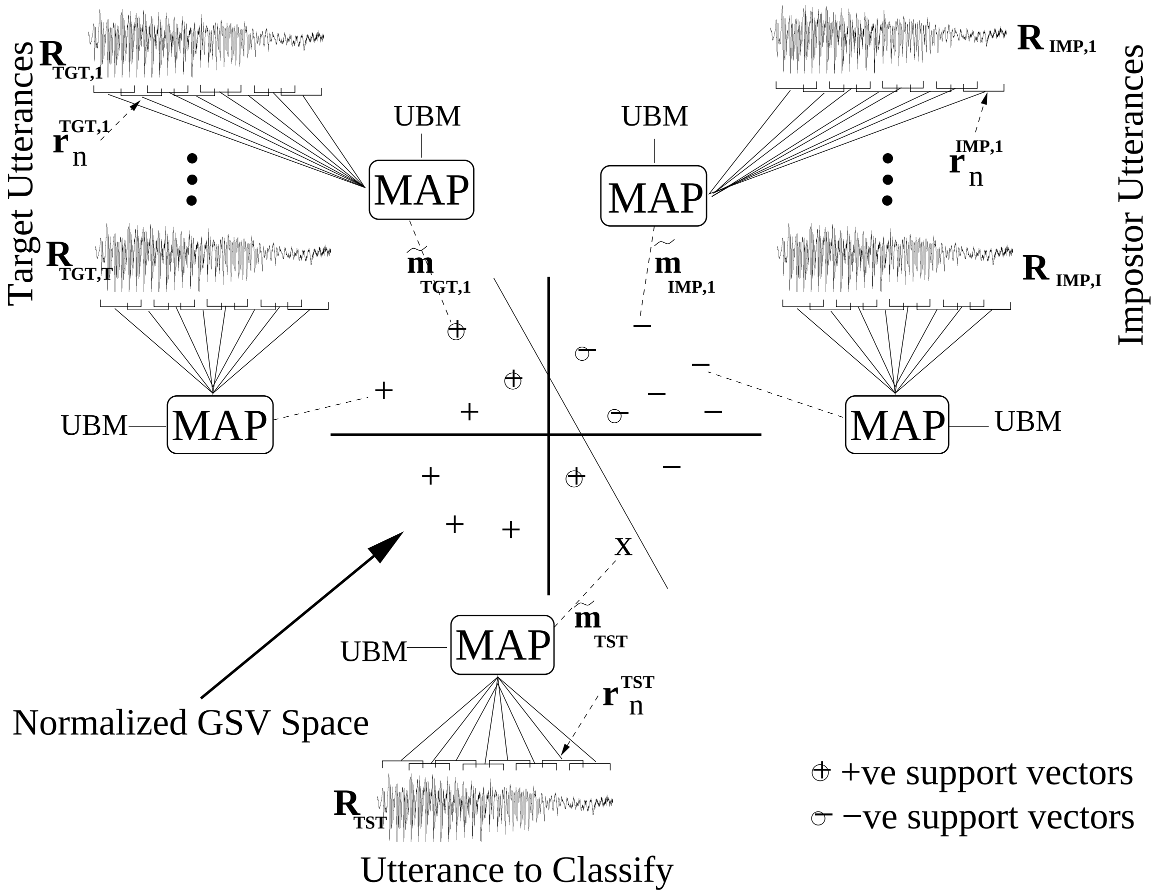

Figure 2-4: Sketch of Using SVMs to Perform Speaker Verification

2.2.4 Nuisance Compensation

The previous section mapped recordings into the GSV space and performed speaker verification there. It is, however, important to note that the GSV space captures most of the variability present in the recording, not just the speaker information. This additional nuisance variability, in the form of channel, session and language differences leads to poor classification. Thus, there is a need to perform nuisance compensation, and here we present two of the most popular approaches in the literature.

Nuisance Attribute Projection (NAP)

Nuisance attribute projection (NAP) [6] assumes there is a relatively small subspace of the SVM space that contains the nuisance variability, and projects the supervectors into the complement of that subspace. Let $\mathbf{U}$ be the matrix whose orthonormal columns form the basis of the nuisance subspace and $\mathbf{m}$ be a recording's supervector in the SVM space, then NAP is applied as follows:

$$\mathbf{m}_{NAP} = (\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{m}. \tag{2.12}$$

The directions of the nuisance subspace are chosen to be the principal components of the within-speaker covariance matrix [6]. An estimate of the covariance can be computed from speaker-labeled supervectors of an auxiliary set containing multiple recordings per speaker. The within-speaker variability serves as a proxy for the nuisance, since if the SVM space contains only speaker variation all recordings of a given speaker should map to the same point. Thus, projecting out these principal directions leads to a reduction in within-speaker and consequently nuisance variability.

Within Class Covariance Normalization (WCCN)

Like NAP, the basis of within class covariance normalization (WCCN) is the estimate of the within-speaker covariance matrix, $\mathbf{W}$, the inverse of which is used to weight the kernel inner product [7]:

$$K_{WCCN}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \mathbf{m}_\alpha \mathbf{W}^{-1} \mathbf{m}_\beta, \tag{2.13}$$

where $\mathbf{m}_\alpha$ and $\mathbf{m}_\beta$ are mappings of the recordings $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$ into the SVM space. Since $\mathbf{W}$ is estimated from the auxiliary set, it may not be of full rank. To overcome this, $\hat{\mathbf{W}} = ((1-\theta)\mathbf{I}+\theta\mathbf{W})$, where $0 \leq \theta < 1$ is a tunable parameter, is used in practice. Another issue with this method is that it requires inverting a matrix that is the size of the SVM space, thus, it wont work well in large spaces.

2.2.5 Joint Factor Analysis (JFA)

Rather than model the recording first and then perform nuisance compensation, joint factor analysis [26] incorporates the expected variability into the modeling step. Specifically, it assumes that the deviation of the mean supervector $\mathbf{m}$ of a particular recording from the UBM mean, $\mathbf{m}_{UBM}$, contains a component that lies in a low-dimensional speaker subspace $\mathbf{V}$, another in a low-dimensional channel subspace $\mathbf{U}$ and a residual not lying in either:

$$\mathbf{m} = \mathbf{m}_{UBM} + \mathbf{V}\mathbf{y} + \mathbf{U}\mathbf{x} + \mathbf{D}\mathbf{z}, \tag{2.14}$$

where $\mathbf{y}$ and $\mathbf{x}$ are the speaker and channel factors respectively, $\mathbf{D}$ is a diagonal matrix, and $\mathbf{Dz}$ represents the residual. The speaker ($\mathbf{V}$) and channel ($\mathbf{U}$) subspaces, and $\mathbf{D}$ are jointly trained using a speaker-labeled auxiliary set.

When training a target model, the parameters $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{z}$ are jointly estimated to best fit the features of the target recording. To obtain the clean target model, that captures only the speaker information present in the recording, one discards the estimated channel contribution, resulting in the target-speaker mean supervector $\hat{\mathbf{m}}$:

$$\hat{\mathbf{m}} = \mathbf{m}_{UBM} + \mathbf{V}\mathbf{y} + \mathbf{D}\mathbf{z}. \tag{2.15}$$

Speaker Factors

Since the low-dimensional speaker space essentially contains the speaker information, it was suggested in [27] that two recordings be compared in that space. This led to the speaker factor kernel, which is an inner product between the speaker factors, normalized to have unit L_2-norm:
Given two recordings $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$ and their mean supervectors

$$\mathbf{m}_\alpha = \mathbf{m}_{UBM} + \mathbf{V}\mathbf{y}_\alpha + \mathbf{U}\mathbf{x}_\alpha + \mathbf{D}\mathbf{z}_\alpha \quad \& \quad \mathbf{m}_\beta = \mathbf{m}_{UBM} + \mathbf{V}\mathbf{y}_\beta + \mathbf{U}\mathbf{x}_\beta + \mathbf{D}\mathbf{z}_\beta, \tag{2.16}$$

the resultant speaker factor kernel is

$$K_{SF}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \frac{\mathbf{y}_\alpha^T \mathbf{y}_\beta}{\sqrt{\mathbf{y}_\alpha^T \mathbf{y}_\alpha}\sqrt{\mathbf{y}_\beta^T \mathbf{y}_\beta}}. \tag{2.17}$$

2.2.6 Total Variability (TV)

Similar to the JFA system, the total variability (TV) system [28] considers the variability in a recording $\mathbf{R}_\alpha$ to be restricted to a low-dimensional subspace. However, rather than proposing a speaker and channel subspace, the TV system proposes a single subspace that captures all the variability, called the TV space. The recording's mean supervector $\mathbf{m}_\alpha$ can then be represented as

$$\mathbf{m}_\alpha = \mathbf{m}_{UBM} + \mathbf{T}\mathbf{t}_\alpha, \tag{2.18}$$

where $\mathbf{m}_{UBM}$ is the UBM mean supervector, $\mathbf{T}$ is the matrix defining the TV subspace, and $\mathbf{t}_\alpha$ is the corresponding factor of the recording $\mathbf{R}_\alpha$.

In the TV space, linear scoring can be performed between the target and test recordings to evaluate whether both were spoken by the target:
The scoring function $s(\mathbf{R}_{TGT}, \mathbf{R}_{TST})$ is computed as a weighted inner-product where the weighting effectively performs channel compensation

$$s(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) = \frac{\mathbf{t}_{TGT}^T A W^{-1} A^T \mathbf{t}_{TST}}{\sqrt{\mathbf{t}_{TGT}^T A W^{-1} A^T \mathbf{t}_{TGT}}\sqrt{\mathbf{t}_{TST}^T A W^{-1} A^T \mathbf{t}_{TST}}}.$$

A corresponds to a linear discriminant analysis (LDA) [29] projection matrix, trained to project into a space that captures inter-speaker variability while avoiding within-speaker variability, and W is the within-speaker covariance matrix computed in the LDA space. Both A and W are estimated using a speaker-labeled auxiliary set of recordings.

2.2.7 Symmetric Score Normalization (SNorm)

It is common for speaker verification systems to be followed by a score normalization technique. The goal being to reduce within trial variability to obtain improved performance, better calibration, and more reliable threshold setting. There are several score normalizing techniques: TNorm [12], ZNorm [13], ATNorm [30], SNorm [11]. Here we present symmetric score normalization (SNorm) as an example.

For every score $s(\mathbf{R}_\alpha, \mathbf{R}_\beta)$ between two recordings, the corresponding SNorm score $\hat{s}(\mathbf{R}_\alpha, \mathbf{R}_\beta)$ is

$$\hat{s}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \frac{s(\mathbf{R}_\alpha, \mathbf{R}_\beta) - \boldsymbol{\mu}_\alpha}{\sigma_\alpha} + \frac{s(\mathbf{R}_\alpha, \mathbf{R}_\beta) - \boldsymbol{\mu}_\beta}{\sigma_\beta}, \tag{2.19}$$

where $\boldsymbol{\mu}_\alpha$ and σ_α are the mean and standard deviation of the scores of $\mathbf{R}_\alpha$ scored against an impostor list, similarly for $\mathbf{R}_\beta$, $\boldsymbol{\mu}_\beta$ and σ_β.

2.3 Thesis Contributions

We now present the thesis contributions to the field of speaker verification. The letters (A-I) represent the different parts of the high-level overview shown in Figure 1-2.

2.3.1 GMM-MLLR Kernel (A)

In [5] & [4], alternatives to the MAP GSV kernel are proposed for SVM speaker verification. The former uses maximum likelihood linear regression (MLLR) to adapt the means of the GMM emission probabilities of HMMs representing phonetic level acoustic models of a speaker-independent large vocabulary continuous-speech recognizer (LVCSR), and the latter uses constrained MLLR (CMMLR) to adapt the means and covariances of a GMM UBM. Both MLLR and CMLLR, constrain the adaptation to affine transformations of the universal model's parameters, the transformations being typically shared across all or subsets of the GMMs in the LVCSR. The kernels proposed by these systems are inner products between vector forms of the affine transformations' parameters, which is a reasonable choice since these parameters capture the required discriminating information. Though reasonable, no theoretical motivation is provided, thus leading both kernels to rely on ad-hoc normalization of the transform vectors in the kernels: [5] uses rank while [4] uses min-max normalization.

In Chapter 3, we follow a similar approach to that in section 2.2.3 to derive a theoretically motivated kernel between two GMMs adapted from a UBM using MLLR mean adaption. As with the other MLLR kernels the resultant is based on an inner-product between the affine transform vectors. Our approach, however, provides a specific way to normalize the vectors that is based on the covariance and mixture weight parameters of the UBM. We compare our motivated weighting against ad-hoc ones and show a clear advantage. Even though this thesis derived the MLLR kernel using a GMM UBM, it can be directly extended to the case where the UBM is a speaker independent LVCSR.

2.3.2 Variability-Compensated Support Vector Machines (VCSVM) (B)

Even though NAP and WCCN, Section 2.2.4, are both based on some estimate of the nuisance space computed using auxiliary data, they present two different approaches to nuisance compensation for SVM classification. NAP was developed under the assumption that the nuisance subspace is a relatively small one and can be discarded. WCCN on the other hand does not make that assumption and instead re-weights the full SVM space based on the nuisance estimate.

Since the end goal of nuisance compensation is improved SVM classification, we chose, in Chapter 4, to combine the compensation and classification into a single training algorithm. We do this by incorporating the nuisance estimate into the SVM optimization. This approach leads to a framework that includes NAP and WCCN as specific instances yet allows for tuning to achieve better compensation. Our method also extends WCCN to work in large dimensional spaces efficiently.

2.3.3 Inner Product Decision Functions (IPDF) (C,D)

The MAP GSV, Section 2.2.3, kernel is one of several speaker comparison techniques that results in an inner product between mean supervectors; other examples include the Fisher kernel [31], GLDS kernel [32], and a linearized version of the JFA scoring [33]. The speaker verification literature also contains several nuisance compensation techniques that result in a linear transformation of the mean supervectors, e.g. NAP results in an orthogonal projection and JFA could be reformulated as an oblique projection. This diversity in linear comparisons and compensations is due to the different approaches to the problem. Even though the resultant algorithms can all be formulated as a combination of linear comparison and compensation, they vary significantly in terms of verification performance and implementation cost.

To better understand the discrepancy in performance between systems, we propose in Chapter 5, a framework that encompasses all these linear comparison and compensation techniques. Placing the competing algorithms in this framework allowed us to compare them side by side and understand which of their sub-components were crucial for good speaker verification performance. The framework was also useful beyond just comparing the existing systems, as it motivated new comparison and compensation combinations that match state of the art performance at a significantly reduced implementation cost.

2.3.4 Relational Algorithms and Features (E-I)

The availability of low-cost comparison functions, such as the TV system [28] or those motivated by the IPDF framework, allows one to leverage auxiliary data-sets in speaker verification by supplementing the comparison score between the trial recordings with their similarity score with the auxiliary recordings. In this thesis we propose several ways to do this:

Local Neighborhood Methods for Reduced False Alarms (F)

The first set of techniques were motivated by the update to the detection cost function (DCF) in the NIST SRE 2010 [20], as is seen in Table 2.1. These changes in the costs move the minDCF point into the low false-alarm region of the DET curve, for which traditional comparison functions and score normalization techniques are not optimized.

In Chapter 6, we propose methods that specifically tackle the low false-alarm region by examining the interaction of the trial recordings with their immediate neighbors, auxiliary recordings that are most similar to those of the trial. This local interaction is then used to identify suspect false-alarm trials, which tend to be trials whose pair of recordings match auxiliary recordings better than they match one another. Once identified, a penalty function based on the degree of neighborhood similarity is used to penalize the trial by reducing its match score. The resultant proposed methods take on the form of adaptive score normalization. Our experiments show that the success of these algorithms hinges on having a good match between the testing data, which contains the trial recordings, and the auxiliary data, with significant improvement in the low false-alarm region when they are well matched.

Graph Embedding to Identify and Leverage Structure (G,H)

The relational interaction between the trial and auxiliary recordings can be extended beyond the local neighborhood to uncover global structure such as manifolds. This can be done by embedding the trial and background recordings as nodes in a graph and edges between the nodes capture local similarity. Though the graph relies on local similarity, it captures global structure in the data and can be used as a proxy to the manifold on which the speech recordings lie. The shortest path along the graph edges between two nodes is, therefore, an approximation to the shortest path along the manifold between the corresponding recordings.

In Chapter 7, we show how graph embedding of speech recordings can be done, and use it to form relational graphs of the test and auxiliary data. We then use the resultant graphs to empirically show that there indeed does exist an underlying low-dimensional manifold that captures the variability in the data. We also propose using the shortest path distance along the graph between two nodes to perform speaker verification. We argue that this provides a more accurate representation of the true similarity between recordings than the score provided by the direct comparison function used to build the graph. We then present experimental results that show the efficacy of this graph distance for speaker verification.

Graph-Relational Features for Speaker Verification (I)

The relational graph captures more than just the local information, which we used for false-alarm reduction, and the shortest path distance, which we used as a similarity score. In Chapter 8, we attempt to extract features that capture additional relational information and use it for speaker verification. These graph-relational features are motivated by the link-prediction problem, which predicts whether a link should exist between two nodes in a graph based on their interaction with the remaining graph nodes. We then use these features in a classifier to discriminate between true and false trials. Our experimental results show that the relational graphs capture information relevant to speaker verification, as evidenced by significantly improved verification with the graph-relational features.

Graph Embedding for Visualization (G)

Another use of graph embedding is for visualization of large data-sets. The visualization can be used, for example: to explore the data-sets and uncover structure, to provide an intuitive sense of system performance, to better understand errors in the system, and to identify errors in provided labels. In Chapter 9, we present two case-studies as examples that highlight the utility of these visualizations for data exploration.

Chapter 3

GMM-MLLR Kernel

Many classification techniques, specifically the ones discussed in this thesis, require that instances, whether they be images, documents, or recordings, be represented as vectors in an inner-product space. The goal of the vector space is to provide invariance and the inner product to provide a metric that contains inter-class similarity. In speaker verification, the invariance needed is to the duration of the recordings, the underlying words spoken, and slowly varying linear time invariant channels. Once an inner-product metric is defined on the space it can then be used as a kernel between the vector representations (alternatively called feature expansions) of the instances for support vector machine (SVM) classification (Appendix A.1).

Using SVMs with vector representations of instances has proven to be a popular and powerful tool in text-independent speaker verification [24, 5, 4, 31, 32]. A common method is to derive the vector space and kernel from adapting a universal background model (UBM) to a recording-specific model that captures the speaker variability in the recording. Examples of this in the literature are:

- The system described in Section 2.2.3 uses maximum a-posteriori (MAP) adaptation to adapt the means of a GMM UBM. Motivated by the KL divergence between two probability distribution functions, the resultant feature expansion is the Gaussian mean supervector (GSV), which consists of the stacked adapted means, and the kernel is a weighted inner product between the supervectors. Since only the means of the models were adapted, it is reasonable to expect the feature expansion to be the mean GSV. However, what is not obvious, yet crucial for good performance, is the choice of weighting in the inner product. Thus, an advantage to following the KL divergence approach, is that it motivates a choice of weighting, based on the UBM covariances and mixture weights, which performs well.

- [4] adapts both the means and covariances of a GMM UBM to a given recording using constrained maximum likelihood linear regression (CMLLR), which adapts the parameters via an affine transformation shared amongst the means and covariances of multiple mixture components. In this case, since the covariances were also adapted, one choice of the feature expansion is a vector consisting of stacking the adapted parameters, means and covariances. Alternatively, one could argue that the transformation captures all the deviation of the recording from the UBM. The argument, thus, suggests another choice for the feature expansion, such as the one used in [4], which consists of stacking the parameters of the affine transformation. This motivation does not, however, suggest a way to weight the transform vectors in the kernel inner product, and [4] resorts to min-max normalization of the vectors.

- In [5], maximum-likelihood linear regression (MLLR) adapts, via a shared affine transformation, the means of the GMMs of a speaker independent large vocabulary speech recognition (LVCSR) system to a given recording. Similar to the previous example, one could use either the MLLR-transform vectors as an expansion, or the mean GSV. However, since the UBM is a LVCSR, the number of Gaussian mixture components are significantly greater than in a GMM UBM. This makes the mean GSV a computationally expensive choice for a feature expansion. The high computational cost and the argument that the MLLR transforms capture the variability are likely what led to using the transform vectors in [5]. As in the CMLLR case, the absence of a motivated weighting leads to using rank-normalized transform vectors in the kernel inner product.

In this thesis we choose to use MLLR adaptation of the means of a GMM UBM to avoid the overhead of the LVCSR system, and in hopes that the constrained nature of the MLLR transform may help mitigate channel effects. Another goal of this work, is to derive a well motivated kernel in the MLLR-transform space that proposes a weighting of the kernel inner product that outperforms ad-hoc techniques, such as min-max and rank normalization. Note that, although we restrict ourselves to GMM adaptation, our kernel derivation and the resultant weighting transfers to the case where the UBM is a LVCSR, as in [5].

This chapter will begin with a brief overview of MLLR adaptation, followed by the two expansions we will be considering: mean GSV and MLLR-transform vector. We then present two kernels in the MLLR-transform vector space which are motivated by the KL divergence. Implementation details for the MLLR transformation are then presented, followed by a discussion on how this work extends to LVCSR UBMs. Finally, we present experimental results that compare our KL divergence kernels to ad-hoc ones.

3.1 Maximum Likelihood Linear Regression

Maximum likelihood linear regression (MLLR) adaptation adapts the means of the mixture components of a GMM by applying an affine transformation. The same affine transform may be shared by all the mixture components:

$$\mathbf{m}_i = \mathbf{A}\mathbf{m}_{UBM,i} + \mathbf{b} \qquad \forall i, \tag{3.1}$$

where $\mathbf{m}_{UBM,i}$ are the means of the unadapted GMM, and $\mathbf{m}_i$ are the adapted means.

Alternatively, the mixture components may be grouped into classes and a different affine transform shared by all the mixture components in each of the classes:

$$\mathbf{m}_i = \mathbf{A}_1\mathbf{m}_{UBM,i} + \mathbf{b}_1 \qquad \forall \mathbf{m}_i \in \text{Class}_1, \tag{3.2}$$

$$\mathbf{m}_i = \mathbf{A}_2\mathbf{m}_{UBM,i} + \mathbf{b}_2 \qquad \forall \mathbf{m}_i \in \text{Class}_2. \tag{3.3}$$

In both the single and multi-class cases the transforms are chosen to maximize the likelihood that the recording was generated by the adapted model [34]. The MLLR algorithm computes the transforms $\mathbf{A}$ and $\mathbf{b}$, not the transformed means $\mathbf{m}_i$ and subsequently additional computation is needed to obtain the transformed means.

Multi-class MLLR adaptation allows for more freedom in adapting the GMM, since all the means are not constrained to move the same way. The choice of how to group mixture components into the different classes and the number of classes is non-trivial. One can group the mixture components via a data-driven approach that combines together mixture components that are close in acoustic space. Alternatively, as in this work, the grouping can be done based on broad phonetic classes. We explore the two and four-class cases: the two-class case groups sonorants into one class and obstruents into the other, the four-class case further divides the sonorants into vowels and sonorant consonants and the obstruents into fricatives and stops. The two and four-class break-up is presented in Figure 3-1. As the number of classes increases, the amount of adaptation data assigned to each class decreases. This leads to instances where there is not enough adaptation data to obtain a good transform for a given class. A common method to handle these instances is to “back-off”

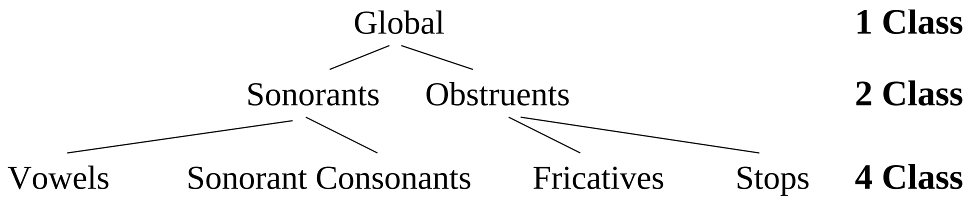

Figure 3-1: Class-division tree structure.

from the class-specific transform and use a more general one to transform the means of that class. For example, if there is not enough data to obtain a transform for the vowels we back-off and use the transform for the sonorants to adapt the vowels. More details on how the mixture components were chosen and the back-off technique used will follow in Section 3.4.

3.2 MLLR Feature Expansions

The feature expansion is the representation of a recording in a high-dimensional vector space. We will focus on the case of two-class MLLR adaptation and will present two expansions which are byproducts of this adaptation. The expansions for the global and four-class MLLR adaptation are a simple extension of the following.

The UBM is an M mixture diagonal covariance GMM, $g(\mathbf{r})$. It is formed by a weighted sum of two $M/2$ mixture GMMs: the first $M/2$ mixture components are assigned to the sonorants and the rest to the obstruents. The process of assigning components and the choice of the weighting ($p_{\mathbf{s}}$ and $p_{\mathbf{o}}$) are discussed in more detail in Section 3.4.

$$g(\mathbf{r}) \quad = \quad p_{\mathbf{s}} \sum_{i=1}^{M/2} \lambda_i \mathcal{N}(\mathbf{r}; \mathbf{m}_{UBM,i}, \mathbf{\Sigma}_i) + p_{\mathbf{o}} \sum_{i=M/2+1}^{M} \lambda_i \mathcal{N}(\mathbf{r}; \mathbf{m}_{UBM,i}, \mathbf{\Sigma}_i), \qquad (3.4)$$

where $\mathcal{N}(\mathbf{r}; \mathbf{m}_{UBM,i}, \mathbf{\Sigma}_i)$ is a Gaussian with mean $\mathbf{m}_{UBM,i}$ and covariance $\mathbf{\Sigma}_i$. Adapting the means of the UBM via two-class MLLR to a given recording $\mathbf{R}_\alpha$ produces transformation matrices and offset vectors $\mathbf{A_s}$ and $\mathbf{b_s}$ for the sonorants and $\mathbf{A_o}$ and $\mathbf{b_o}$ for the obstruents. These can be used to adapt the means of the UBM assigned to the sonorants and obstruents respectively.

The first expansion is the Gaussian mean supervector $\mathbf{m}$, which is constructed by stacking the means of the adapted model. The second is the MLLR-transform vector $\boldsymbol{\tau}$ which consists of stacking the transposed rows of the transform matrix $\mathbf{A_s}$ separated by the corresponding entries of the vector $\mathbf{b_s}$ followed by the transposed rows of $\mathbf{A_o}$ separated by the corresponding entries of $\mathbf{b_o}$. The process is shown in Figure 3-2.

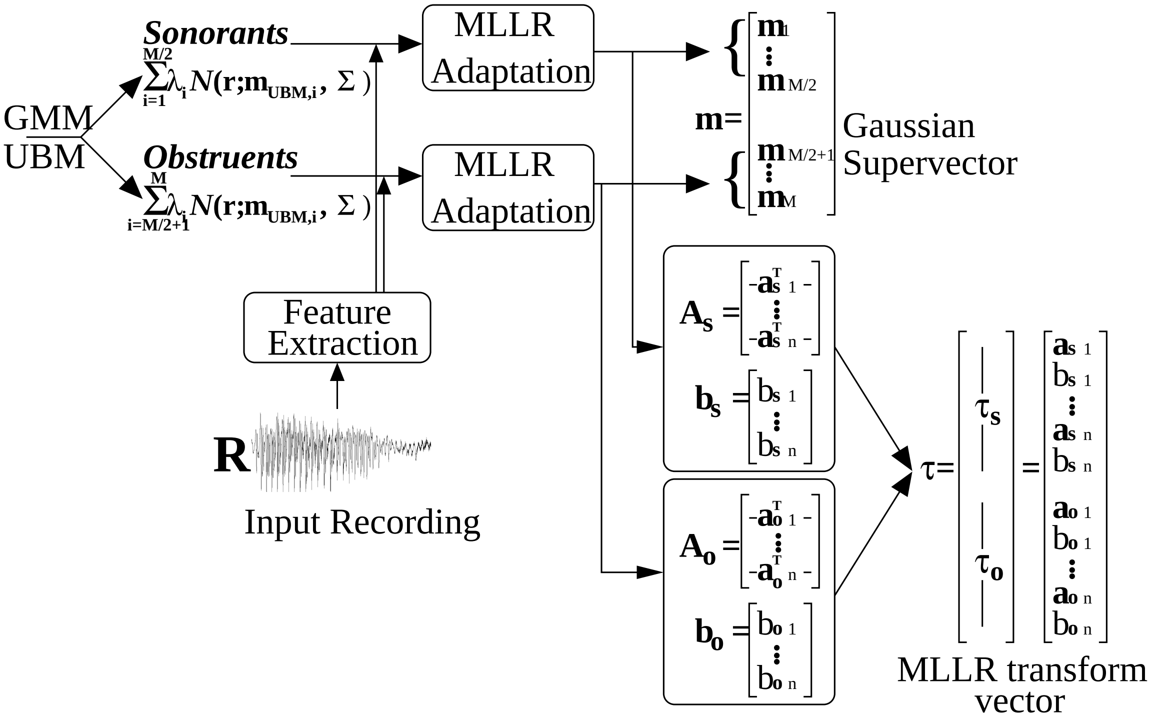

Figure 3-2: Two choices of feature expansions for the two-class case.

3.3 Choice of Metrics

A major component of an SVM is the kernel which defines a metric that induces a distance between two different points in the feature expansion space. In our context, this translates to defining a distance between two recordings. In this section we will discuss the different kernels we have explored. Our focus will be on the Gaussian supervector kernel since it is well-motivated and performs well.

3.3.1 Gaussian Supervector (GSV) Kernel

Suppose we have two recordings, $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$. We adapt the means of the GMM UBM $g(\mathbf{r})$ to obtain two new GMMs $g_\alpha(\mathbf{r})$ and $g_\beta(\mathbf{r})$ respectively that represent the recordings. This results in mean supervectors, $\mathbf{m}_\alpha$ and $\mathbf{m}_\beta$. A natural distance between the two recordings is the KL divergence between the two adapted GMMs,

$$D(g_\alpha \| g_\beta) = \int_{R^n} g_\alpha(\mathbf{r}) \log \left(\frac{g_\alpha(\mathbf{r})}{g_\beta(\mathbf{r})} \right) d\mathbf{r} \tag{3.5}$$

Unfortunately, the KL divergence does not satisfy the Mercer condition, so using it in an SVM is difficult.

Instead of using the KL divergence directly, we consider an approximation [35] which upper bounds it,

$$d(\mathbf{m}_\alpha, \mathbf{m}_\beta) = \frac{1}{2} \sum_{i=1}^{M} \lambda_i (\mathbf{m}_{\alpha,i} - \mathbf{m}_{\beta,i}) \boldsymbol{\Sigma}_i^{-1} (\mathbf{m}_{\alpha,i} - \mathbf{m}_{\beta,i}). \tag{3.6}$$

The distance in (3.6) has a corresponding kernel function [35]:

$$K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \sum_{i=1}^{M} \left(\sqrt{\lambda_i} \boldsymbol{\Sigma}_i^{-\frac{1}{2}} \mathbf{m}_{\alpha,i} \right)^t \left(\sqrt{\lambda_i} \boldsymbol{\Sigma}_i^{-\frac{1}{2}} \mathbf{m}_{\beta,i} \right),$$

which can be rewritten in terms of the mean supervectors:

$$K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \mathbf{m}_\alpha^{T}\text{‘} \boldsymbol{\Delta} \mathbf{m}_\beta. \tag{3.7}$$

The GSV kernel in (3.7) results in a diagonal weighting between the mean supervectors. When global MLLR adaptation is used, we will call the resulting kernel the G_MLLRSV kernel.

3.3.2 G_MLLRSV Kernel in MLLR Transform Space

MLLR adaptation transforms the means of all the mixtures of the UBM GMM by the same affine transformation in equation (3.1). This constraint allows us to derive a kernel in MLLR-transform vector space that is equivalent to the Gaussian supervector kernel. We begin by replacing the adapted means in equation (3.7) with the affine transform of the UBM means.

$$K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\mathbf{A}\mathbf{m}_{UBM,i} + \mathbf{b}) \right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\mathbf{C}\mathbf{m}_{UBM,i} + \mathbf{d}) \right), \tag{3.8}$$

where M is the number of mixtures of the UBM, $\mathbf{m}_{UBM,i}$ is the mean vector of the i^{th} mixture of the UBM, and $\boldsymbol{\Delta}_i = \lambda_i \boldsymbol{\Sigma}_i^{-1}$ which is diagonal. Expanding equation (3.8) yields

$$\begin{aligned} K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = &\ \textstyle\sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{A}\mathbf{m}_{UBM,i}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{C}\mathbf{m}_{UBM,i}\right) \\ + &\ \textstyle\sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{A}\mathbf{m}_{UBM,i}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{d}\right) \\ + &\ \textstyle\sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{b}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{C}\mathbf{m}_{UBM,i}\right) \\ + &\ \textstyle\sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{b}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{d}\right). \end{aligned} \tag{3.9}$$

We will focus on the first term in equation (3.9). Note that tr$(\mathbf{A})$ is the trace of the matrix $\mathbf{A}$, $\mathbf{e}_k$ is a vector that has a value of 1 as its k^{th} entry and 0 for every other entry, Δ_{ik} is the k^{th} diagonal element of the diagonal matrix $\boldsymbol{\Delta}_i$, n is the number of rows in $\mathbf{A}$, and that $\mathbf{a}_k$ is the transpose of the k^{th} row of the matrix $\mathbf{A}$.

$$\begin{aligned} \sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{A}\mathbf{m}_{UBM,i}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{C}\mathbf{m}_{UBM,i}\right) = &\ \textstyle\sum_{i=1}^{M} \text{tr}\left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{A}\mathbf{m}_{UBM,i}\mathbf{m}_{UBM,i}^T \mathbf{C}^T \boldsymbol{\Delta}_i^{\frac{1}{2}}\right) \\ = &\ \textstyle\sum_{i=1}^{M} \text{tr}\left(\boldsymbol{\Delta}_i \mathbf{A}\mathbf{m}_{UBM,i}\mathbf{m}_{UBM,i}^T \mathbf{C}^T\right) \\ = &\ \textstyle\sum_{i=1}^{M} \text{tr}\left((\sum_{k=1}^{n} \Delta_{ik}\mathbf{e}_k\mathbf{e}_k^T)\mathbf{A}\mathbf{m}_{UBM,i}\mathbf{m}_{UBM,i}^T \mathbf{C}^T\right) \\ = &\ \textstyle\sum_{k=1}^{n} \sum_{i=1}^{M} \text{tr}\left(\mathbf{e}_k\mathbf{e}_k^T \mathbf{A}(\Delta_{ik}\mathbf{m}_{UBM,i}\mathbf{m}_{UBM,i}^T)\mathbf{C}^T\right) \\ = &\ \textstyle\sum_{k=1}^{n} \text{tr}\left(\mathbf{e}_k^T \mathbf{A}(\sum_{i=1}^{M} \Delta_{ik}\mathbf{m}_{UBM,i}\mathbf{m}_{UBM,i}^T)\mathbf{C}^T\mathbf{e}_k\right) \\ = &\ \textstyle\sum_{k=1}^{n} \mathbf{a}_k^T(\sum_{i=1}^{M} \Delta_{ik}\mathbf{m}_{UBM,i}\mathbf{m}_{UBM,i}^T)\mathbf{c}_k \\ = &\sum_{k=1}^{n} \mathbf{a}_k^T \mathbf{H}_k \mathbf{c}_k. \end{aligned} \tag{3.10}$$

In a similar fashion we can rewrite the remaining terms in equation (3.9) as follows:

$$\sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{A}\mathbf{m}_{UBM,i}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{d}\right) = \sum_{k=1}^{n} d_k \mathbf{a}_k^T \mathbf{h}_k, \tag{3.11}$$

$$\sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{b}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{C}\mathbf{m}_{UBM,i}\right) = \sum_{k=1}^{n} b_k \mathbf{h}_k^T \mathbf{c}_k, \tag{3.12}$$

$$\sum_{i=1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{b}\right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} \mathbf{d}\right) = \sum_{k=1}^{n} b_k d_k \boldsymbol{\delta}_k, \tag{3.13}$$

where $\mathbf{h}_k = \sum_{i=1}^{M} \Delta_{ik}\mathbf{m}_{UBM,i}$, b_k is the k^{th} element of the vector $\mathbf{b}$, and $\boldsymbol{\delta}_k = \sum_{i=1}^{M} \Delta_{ik}$. Therefore the Gaussian supervector kernel can be rewritten as

$$\begin{aligned} K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = & \sum_{k=1}^{n} \mathbf{a}_k^T \mathbf{H}_k \mathbf{c}_k + \sum_{k=1}^{n} d_k \mathbf{a}_k^T \mathbf{h}_k + \sum_{k=1}^{n} b_k \mathbf{h}_k^T \mathbf{c}_k + \sum_{k=1}^{n} b_k d_k \boldsymbol{\delta}_k \\ = & \qquad \boldsymbol{\tau}_\alpha^T \mathbf{Q} \boldsymbol{\tau}_\beta, \end{aligned} \tag{3.14}$$

where $\boldsymbol{\tau}$ is the MLLR-transform vector defined in Section 3.1.

The matrix $\mathbf{Q}$ must be positive-definite because equation (3.14) computes the same quantity as (3.7). $\mathbf{Q}$ is a block diagonal matrix consisting of n blocks $\mathbf{Q}_k$ of size $(n+1)$x$(n+1)$. Equation (3.15) shows the structure of the blocks $\mathbf{Q}_k$,

$$\mathbf{Q}_k = \begin{pmatrix} \mathbf{H}_k & \mathbf{h}_k \\ \mathbf{h}_k^T & \boldsymbol{\delta}_k \end{pmatrix}. \tag{3.15}$$

It is important to note that since $\mathbf{Q}$ depends only on the UBM means, covariances and mixture weights it can be computed offline. The block-diagonal nature of $\mathbf{Q}$ also allows us to easily compute its square root. This in turn allows us to apply the model compaction technique in [35].

An advantage of equation (3.14) over (3.7) is that the number of multiplies it requires only depends on the size of the GMM feature vectors (38 in our case) and not on the number of mixtures in the GMM. Another advantage is that it does not require transforming the means which saves computation and removes the need for storing the adapted means. These two advantages and the block diagonal structure of $\mathbf{Q}$ result in an overall reduction of the number of multiplies from $O(n*(M+M^2))$ in equation (3.7) to $O((n+1)^3)$ in (3.14), where n is the size of the GMM feature vectors and M is the number of mixtures in the GMM. This equates to an order of magnitude reduction in the number of multiplies for our case. Note that this reduction in number of multiplies and storage requirements will have a significantly greater impact if this kernel is applied to an LVCSR system.

3.3.3 MC_MLLRSV kernel

In this section we present the MC_MLLRSV kernel which extends the G_MLLRSV formulation to the case where multi-class MLLR is used. Since $\boldsymbol{\Delta}$ in equation (3.7) is a diagonal matrix and $\mathbf{m}$ is the stacked means of the different classes, then the multi-class extension to the GSV kernel is:

$$K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = p_{\mathbf{s}} K_{SV,S}(\mathbf{R}_\alpha, \mathbf{R}_\beta) + p_{\mathbf{o}} K_{SV,O}(\mathbf{R}_\alpha, \mathbf{R}_\beta), \tag{3.16}$$

where $K_{SV,S}(\mathbf{R}_\alpha, \mathbf{R}_\beta)$ and $K_{SV,O}(\mathbf{R}_\alpha, \mathbf{R}_\beta)$ are the class-dependent GSV kernels for the sonorants and obstruents respectively.

Similar to the global case, we can implement the multi-class MLLRSV in MLLR-transform space: we begin by replacing the adapted means in equation (3.16) with the affine transforms of the UBM means. $\mathbf{A_s}$, $\mathbf{A_o}$, $\mathbf{b_s}$, $\mathbf{b_o}$ are the transforms for $\mathbf{R}_\alpha$ and $\mathbf{C_s}$, $\mathbf{C_o}$, $\mathbf{d_s}$, $\mathbf{d_o}$ are the transforms for $\mathbf{R}_\beta$.

$$\begin{aligned} K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) =& p_\mathbf{s} \sum_{i=1}^{M/2} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\mathbf{A_s m}_{UBM,i} + \mathbf{b_s}) \right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\mathbf{C_s m}_{UBM,i} + \mathbf{d_s}) \right) \\ &+ p_\mathbf{o} \sum_{i=M/2+1}^{M} \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\mathbf{A_o m}_{UBM,i} + \mathbf{b_o}) \right)^T \left(\boldsymbol{\Delta}_i^{\frac{1}{2}} (\mathbf{C_o m}_{UBM,i} + \mathbf{d_o}) \right), \end{aligned}$$

where $\mathbf{m}_{UBM,i}$ is the mean vector of the i^{th} mixture component of the UBM, the diagonal matrix $\boldsymbol{\Delta}_i = \lambda_i \boldsymbol{\Sigma}_i^{-1}$.

After similar manipulation as was done for the global MLLR case, we obtain:

$$\begin{aligned} K_{SV,S}(\mathbf{R}_\alpha, \mathbf{R}_\beta) =& \sum_{k=1}^{n} b_{\mathbf{s}k} d_{\mathbf{s}k} \boldsymbol{\delta}_{\mathbf{s}k} + \sum_{k=1}^{n} \mathbf{a}_{\mathbf{s}k}^T \mathbf{H}_{\mathbf{s}k} \mathbf{c}_{\mathbf{s}k} + \sum_{k=1}^{n} d_{\mathbf{s}k} \mathbf{a}_{\mathbf{s}k}^T \mathbf{h}_{\mathbf{s}k} + \sum_{k=1}^{n} b_{\mathbf{s}k} \mathbf{h}_{\mathbf{s}k}^T \mathbf{c}_{\mathbf{s}k} \\ =& \boldsymbol{\tau}_{\mathbf{s}\alpha}^T \mathbf{Q_s} \boldsymbol{\tau}_{\mathbf{s}\beta} \end{aligned} \tag{3.17}$$

where M is the number of rows in $\mathbf{A_s}$, $\mathbf{a}_{\mathbf{s}k}$ and $\mathbf{c}_{\mathbf{s}k}$ are the transpose of the k^{th} rows of $\mathbf{A_s}$ and $\mathbf{C_s}$ respectively, $b_{\mathbf{s}k}$ and $d_{\mathbf{s}k}$ are the k^{th} elements of $\mathbf{b_s}$ and $\mathbf{d_s}$ respectively, Δ_{ik} is the k^{th} diagonal element of the diagonal matrix $\boldsymbol{\Delta}_i$, $\mathbf{H}_{\mathbf{s}k} = \sum_{i=1}^{M/2} \Delta_{ik} \mathbf{m}_{UBM,i} \mathbf{m}_{UBM,i}^T$, $\mathbf{h}_{\mathbf{s}k} = \sum_{i=1}^{M/2} \Delta_{ik} \mathbf{m}_{UBM,i}$, $\boldsymbol{\delta}_{\mathbf{s}k} = \sum_{i=1}^{M/2} \Delta_{ik}$, $\boldsymbol{\tau}_{\mathbf{s}\alpha}$ and $\boldsymbol{\tau}_{\mathbf{s}\beta}$ are the sonorant parts of the MLLR-transform vectors of the recordings, and $\mathbf{Q_s}$ is a block diagonal matrix consisting of n blocks $\mathbf{Q}_{\mathbf{s}k}$ of size $(n+1)$x$(n+1)$. Equation (3.18) shows the structure of the blocks:

$$\mathbf{Q}_{\mathbf{s}k} = \begin{pmatrix} \mathbf{H}_{\mathbf{s}k} & \mathbf{h}_{\mathbf{s}k} \\ \mathbf{h}_{\mathbf{s}k}^T & \boldsymbol{\delta}_{\mathbf{s}k} \end{pmatrix}. \tag{3.18}$$

Note that the summations in $\mathbf{H}_{\mathbf{s}k}$, $\mathbf{h}_{\mathbf{s}k}$ and $\boldsymbol{\delta}_{\mathbf{s}k}$ are from $i = 1$ to $M/2$, only over the mixture components pertaining to the sonorant class. With this in mind the form of the obstruent part of the kernel is

$$K_{SV,O}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \boldsymbol{\tau}_{\mathbf{o}\alpha}^T \mathbf{Q_o} \boldsymbol{\tau}_{\mathbf{o}\beta}, \tag{3.19}$$

where the summations in $\mathbf{H}_{\mathbf{o}k}$, $\mathbf{h}_{\mathbf{o}k}$ and $\boldsymbol{\delta}_{\mathbf{o}k}$ are from $i = M/2+1$ to M, only over the mixture components pertaining to the obstruent class.

From equations (3.17) and (3.19) we note that the GSV kernel can be written as a weighted inner product between the MLLR-transform vectors.

$$\begin{aligned} K_{SV}(\mathbf{R}_\alpha, \mathbf{R}_\beta) &= \begin{bmatrix} \boldsymbol{\tau}_{\mathbf{s}\alpha}^T & \boldsymbol{\tau}_{\mathbf{o}\alpha}^T \end{bmatrix} \begin{bmatrix} p_{\mathbf{s}}\mathbf{Q}_{\mathbf{s}} & \mathbf{0} \\ \mathbf{0} & p_{\mathbf{o}}\mathbf{Q}_{\mathbf{o}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}_{\mathbf{s}\beta} \\ \boldsymbol{\tau}_{\mathbf{o}\beta} \end{bmatrix} \\ &= \boldsymbol{\tau}_\alpha^T \mathbf{Q} \boldsymbol{\tau}_\beta \end{aligned} \tag{3.20}$$

It is important to note that, similar to the global MLLR case, since the $\mathbf{Q}$ matrix depends only on the UBM means, covariances and mixture weights it can be computed offline.

3.3.4 Alternative Kernels in MLLR Transform Space

We also explore four alternative kernels in MLLR-transform vector space. The first replaces the matrix $\mathbf{Q}$ by its diagonal approximation, which we refer to as the diag-supervector (D_MLLRSV) kernel. The second is the zero-one (Z-O) kernel which subtracts the means and divides by the standard deviations along each of the feature dimensions of the MLLR-transform vectors (determined from a held-out dataset). The third is the Frobenius (FROB) kernel which does not apply any scale or shift to the MLLR-transform vectors; tr$([\mathbf{Ab}]^T[\mathbf{Cd}])$. The last is the rank-normalized [5] (R-N) kernel which rank normalizes the MLLR-transform vectors.

3.4 MC_MLLRSV Implementation

There are a number of issues that have to be addressed when building the multi-class MLLR/GMM system. The first, is how to divide the mixture components of the GMM into multiple classes. For the two-class case, we chose to perform the divide along broad phonetic classes: sonorants and obstruents. However, since our UBM is not an LVCSR system where it is clear which mixture components belong to which phoneme and thus to which of our two classes, we have to explicitly assign them: we assign the first $M/2$ mixture components to the sonorants class and the remaining $M/2$ to the obstruents class. We also perform open-loop phonetic recognition on all the data used to train the UBM, the background, and the speaker recognition system and to test the system; this allows us to assign which part of the data will be used to train/test each class. We also tried unequal splitting of the GMM mixture components amongst the classes, however, this reduced performance.

Second, we use EM to train two class-dependent $M/2$ mixture GMMs each using the corresponding class-specific UBM training data. The M mixture GMM UBM is then created by combining the two $M/2$ mixture GMMs and scaling their weights so that the weights of the UBM add up to 1. The scaling, $p_{\mathbf{s}}$ and $p_{\mathbf{o}}$, is done according to the class priors, calculated as the percentage of frames assigned to each of the two classes in the background training data.

Third, the MLLR-transformation matrix and offset vector for each of the two classes are computed by separately adapting, via MLLR, the class-dependent GMMs using only the frames of the adaptation recording corresponding to each class. If the number of frames of the recording assigned to a class is below a set number (empirically we chose 500) we back-off and use the full M mixture GMM and all the frames of the recording to obtain the MLLR-transformation matrix and vector. This transform computed by backing-off is then used to adapt *only* the $M/2$ means of the *original* class-dependent GMM. Similarly, in the four-class case if the number of frames allocated to one of the four classes is below 250 then for that class one would back-off one level, e.g. from Vowels to Sonorants; if after backing-off one level the number of allocated frames is less than 500 then one would back-off one more level.

3.5 MC_MLLRSV Kernel for LVCSR systems

The LVCSR/SVM system presented in [5] uses MLLR adaptation with a speaker independent LVCSR system and a kernel consisting of an inner product between rank-normalized transform-vectors. In the next section we show the advantage of the GSV kernel over other kernels that are inner products between normalized MLLR-transform vectors, including the one used in [5], for the case where the UBM is a GMM. Unfortunately, the GSV kernel, if applied in its original form (3.16), can be computationally prohibitive since the number of multiplies increases as $O(M^2)$ where M is the number of Gaussian mixture components in the system, which is typically more than a hundred thousand for an LVCSR system. However, since MLLR adaptation is being used to adapt the means, one can follow the steps taken in Section 3.3.3 to derive a similar way to compute the GSV kernel in terms of an inner product between the MLLR-transform vectors significantly reducing computation.

3.6 Experiments

We performed experiments on the 2006 NIST speaker recognition (SRE) corpus. We focused on the single-side 1 conversation train, single-side 1 conversation test, and the multi-language handheld telephone task (the core test condition) [18]. This setup resulted in $3,612$ true trials and $47,836$ false trials.

For feature extraction, a 19-dimensional MFCC vector is found from pre-emphasized speech every 10 ms using a 20 ms Hamming window. Delta-cepstral coefficients are computed over a ± 2 frame span and appended to the cepstra producing a 38-dimensional feature vector. An energy-based speech detector is applied to discard vectors from low-energy frames. To mitigate channel effects, RASTA and mean and variance normalization are applied to the features.

The GMM UBM consists of 512 mixture components. The GMM UBM was trained using EM from the following corpora: Switchboard 2 phase 1, Switchboard 2 phase 4 (cellular), and OGI national cellular. For the two-class case, two class-specific $M/2 = 256$ mixture GMM UBMs were trained using EM on the corresponding class-dependent parts of the data. These GMMs were combined with weights $p_{\mathbf{s}} = .71$ and $p_{\mathbf{o}} = .29$ to form a $M = 512$ mixture GMM UBM. For the four-class case, four class-specific $M/4 = 128$ mixture GMM UBMs were trained and combined to form a 512 mixture GMM with weights: .46 for vowels, .25 for sonorant consonants, .15 for fricatives, and .14 for stops.

We produced the feature expansion on a per conversation (recording) basis using multi-class MLLR adaptation. The adaptation was done per class-specific GMM. We used the HTK toolbox version 3.3 [36] to perform one iteration of MLLR to obtain the transformation. The various kernels were implemented using SVMTorch as an SVM trainer [37]. A background for SVM training consists of SVM features labeled as -1 extracted from recordings from example impostors [35]. An SVM background was obtained by extracting SVM features from 4174 conversations in a multi-language subset of the LDC Fisher corpus. In our experiments the size of the SVM features are $38 * 512 + 1$ for the mean supervector features and $38 * 39 + 1$ for the MLLR-transform vector features; note that we stack an element of value 1 at the end of each feature vector to incorporate the bias ξ into the SVM features.

For enrollment of target speakers, we produced 1 SVM feature vector per conversation side. We then trained an SVM model using the target SVM feature and the SVM background. This resulted in selecting support vectors from the target speaker and background SVM feature vectors and assigning the associated weights.

3.7 Results and Discussion

We will present our results in two parts: the first will highlight the difference between different kernels in MLLR-transform space when global MLLR adaptation is used, and the second will present the results for the MLLRSV kernel for the global and multi-class cases.

3.7.1 MLLR Transform Space Kernels Comparison

We compared the G_MLLRSV kernel, the D_MLLRSV kernel, the Z-O kernel, the FROB kernel, the R-N kernel, and a MAP Gaussian supervector kernel (MAPSV) as in [35] where the UBM is adapted via MAP adaptation instead of MLLR. Equal error rates (EER) and NIST minimum decision cost function (DCF) for the various kernels are shown in Table 3.1 and Figure 3-3.

The results show that of the examined kernels, the G_MLLRSV kernel yields the best performance, followed by the D_MLLRSV kernel. We believe the superiority of G_MLLRSV is due to its derivation from an approximation of the KL divergence as a distance between two GMMs. When examining the results for the diagonally-weighted kernels in MLLR-transform vector space we note that D_MLLRSV kernel (the diagonal approximation to the G_MLLRSV kernel) produced the best

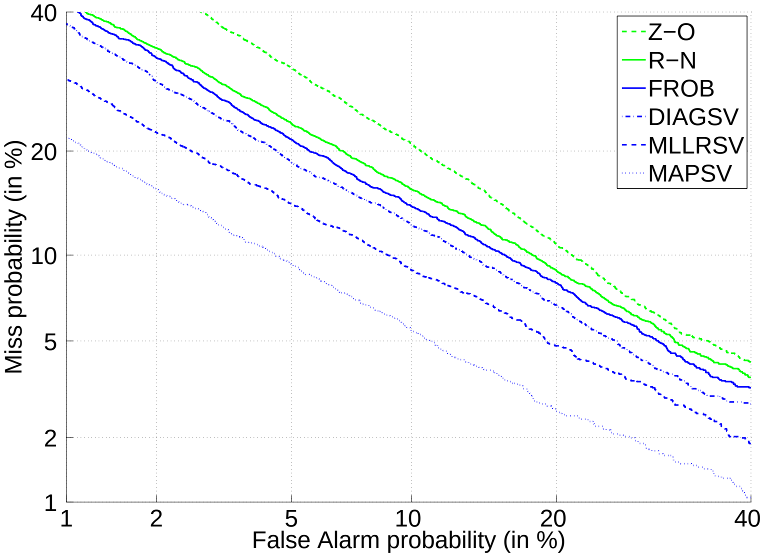

Figure 3-3: DET plot of the MLLR kernels and the baseline.

Table 3.1: EER and min DCF scores of the MLLR kernels and the baseline.

Kernel	EER	DCF
Z-O	14.95%	.064
R-N	13.19%	.051
FROB	12.38%	.05
D_MLLRSV	11.43%	.047
G_MLLRSV	9.46%	.039
MAPSV	7.24%	.031

results while the Z-O kernel produced the worst. To attempt and understand why the Z-O kernel performed poorly, we compared its scaling matrix to that of D_MLLRSV. The comparison showed that the Z-O kernel tended to emphasize dimensions that were weighted down by D_MLLRSV and vice versa.

3.7.2 Global vs Multi-Class MLLRSV

We compared the G_MLLRSV kernel system, the two and four-class MC_MLLRSV kernel systems (2C_MLLRSV and 4C_MLLRSV), and a state of the art MAPSV. Equal error rates (EER) and NIST minimum decision cost functions (DCF) for the various kernels are shown in Table 3.2.

Table 3.2: EER and min DCF scores.

Kernel	EER	min DCF
G_MLLRSV	9.46%	.039
2C_MLLRSV	7.81%	.035
4C_MLLRSV	8.19%	.037
MAPSV	7.24%	.031

Examining the results we note the following: two-class system yields a 15% improvement over the global system, however, there was no further improvement for the four-class system. This lack of improvement for the four-class is most likely due to the unstable transcripts provided by the open-loop phonetic recognizer, which becomes less reliable as the number of classes increases. It is important to note that the gain in performance obtained by two-class MLLR does require additional computation due to the phonetic recognition.

The performance of the 2C_MLLR system approaches but does not surpass that of the MAPSV system, as seen in Figure 3-4. However, it remains to be seen whether 2C_MLLR may outperform MAPSV in scenarios with high channel variability or with shorter training recordings. Under such conditions the constrained nature of 2C_MLLR may cause it to outperform MAPSV.

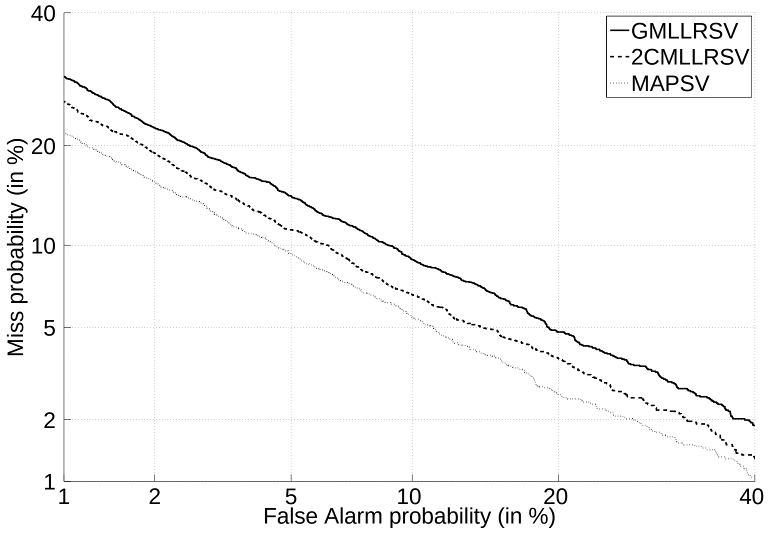

Figure 3-4: DET plot of the global MLLRSV, two-class MLLRSV, and MAPSV kernels.

3.8 Discussion

In this chapter we examined a vector space representation and corresponding metric that are derived from adapting a GMM universal background model via maximum likelihood linear regression adaptation. Support vector machines whose kernels are based on the derived metrics performed well in speaker verification tasks with the results clearly highlighting the importance of choosing properly motivated kernels. Experiments on the NIST SRE 2006 corpus showed the superiority of our proposed G_MLLRSV kernel over ad-hoc kernels in the MLLR-transform space. We also showed that using the two-class MLLRSV kernel we approach state of the art performance. The main contribution of this work is the formulation of the MLLR supervector kernel in MLLR-transform vector space. The advantage of this formulation is that its storage and computation requirements do not increase with the number of mixtures. This advantage allows the use of the MLLRSV kernel in systems such as [5], where using the original GSV kernel is prohibitive due to the large size of the mean supervectors. Possible avenues for future work are: to use data-driven class selection rather than phonetic ones used here, to apply the KL-motivated MLLR-transform kernel to a system with a LVCSR UBM, and to use lattice-based MLLR [38] which is more robust to transcription errors.

Chapter 4

Variability-Compensated Support Vector Machines

In a classification task there are two types of variability: across class (good) which reflects the anticipated diversity needed for proper classification, and within class (bad) which introduces undesirable information that confuses the classifier. A good classifier should, therefore, exploit the good and mitigate the bad. This chapter proposes a method to leverage class-labeled auxiliary data to do this when instances are represented in an inner-product space.

In Section 2.2.3 and in the previous chapter we presented several ways to map recordings into an inner-product space that contained the inter-speaker variability needed for speaker verification. However, this space also contains within-speaker (nuisance), e.g. channel and language, variability which is undesirable. Techniques for handling nuisance, such as nuisance attribute projection (NAP) and within class covariance normalization (WCCN), Section 2.2.4, are already used in SVM speaker verification. More recently, joint factor analysis (JFA), Section 2.2.5, used a Bayesian framework to incorporate nuisance and inter-speaker variability into the verification task.

In this chapter, we introduce variability-compensated SVM (VCSVM) which is a method to handle both the good and bad variability by incorporating them directly into the SVM optimization. We will begin by motivating and describing our approach in a nuisance compensation framework. Modifications to the algorithm are then presented that allow for handling inter-speaker variability. We then discuss a probabilistic interpretation of the algorithm and finally present experimental results that demonstrate the algorithm's efficacy.

4.1 Importance of Handling Nuisance Variability

Evidence of the importance of handling variability can be found in the discrepancy in verification performance between one, three and eight conversation enrollment tasks for the same SVM system. Specifically, for the MAP Gaussian supervector SVM system, Section 2.2.3, performance improves from 5.0% EER for one conversation enrollment to 2.9% and 2.6% for three and eight, on trials of the NIST SRE-Eval 06. One explanation for this is that when only one target conversation is available to enroll a speaker, then the orientation of the separating hyperplane is set by the impostor recordings. As more target enrollment recordings are provided the orientation of the separating hyperplane can change drastically, as sketched in Figure 4-1. The additional information that the extra enrollment recordings provide is intra- (or within-) speaker variability, due to channel, language, and other nuisance variables.

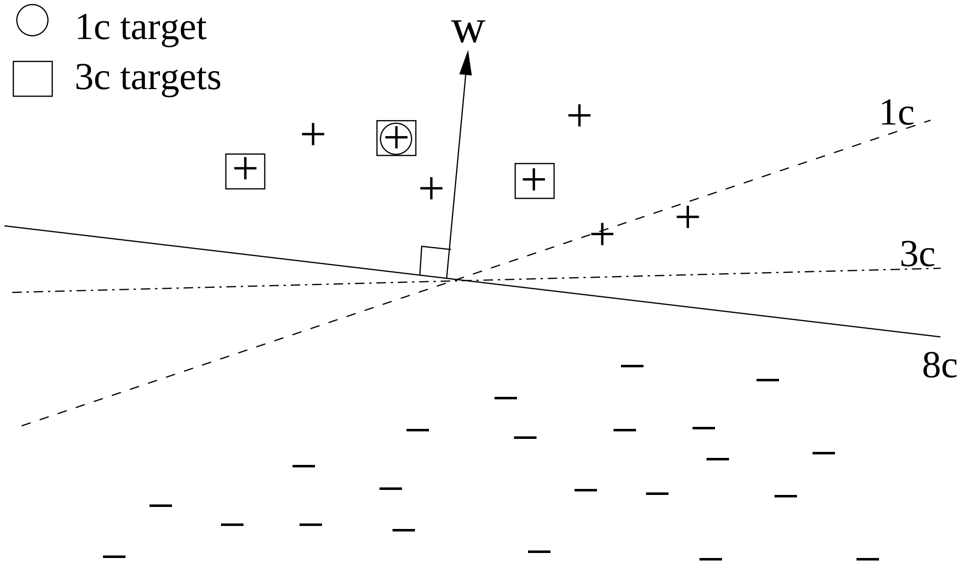

Figure 4-1: Different separating hyperplanes obtained with 1, 3, and 8 conversation enrollment.

If an estimate of the principal components of intra-speaker variability for a given speaker were available then one could prevent the SVM from using that variability when choosing a separating hyperplane. However, it is not possible, in general, to estimate intra-speaker variability for the target speakers. One could instead employ a speaker-labeled auxiliary set of recordings to obtain a global estimate of the within-speaker variability. An example algorithm that uses this global estimate is NAP, Section 2.2.4, which estimates a small subspace where the nuisance lives and removes it completely from the SVM features, i.e., it does not allow any information from the nuisance subspace to affect the SVM decision. Figure 4-2 sketches the effect of NAP on the orientation of the separating hyperplane.

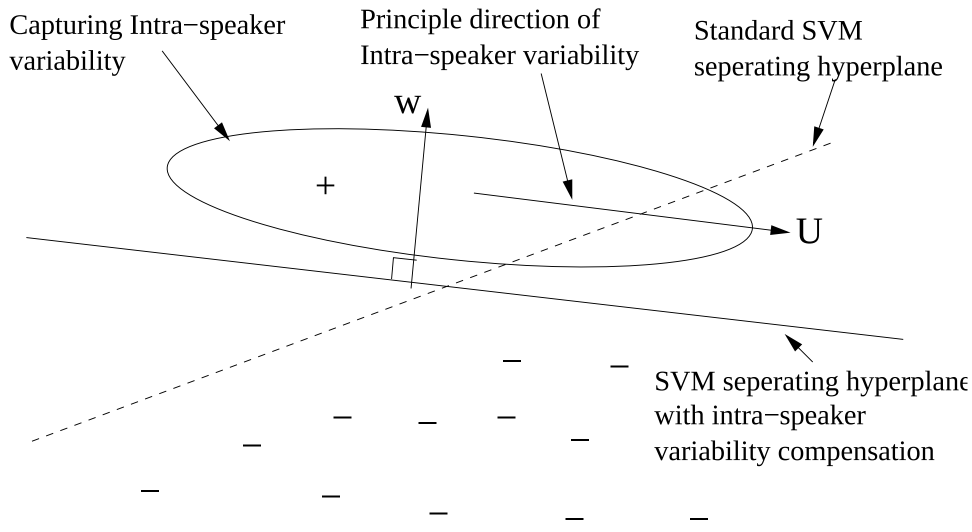
Capturing Intra−speaker variability
Principle direction of Intra−speaker variability
Standard SVM seperating hyperplane
w
+
U
SVM seperating hyperplane with intra−speaker variability compensation
−
−
−
−
−
−
−
−
−
−
−
−
−

Figure 4-2: Effect of removing the nuisance direction from the SVM optimization.

4.2 Handling Nuisance Variability

In this thesis we propose VCSVM to handle nuisance variability, which allows for varying the degree to which the nuisance subspace is avoided by the classifier rather than completely removing it: Assume that the nuisance subspace is spanned by a set of U orthonormal vectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_U\}$, e.g., top U eigenvectors of the within-class covariance matrix, and let $\mathbf{U}$ be the matrix whose columns are those eigenvectors. Let the vector normal to the separating hyperplane be $\mathbf{w}$. Ideally, if the nuisance was restricted to the subspace $\mathbf{U}$ then one would require the orthogonal projection of $\mathbf{w}$ in the nuisance subspace to be zero, i.e. $\left|\left|\mathbf{U}\mathbf{U}^T\mathbf{w}\right|\right|_2^2 = 0$. This requirement can be introduced directly into the primal formulation of the SVM optimization:

$$\begin{gathered}\min J(\mathbf{w}, \epsilon) = ||\mathbf{w}||_2^2/2 + \xi \left|\left|\mathbf{U}\mathbf{U}^T\mathbf{w}\right|\right|_2^2/2 + C\sum_{i=1}^{k}\epsilon_i \\ \text{subject to } l_i(\mathbf{w}^T\mathbf{m}_i + b) \geq 1 - \epsilon_i \;\&\; \epsilon_i \geq 0,\; i = 0, \ldots, k\end{gathered} \tag{4.1}$$

where $\xi \geq 0$, k is the total number of training examples, $\mathbf{m}_i$ denotes the recording specific SVM features (supervectors) and l_i denotes the corresponding labels. Note that the only difference between (4.1) and the standard SVM formulation is the addition of the $\xi\left|\left|\mathbf{U}\mathbf{U}^T\mathbf{w}\right|\right|_2^2$ term, where ξ is a tunable (on some held out set) parameter that regulates the amount of bias desired. If $\xi = \infty$ then this formulation becomes similar to NAP compensation, and if $\xi = 0$ then we obtain the standard SVM formulation. Figure 4-3 sketches the separating hyperplane obtained for different values of ξ . We can rewrite the additional term in (4.1) as follows:

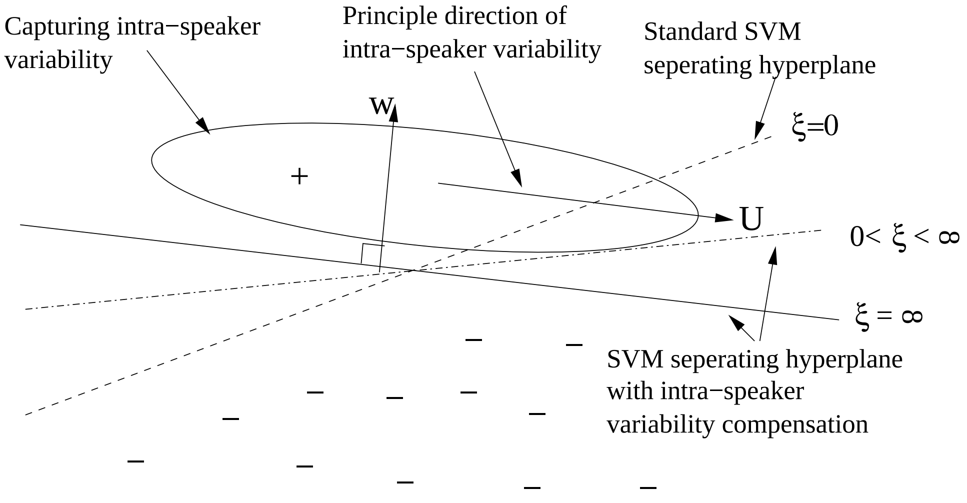
Capturing intra−speaker variability
Principle direction of intra−speaker variability
Standard SVM seperating hyperplane
w
ξ=0
+
U
0< ξ < ∞
ξ = ∞
SVM seperating hyperplane with intra−speaker variability compensation
−

Figure 4-3: Sketch of the separating hyperplane for different values of ξ.

$$\begin{aligned} \left|\left|\mathbf{U}\mathbf{U}^T\mathbf{w}\right|\right|_2^2 = & \quad (\mathbf{U}\mathbf{U}^T\mathbf{w})^T(\mathbf{U}\mathbf{U}^T\mathbf{w}) && (4.2)\\ = & \quad \mathbf{w}^T\mathbf{U}\mathbf{U}^T\mathbf{U}\mathbf{U}^T\mathbf{w} \\ = & \quad \mathbf{w}^T\mathbf{U}\mathbf{U}^T\mathbf{w}, && (4.3) \end{aligned}$$

where the final equality follows from the eigenvectors being orthonormal ($\mathbf{U}^T\mathbf{U} = \mathbf{I}$). Since $\mathbf{U}\mathbf{U}^T$ is a positive semi-definite matrix, we can follow the recipe presented in [39] to re-interpret this reformulation as a standard SVM with the bias absorbed into the kernel. As done in [39], we rewrite $J(\mathbf{w}, \epsilon)$ in (4.1) as:

$$J(\mathbf{w}, \epsilon) \quad = \quad \mathbf{w}^T(\mathbf{I} + \xi\mathbf{U}\mathbf{U}^T)\mathbf{w}/2 + C\sum_{i=1}^{k}\epsilon_i, \tag{4.4}$$

and since $(\mathbf{I} + \xi\mathbf{U}\mathbf{U}^T)$ is a positive definite symmetric matrix, then

$$J(\mathbf{w}, \epsilon) \quad = \quad \mathbf{w}^T\mathbf{B}^T\mathbf{B}\mathbf{w}/2 + C\sum_{i=1}^{k}\epsilon_i, \tag{4.5}$$

where $\mathbf{B}$ can be chosen to be real and symmetric and is invertible. A change of variables $\tilde{\mathbf{w}} = \mathbf{B}\mathbf{w}$ and $\tilde{\mathbf{m}} = \mathbf{B}^{-T}\mathbf{m}$ allows us to rewrite the optimization in (4.1) as

$$\begin{array}{ll} \text{minimize} & J(\mathbf{w}, \epsilon) = ||\tilde{\mathbf{w}}||_2^2 /2 + C \sum_{i=1}^{k} \epsilon_i \\ \text{subject to} & l_i(\tilde{\mathbf{w}}^T \tilde{\mathbf{m}}_i + b) \geq 1 - \epsilon_i \;\; \& \;\; \epsilon_i \geq 0, \quad i = 0, \ldots, k \end{array}$$

which is then the standard SVM formulation with the following kernel:

$$K(\mathbf{m}_i, \mathbf{m}_j) = \mathbf{m}_i^T \mathbf{B}^{-1} \mathbf{B}^{-T} \mathbf{m}_j = \mathbf{m}_i^T (\mathbf{I} + \xi \mathbf{U}\mathbf{U}^T)^{-1} \mathbf{m}_j. \tag{4.6}$$

Examining the kernel presented in (4.6), we realize that $(\mathbf{I}+\xi\mathbf{U}\mathbf{U}^T)$ can have large dimension. This is of concern, since the kernel requires its inverse. To circumvent this, we use the matrix inversion lemma [40] and $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ to obtain:

$$\begin{aligned} (\mathbf{I} + \xi \mathbf{U}\mathbf{U}^T)^{-1} &= \mathbf{I} - \sqrt{\xi}\mathbf{U}(\mathbf{I} + \xi \mathbf{U}^T\mathbf{U})^{-1}\sqrt{\xi}\mathbf{U}^T \\ &= \mathbf{I} - \xi\mathbf{U}[(1+\xi)\mathbf{I}]^{-1}\mathbf{U}^T \\ &= \mathbf{I} - \frac{\xi}{1+\xi}\mathbf{U}\mathbf{U}^T. \end{aligned} \tag{4.7}$$

The kernel can therefore be rewritten as:

$$K(\mathbf{m}_i, \mathbf{m}_j) = \mathbf{m}_i^T (\mathbf{I} - \frac{\xi}{1+\xi}\mathbf{U}\mathbf{U}^T)\mathbf{m}_j. \tag{4.8}$$

We notice in (4.8) that when $\xi = 0$ we recover the standard linear kernel, and more importantly when $\xi = \infty$ we recover exactly the kernel suggested in [6] for performing NAP channel compensation. An advantage of this formulation over NAP is that it does not make a hard decision to completely remove dimensions from the SVM features but instead leaves that decision to the SVM optimization.

It is of practical importance to note that (4.8) can be written as a combination of two kernel matrices, and defining $\mathbf{x}_i = \mathbf{U}^T\mathbf{m}_i$ to be the channel factors:

$$\begin{aligned} K(\mathbf{m}_i, \mathbf{m}_j) &= \mathbf{m}_i^T\mathbf{m}_j - \frac{\xi}{1+\xi}\mathbf{m}_i^T\mathbf{U}\mathbf{U}^T\mathbf{m}_j \\ &= \mathbf{m}_i^T\mathbf{m}_j - \frac{\xi}{1+\xi}\mathbf{x}_i^T\mathbf{x}_j. \end{aligned} \tag{4.9}$$

This allows for a less costly implementation, because the two kernel matrices need not be recomputed for each value of ξ and relatively little computation is required to obtain the second kernel, since the $\mathbf{x}_i$'s are typically low dimensional.

4.2.1 Should All Nuisance be Treated Equally?

As the choice of nuisance subspace gets larger one may find it is more appropriate to handle directions within that subspace unequally, for example we might want to avoid using larger nuisance directions in discrimination more than we would smaller ones. One approach to do this is to reformulate (4.9) to obtain the following kernel:

$$K(\mathbf{m}_i, \mathbf{m}_j) = \mathbf{m}_i^T \mathbf{m}_j - \mathbf{x}_i^T \mathbf{W} \mathbf{x}_j, \tag{4.10}$$

where $\mathbf{W}$ is a diagonal matrix with $[\frac{\xi d_1}{1+\xi d_1}, \frac{\xi d_2}{1+\xi d_2}, ..., \frac{\xi d_N}{1+\xi d_N}]$ on the diagonal.

This resultant kernel can be obtained by replacing $\mathbf{U}$ with $\tilde{\mathbf{U}} = \mathbf{U}\mathbf{D}^{1/4}$ in (4.1), where $\mathbf{D}$ is a positive diagonal matrix whose diagonal elements are $[d_1, d_2, ..., d_N]$. Note, $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{w}$ is no longer an orthogonal projection.

Using $\tilde{\mathbf{U}}$ instead of $\mathbf{U}$ and following the steps outlined in the previous section and using the matrix inversion lemma we obtain the following kernel:

$$\begin{aligned}
K(\mathbf{m}_i, \mathbf{m}_j) &= \mathbf{m}_i^T(\mathbf{I} + \xi\tilde{\mathbf{U}}\mathbf{D}^{1/2}\tilde{\mathbf{U}}^T)^{-1}\mathbf{m}_j \\
&= \mathbf{m}_i^T(\mathbf{I} + \xi\mathbf{U}\mathbf{D}\mathbf{U}^T)^{-1}\mathbf{m}_j \qquad (4.11)\\
&= \mathbf{m}_i^T(\mathbf{I} - \xi\mathbf{U}(\mathbf{D}^{-1} + \xi\mathbf{U}^T\mathbf{U})\mathbf{U}^T)\mathbf{m}_j \\
&= \mathbf{m}_i^T\mathbf{m}_j - \mathbf{m}_i^T\xi\mathbf{U}(\mathbf{D}^{-1} + \xi\mathbf{I})\mathbf{U}^T\mathbf{m}_j \\
&= \mathbf{m}_i^T\mathbf{m}_j - \mathbf{m}_i^T\mathbf{U}\mathbf{W}\mathbf{U}^T\mathbf{m}_j \\
&= \mathbf{m}_i^T\mathbf{m}_j - \mathbf{x}_i^T\mathbf{W}\mathbf{x}_j
\end{aligned}$$

One possible choice, and the one used in our experiments, is to set $\mathbf{D} = \Lambda$, the diagonal matrix whose elements are the eigenvalues (λ_is) corresponding to the columns of $\mathbf{U}$. For that particular choice, the resultant weighting matrix $\mathbf{W}$ in (4.12) is diagonal with the elements $[\frac{\xi\lambda_1}{1+\xi\lambda_1}, \frac{\xi\lambda_2}{1+\xi\lambda_2}, ..., \frac{\xi\lambda_N}{1+\xi\lambda_N}]$ on the diagonal.

4.2.2 Motivating and Extending WCCN

In the previous section we allowed for non-equal weighting of the nuisance subspace, and choosing $\mathbf{D} = \Lambda$ provides us with another way to motivate within class covariance normalization (WCCN) [7]. To do that we begin with equation (4.11) and consider the case where the whole SVM space is considered to contain nuisance information (i.e. $\mathbf{U}\Lambda\mathbf{U}^T$ is full rank).

$$\begin{aligned} K(\mathbf{m}_i, \mathbf{m}_j) &= \mathbf{m}_i^T(\mathbf{I} + \xi\mathbf{U}\Lambda\mathbf{U}^T)^{-1}\mathbf{m}_j \\ &= \mathbf{m}_i^T(\mathbf{I} + \xi\Sigma)^{-1}\mathbf{m}_j, \end{aligned} \tag{4.12}$$

where $\Sigma = \mathbf{U}\Lambda\mathbf{U}^T$ is the eigenvalue decomposition of the within-speaker covariance matrix Σ.

We now examine WCCN, which proposes using the inverse of the intra-speaker covariance matrix $\Sigma = \mathbf{U}\Lambda\mathbf{U}^T$ to weight the kernel inner product:

$$K(\mathbf{m}_i, \mathbf{m}_j) = \mathbf{m}_i^T\Sigma^{-1}\mathbf{m}_j = \mathbf{m}_i^T(\mathbf{U}\Lambda\mathbf{U}^T)^{-1}\mathbf{m}_j. \tag{4.13}$$

However, in practice Σ is ill-conditioned due to noisy estimates and directions of very small nuisance variability, therefore smoothing is applied to the intra-speaker covariance matrix to make inversion possible, and the WCCN suggested kernel becomes:

$$K(\mathbf{m}_i, \mathbf{m}_j) = \mathbf{m}_i^T((1-\theta)\mathbf{I} + \theta\mathbf{U}\Lambda\mathbf{U}^T)^{-1}\mathbf{m}_j \quad 0 \le \theta < 1. \tag{4.14}$$

Comparing (4.14) with (4.12) we see that they are similar. We should, however, mention that when $\mathbf{U}\Lambda\mathbf{U}^T$ spans the full SVM space the ξ (in our implementation) and θ (in the WCCN implementation) no longer set the amount of bias desired, instead they ensure that the kernel does not over-amplify directions with small amounts of nuisance variability.

A concern when applying WCCN is that it requires taking the inverse of a matrix the size of the SVM space. However, considering WCCN in this framework and examining equation (4.12), we realize that by focusing on the directions of greatest variability we can bypass performing the inverse of the within-class covariance matrix. Instead, iterative methods for obtaining the largest eigenvalues and eigenvectors of symmetric matrices can be used [41].

4.3 Using Inter-speaker Variability

Joint factor analysis [42] has been successful in the speaker verification task. Joint factor analysis estimates a “speaker” subspace, that captures good variability and is spanned by the columns of $\mathbf{V}$, and a “channel” subspace, that captures the nuisance and is spanned by the columns of $\mathbf{U}$. A recording $\mathbf{m}_i$ is represented as a linear combination of a contribution from the speaker, $\mathbf{V}\mathbf{y}_i$, one from the channel, $\mathbf{U}\mathbf{x}_i$, and a residual; where $\mathbf{y}_i$ are the speaker factors and $\mathbf{x}_i$ are the channel factors. Recently, promising results have been obtained by using just the speaker factors as features in a SVM speaker verification system. Based on this, we propose a VCSVM formulation similar to the one presented in the previous section to bias the SVM towards mostly using the data present in the inter-speaker variability space.

Assume that the inter-speaker subspace is spanned by a set of V orthonormal vectors (eigenvoices) $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_V\}$, and let $\mathbf{V}$ be the matrix whose columns are these eigenvectors. Let the vector normal to the separating hyperplane be $\mathbf{w}$. Ideally if $\mathbf{V}$ captured all inter-speaker variability, then we would want $\mathbf{w}$ to live in the $\mathbf{V}$ subspace and therefore be orthogonal to its complement, i.e. $\left|\left|(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{w}\right|\right|_2^2 = 0$. Similar to the previous section this requirement can be introduced directly into the primal formulation of the SVM optimization:

$$\begin{gathered}\text{min } J(\mathbf{w}, \epsilon) = ||\mathbf{w}||_2^2/2 + \gamma \left|\left|(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{w}\right|\right|_2^2/2 + C\sum_{i=1}^{k}\epsilon_i \\ \text{subject to } l_i(\mathbf{w}^T\mathbf{m}_i + b) \geq 1 - \epsilon_i \;\;\&\;\; \epsilon_i \geq 0, \quad i = 0, \ldots, k\end{gathered}$$

where $\gamma \geq 0$ is a tunable (on some held out set) parameter that enforces the amount of bias desired. If $\gamma = \infty$ then this formulation becomes similar to just using the speaker factors, and if $\gamma = 0$ then we obtain the standard SVM formulation. Note that, since $\mathbf{I} - \mathbf{V}\mathbf{V}^T$ is a projection into the complement of $\mathbf{V}$ then we can replace it by $\bar{\mathbf{V}}\bar{\mathbf{V}}^T$, where $\bar{\mathbf{V}}$ is a matrix whose columns are the orthonormal eigenvectors that span the complement. With this substitution we obtain a formulation that is almost equivalent to that in (4.1), hence following the recipe in the previous section we see again can push the bias into the kernel of a standard SVM formulation. The kernel in this case is

$$K(\mathbf{m}_i, \mathbf{m}_j) = \mathbf{m}_i^T(\mathbf{I} - \frac{\gamma}{1+\gamma}\bar{\mathbf{V}}\bar{\mathbf{V}}^T)\mathbf{m}_j. \tag{4.15}$$

By substituting back $\bar{\mathbf{V}} = \mathbf{I} - \mathbf{V}\mathbf{V}^T$ we can rewrite (4.15) as:

$$K(\mathbf{m}_i, \mathbf{m}_j) = \mathbf{m}_i^T(\mathbf{I} - \frac{\gamma}{1+\gamma}(\mathbf{I} - \mathbf{V}\mathbf{V}^T))\mathbf{m}_j. \tag{4.16}$$

Note that we do not have to explicitly compute the orthonormal basis $\bar{\mathbf{V}}$, which can be rather large. When $\gamma = \infty$ the kernel becomes an inner product between the speaker factors $\mathbf{y}_i = \mathbf{V}^T\mathbf{m}_i$:

$$K(\mathbf{m}_i, \mathbf{m}_j) = \quad \mathbf{m}_i^T\mathbf{V}\mathbf{V}^T\mathbf{m}_j = \mathbf{y}_i^T\mathbf{y}_j. \tag{4.17}$$

This kernel suggests that when one chooses to perform classification using only the inter-speaker subspace the resultant kernel is just an inner product between the speaker factors.

4.4 Probabilistic Interpretation

In [39], the author makes a connection between the suggested kernel and the probabilistic interpretation of SVMs proposed in [43]. The SVM problem can be thought of as one of maximization of the likelihood of $\mathbf{w}$ given the training data ($\{\mathbf{m}_i, l_i\}$ pairs) by writing it as

$$\max \ l(\mathbf{w}|\{\mathbf{m}_i, l_i\}) = -\mathbf{w}^T\mathbf{w}/2 - C\sum_{i=1}^{k} h(l_i(\mathbf{w}^T\mathbf{m}_i + b)), \tag{4.18}$$

where $h()$ is the hinge loss. In this formulation, the SVM can be though of as just computing the MAP estimate of $\mathbf{w}$ given the training data, where the $\mathbf{w}^T\mathbf{w}$ term is essentially a Gaussian, $N(0, \mathbf{I})$, prior and the second term is the log-likelihood of the training data given $\mathbf{w}$. This Gaussian prior on $\mathbf{w}$ in the standard SVM does not bias the orientation of $\mathbf{w}$ in any direction since the components of $\mathbf{w}$ in the prior are independent. In VCSVM, when we introduce the bias to handle the variability this only affects the first term in (4.18) and therefore changes the prior on $\mathbf{w}$ in the MAP estimation interpretation (we will focus on nuisance variability):

$$\max \ l(\mathbf{w}|\{\mathbf{m}_i, l_i\}) \quad = \quad -\mathbf{w}^T(\mathbf{I} + \xi\mathbf{U}\mathbf{U}^T)\mathbf{w}/2 - C\sum_{i=1}^{k} h(l_i(\mathbf{w}^T\mathbf{m}_i + b)). \tag{4.19}$$

The prior on the MAP estimate of $\mathbf{w}$ is still a Gaussian $N(0, (\mathbf{I} + \xi\mathbf{U}\mathbf{U}^T)^{-1})$ but with its principal components orthogonal to the nuisance subspace and the variance along the principle components set by ξ. Hence, the prior is biasing $\mathbf{w}$ to be orthogonal to the nuisance subspace.

4.5 Experimental Results

We have chosen to demonstrate VCSVM in two scenarios, the first is as an alternative to NAP to handle nuisance in the GSV system presented in [24], and the second to handle nuisance in a system presented in [27] where SVM speaker verification is performed using low-dimensional speaker factors. The goal of this section is not to compare the performance of these two systems, but rather to show that VCSVM is applicable to both. Results on handling inter-speaker variability and all variability will be deferred to future work.

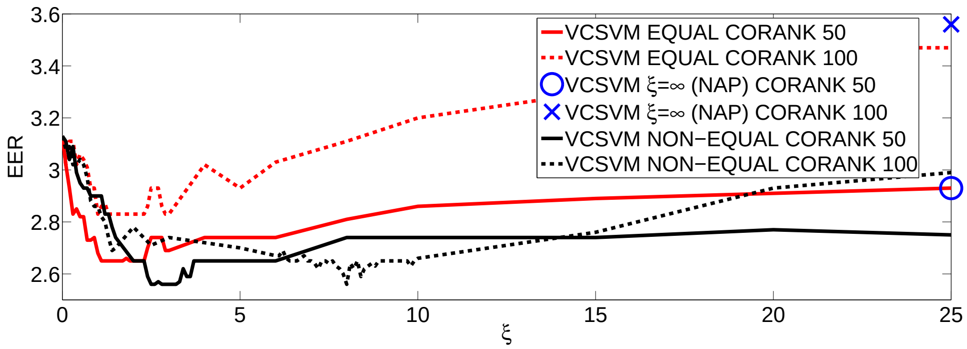

Figure 4-4: Results on English trials of the NIST SRE-Eval 06 core task with speaker factor SVM system: EER vs ξ for equal and non-equal weighting of nuisance subspace, and various subspace sizes.

We begin with the speaker verification system proposed in [27], which represents each recording using a vector of 300 speaker factors from the joint factor analysis system in [33]. The speaker factor vectors, of length 300, are normalized to have unit L_2-norm and used as features in a SVM speaker verification system. Figure 4-4 shows how the equal error rate (EER) changes as a function of ξ on our development set, the English trials of the NIST SRE-Eval 06 core task, for nuisance subspaces, spanned by the eigenvectors of the within-class covariance matrix, of dimension (corank) 50 and 100 dimensional nuisance subspaces when equal and non-equal weighting of the nuisance dimensions are used. The figure shows that non-equal weighting of the nuisance directions yields more favorable results than equal weighting. It also shows that VCSVM allows for nuisance compensation in such a small space, while NAP performs poorly since it completely removes the estimated nuisance dimensions which are a large percentage of the total dimensionality. Based on the development results we choose $\xi = 3$ and a corank of 50 for the VCSVM system and present results on all trials of the Eval 08 core task in Figure 2-1 (a).

Next, we present the performance of VCSVM using a GSV system [24] with 512 mixture

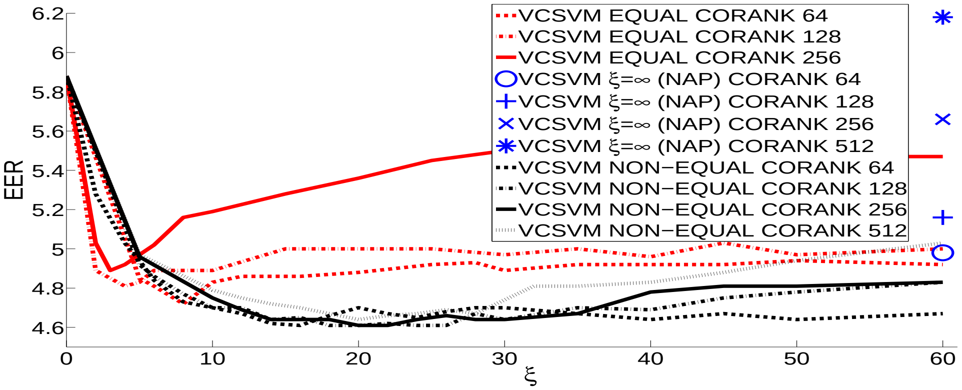

Figure 4-5: Results on all trials of the NIST SRE-Eval 06 core task with GSV system: EER vs ξ for equal and non-equal weighting of nuisance subspace, and various subspace sizes.

GMMs and 38 dimensional, 19 cepstral and deltas, RASTA compensated feature vectors. Figures 4-6 &4-7 present results on the development set, all trials of the NIST SRE-Eval 06 core condition. They show how the EER changes as a function of ξ, corank, and whether equal or non-equal weighting was used. Again this shows that non-equal weighting of the nuisance directions is preferable over equal weighting. It also shows that non-equally weighted VCSVM is fairly stable with regards to varying ξ and the corank, which is not the case with NAP. Based on these development results we compare, in Figure 2-1 (b), no nuisance compensation to the best-performing NAP system, with a corank of 64, and the best VCSVM system, with $\xi = 22$ and corank of 256. We see that even in a large dimensional space such as this, it is preferable to not completely remove the nuisance subspace.

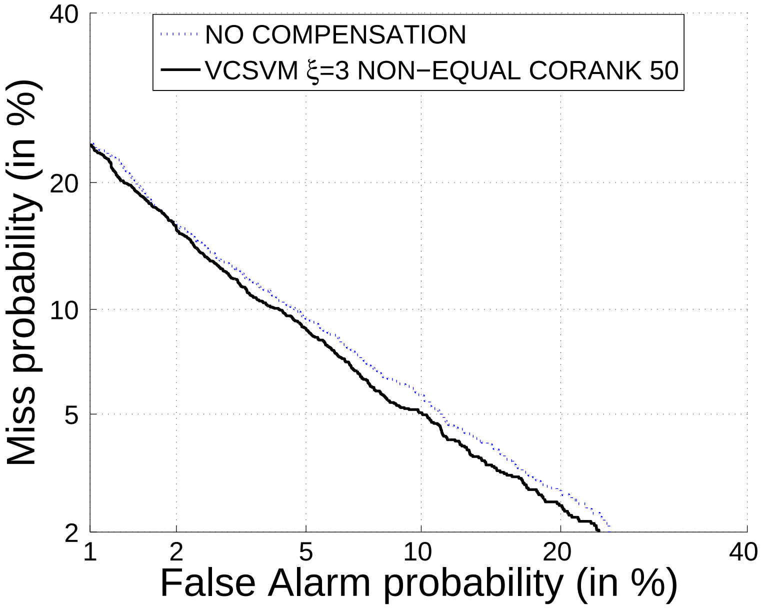

Figure 4-6: DET plot of the speaker factor SVM system on all trials of the NIST SRE 08 core task.

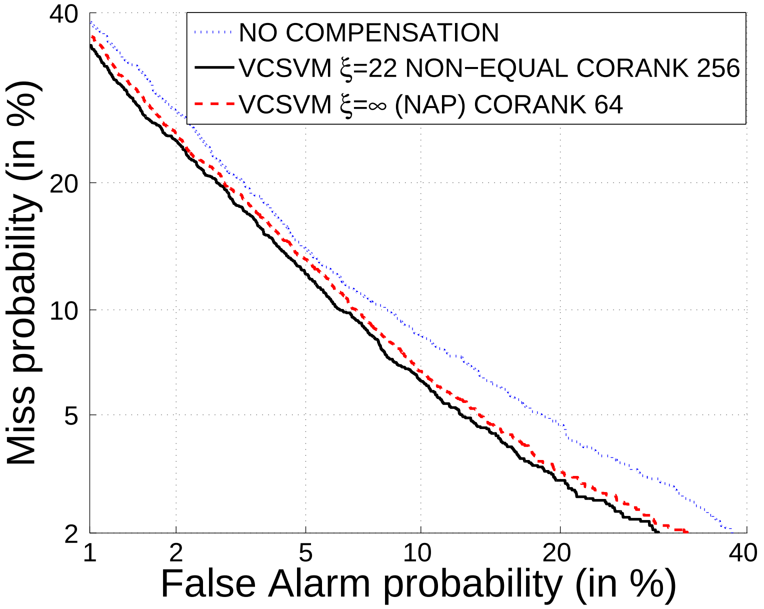

Figure 4-7: DET plot of the GSV system on all trials of the NIST SRE 08 core task.

4.6 Discussion

This chapter presents variability-compensated SVM (VCSVM), a method for handling both good and bad variability directly in the SVM optimization. This is accomplished by introducing into the minimization a regularized penalty, which biases the classifier to avoid nuisance directions and use directions of inter-speaker variability.

With regard to nuisance compensation our method encompasses and extends both NAP and WCCN. An advantage of our proposed method over NAP, is that it does not make a hard decision on removing nuisance directions, rather it decides according to performance on a held out set. Also, it allows for unequal weighting of the estimated nuisance directions, e.g., according to their associated eigenvalues which leads to improved performance over NAP, increased robustness with regards to the size of the estimated nuisance subspace, and successful nuisance compensation in small SVM spaces. This work also provides another motivation for WCCN and extends it to better handle large vector spaces.

In this work, we have focused on nuisance compensation to present the framework and highlight its merits, however, we have not fully explored how to best incorporate speaker variability into the framework and handle both nuisance and speaker variability simultaneously. These questions provide directions for future work.

Chapter 5

Speaker Comparison with Inner Product Decision Functions

In Section 2.2.3, we presented the GSV kernel for SVM speaker verification, a popular method in the literature, which consists of an inner product between mean supervectors of adapted GMMs. The GSV is one way to compare speech recordings with kernel functions, however, this has been a common theme in the speaker verification SVM literature resulting in several other kernels [35, 31, 32]. The space defined by the kernel is then compensated to eliminate nuisances using methods such as NAP and WCCN, Section 2.2.4.

A recent trend in the literature has been to move towards a more linear geometric view for non-SVM systems. Joint factor analysis (JFA), Section 2.2.5, uses a Bayesian approach to compensate GMMs representing recordings using linear subspaces. Also, comparison of recordings via inner products between the compensated GMM parameters, obtained via JFA, is presented in [44]. These approaches have introduced many new ideas and perform well in speaker comparison tasks.

An unrealized effort is to bridge the gap between SVMs and some of the new proposed GMM methods. One difficulty is that most SVM kernel functions in speaker comparison satisfy the Mercer condition. This restricts the scope of investigation of potential comparison strategies for two speaker recordings. Therefore, in this chapter, we introduce the idea of inner product discriminant functions (IPDFs).

IPDFs are based upon the same basic operations as SVM kernel functions with some relaxation in structure. First, we map input recordings to vectors of fixed dimension. Second, we *compensate* the input feature vectors. Typically, this compensation takes the form of a linear transform. Third, we *compare* two compensated vectors with an inner product. The resulting comparison function is then used in an application specific way.

The focus of our initial investigations of the IPDF structure are the following. First, we show that many of the common techniques such as factor analysis, nuisance projection, and various types of scoring can be placed in the framework. Second, we systematically describe the various inner product and compensation techniques used in the literature. Third, we propose new inner products and compensation. Finally, we explore the space of possible combinations of techniques and demonstrate several novel methods that are computationally efficient and produce excellent error rates.

The outline of the chapter is as follows. In Section 5.1, we describe the general setup for speaker comparison using GMMs. In Section 5.2, we introduce the IPDF framework. Section 5.3 explores inner products for the IPDF framework. Section 5.4 looks at methods for compensating for variability. In Section 5.5, we perform experiments on the NIST 2006 speaker recognition evaluation and explore different combinations of IPDF comparisons and compensations.

5.1 Speaker Comparison

A standard distribution used for text-independent speaker recognition is the Gaussian mixture model [2],

$$g(\mathbf{r}) = \sum_{i=1}^{M} \lambda_i \mathcal{N}(\mathbf{r}|\mathbf{m}_i, \Sigma_i). \tag{5.1}$$

Feature vectors are typically cepstral coefficients with associated smoothed first- and second-order derivatives.

We map a sequence of feature vectors, $\mathbf{r}^{\alpha}_{1-N_\alpha} = \{\mathbf{r}^{\alpha}_1, ..., \mathbf{r}^{\alpha}_{N_\alpha}\}$, from a recording $\mathbf{R}_\alpha$ to a GMM by adapting a GMM universal background model (UBM). For the purpose of this chapter, we will assume only the mixture weights, λ_i, and means, $\mathbf{m}_i$, in (5.1) are adapted. Adaptation of the means is performed with standard relevance MAP, refer to Appendix A.3. We estimate the mixture weights using the standard ML estimate. The adaptation yields new parameters which we stack into a parameter vector, $\mathbf{a}_\alpha$, where

$$\begin{aligned} \mathbf{a}_\alpha &= \begin{bmatrix} \boldsymbol{\lambda}^T_\alpha & \mathbf{m}^T_\alpha \end{bmatrix}^T && (5.2) \\ &= \begin{bmatrix} \lambda_{x,1} & \cdots & \lambda_{x,N} & \mathbf{m}^T_{x,1} & \cdots & \mathbf{m}^T_{x,N} \end{bmatrix}^T . && (5.3) \end{aligned}$$

In speaker comparison, the problem is to compare two sequences of feature vectors, e.g., $\mathbf{r}_1^{N_\alpha}$ and $\mathbf{y}_1^{N_\beta}$. To compare these two sequences, we adapt a GMM UBM to produce two sets of parameter vectors, $\mathbf{a}_\alpha$ and $\mathbf{a}_\beta$, as in (5.2). The goal of our speaker comparison process can now be recast as a function that compares the two parameter vectors, $s(\mathbf{R}_\alpha, \mathbf{R}_\beta) = C(\mathbf{a}_\alpha, \mathbf{a}_\beta)$, and produces a value that reflects the similarity of the speakers. Initial work in this area was performed using kernels from support vector machines [32, 45, 35], but we expand the scope to other types of discriminant functions.

5.2 Inner Product Discriminant Functions

The basic framework we propose for speaker comparison functions is composed of two parts—compensation and comparison. For compensation, the parameter vectors generated by adaptation in (5.2) can be transformed to remove nuisances or projected onto a speaker subspace. The second part of our framework is comparison. For the comparison of parameter vectors, we will consider natural distances that result in inner products between parameter vectors.

We propose the following inner product discriminant function (IPDF) framework for exploring speaker comparison,

$$C(\mathbf{a}_\alpha, \mathbf{a}_\beta) = (L_\alpha \mathbf{a}_\alpha)^T D_{\alpha,\beta}^2 (L_\beta \mathbf{a}_\beta) \tag{5.4}$$

where L_α, L_β are linear transforms and potentially dependent on $\boldsymbol{\lambda}_\alpha$ and/or $\boldsymbol{\lambda}_\beta$. The matrix D is positive definite, usually diagonal, and possibly dependent on $\boldsymbol{\lambda}_\alpha$ and/or $\boldsymbol{\lambda}_\beta$. Note, we also consider simple combinations of IPDFs to be in our framework—e.g., positively-weighted sums of IPDFs.

Several questions from this framework are: 1) what inner product gives the best speaker comparison performance, 2) what compensation strategy works best, 3) what tradeoffs can be made between accuracy and computational cost, and 4) how do the compensation and the inner product interact. We explore theoretical and experimental answers to these questions in the following sections.

5.3 Inner Products for IPDFs

In general, an inner product of the parameters should be based on a distance arising from a statistical comparison. We derive three straightforward methods in this section. We also relate some other methods, without being exhaustive, that fall in this framework that have been described in detail in the literature.

5.3.1 Approximate KL Comparison (C_{KL})

A straightforward strategy for comparing the GMM parameter vectors is to use an approximate form of the KL divergence applied to the induced GMM models. This strategy was used in [35] successfully with an approximation based on the log-sum inequality; i.e., for the GMMs, g_α and g_β, with parameters $\mathbf{a}_\alpha$ and $\mathbf{a}_\beta$,

$$D(g_\alpha \| g_\beta) \leq \sum_{i=1}^{M} \lambda_{x,i} D\left(\mathcal{N}(\cdot; \mathbf{m}_{x,i}, \Sigma_i) \| \mathcal{N}(\cdot; \mathbf{m}_{y,i}, \Sigma_i)\right). \tag{5.5}$$

Here, $D(\cdot\|\cdot)$ is the KL divergence, and Σ_i is from the UBM.

By symmetrizing (5.5) and substituting in the KL divergence between two Gaussian distributions, we obtain a distance, d_s, which upper bounds the symmetric KL divergence,

$$d_s(\mathbf{a}_\alpha, \mathbf{a}_\beta) = D_s(\boldsymbol{\lambda}_\alpha \| \boldsymbol{\lambda}_\beta) + \sum_{i=1}^{M} (0.5\lambda_{x,i} + 0.5\lambda_{y,i})(\mathbf{m}_{x,i} - \mathbf{m}_{y,i})^T \Sigma_i^{-1} (\mathbf{m}_{x,i} - \mathbf{m}_{y,i}). \tag{5.6}$$

We focus on the second term in (5.6) for this chapter, but note that the first term could also be converted to a comparison function on the mixture weights. Using polarization on the second term, we obtain the inner product

$$C_{\mathrm{KL}}(\mathbf{a}_\alpha, \mathbf{a}_\beta) = \sum_{i=1}^{M} (0.5\lambda_{x,i} + 0.5\lambda_{y,i}) \mathbf{m}_{x,i}^T \Sigma_i^{-1} \mathbf{m}_{y,i}. \tag{5.7}$$

Note that (5.7) can also be expressed more compactly as

$$C_{\mathrm{KL}}(\mathbf{a}_\alpha, \mathbf{a}_\beta) = \mathbf{m}_\alpha^T \left((0.5\boldsymbol{\lambda}_\alpha + 0.5\boldsymbol{\lambda}_\beta) \otimes I_n\right) \Sigma^{-1} \mathbf{m}_\beta \tag{5.8}$$

where Σ is the block matrix with the Σ_i on the diagonal, n is the feature vector dimension, and $\otimes$ is the Kronecker product [46]. Note that the non-symmetric form of the KL distance in (5.5) would result in the average mixture weights in (5.8) being replaced by $\boldsymbol{\lambda}_\alpha$. Also, note that shifting the means by the UBM will not affect the distance in (5.6), so we can replace means in (5.8) by the UBM centered means.

5.3.2 GLDS kernel (C_{GLDS})

An alternate inner product approach is to use generalized linear discriminants and the corresponding kernel [32]. The overall structure of this GLDS kernel is as follows:

A per feature vector expansion function is defined as

$$\mathbf{b}(\mathbf{r}_i) = \begin{bmatrix} b_1(\mathbf{r}_i) & \cdots & b_m(\mathbf{r}_i) \end{bmatrix}^T . \tag{5.9}$$

The mapping between an input sequence, $\mathbf{r}^{\alpha}_{1-N_\alpha}$ is then defined as

$$\mathbf{r}^{\alpha}_{1-N_\alpha} \mapsto \mathbf{b}_\alpha = \frac{1}{N_\alpha} \sum_{i=1}^{N_\alpha} \mathbf{b}(\mathbf{r}_i). \tag{5.10}$$

The corresponding kernel between two sequences is then

$$K_{\text{GLDS}}(\mathbf{r}^{\alpha}_{1-N_\alpha}, \mathbf{r}^{\beta}_{1-N_\beta}) = \mathbf{b}_\alpha^T \Gamma^{-1} \mathbf{b}_\beta \tag{5.11}$$

where

$$\Gamma = \frac{1}{N_B} \sum_{i=1}^{N_B} \mathbf{b}(\mathbf{R}_i^B) \mathbf{b}(\mathbf{R}_i^B)^T, \tag{5.12}$$

and $\mathbf{r}^{B}_{1-N_B}$ is a large set of feature vectors which is representative of the speaker population, i.e. an aggregate of features from a large number of recordings.

In the context of a GMM UBM, we can define an expansion as follows

$$\mathbf{b}(\mathbf{r}_i) = \begin{bmatrix} p(1|\mathbf{r}_i)(\mathbf{r}_i - \mathbf{m}_{UBM,1})^T & \cdots & p(N|\mathbf{r}_i)(\mathbf{r}_i - \mathbf{m}_{UBM,N})^T \end{bmatrix}^T \tag{5.13}$$

where $p(j|\mathbf{r}_i)$ is the posterior probability of mixture component j given $\mathbf{r}_i$, and $\mathbf{m}_{UBM,j}$ is from a UBM. Using (5.13) in (5.10), we see that

$$\mathbf{b}_\alpha = (\boldsymbol{\lambda}_\alpha \otimes I_n)(\mathbf{m}_\alpha - \mathbf{m}_{UBM}) \text{ and } \mathbf{b}_\beta = (\boldsymbol{\lambda}_\beta \otimes I_n)(\mathbf{m}_\beta - \mathbf{m}_{UBM}) \tag{5.14}$$

where $\mathbf{m}_{UBM}$ is the stacked means of the UBM. Thus, the GLDS kernel inner product is

$$C_{\text{GLDS}}(\mathbf{a}_\alpha, \mathbf{a}_\beta) = (\mathbf{m}_\alpha - \mathbf{m}_{UBM})^T (\boldsymbol{\lambda}_\alpha \otimes I_n) \Gamma^{-1} (\boldsymbol{\lambda}_\beta \otimes I_n)(\mathbf{m}_\beta - \mathbf{m}_{UBM}). \tag{5.15}$$

Note that Γ in (5.12) is almost the UBM covariance matrix, but is not quite the same because of a squaring of the $p(j|\mathbf{R}_i^B)$ in the diagonal. As is commonly assumed, we will consider a diagonal approximation of Γ, see [32].

5.3.3 Gaussian-Distributed Vectors

A common assumption in the factor analysis literature [8] is that the parameter vector $\mathbf{m}_x$ as x varies has a Gaussian distribution. If we assume a single covariance for the entire space, then the resulting likelihood ratio test between two Gaussian distributions results in a linear discriminant [47].

More formally, suppose that we have a distribution of the features of $\mathbf{R}_\alpha$ with mean $\mathbf{m}_\alpha$ and we are trying to distinguish from a distribution with the UBM mean $\mathbf{m}_{UBM}$, then the discriminant function is [47],

$$h(\mathbf{r}) = (\mathbf{m}_\alpha - \mathbf{m}_{UBM})^T \Upsilon^{-1} (\mathbf{r} - \mathbf{m}_{UBM}) + c_\alpha \tag{5.16}$$

where c_α is a constant that depends on $\mathbf{m}_\alpha$, and Υ is the covariance in the parameter vector space. We will assume that the comparison function can be normalized (e.g., by Z-norm [2]), so that c_α can be dropped. We now apply the discriminant function to another mean vector, $\mathbf{m}_\beta$, and obtain the following comparison function

$$C_G(\mathbf{a}_\alpha, \mathbf{a}_\beta) = (\mathbf{m}_\alpha - \mathbf{m}_{UBM})^T \Upsilon^{-1} (\mathbf{m}_\beta - \mathbf{m}_{UBM}). \tag{5.17}$$

5.3.4 Other Methods

Several other methods are possible for comparing the parameter vectors that arise either from ad-hoc methods or from work in the literature. We describe a few of these in this section.

Geometric Mean Comparison (C_{GM}): A simple symmetric function that is similar to the KL (5.8) and GLDS (5.15) comparison functions is arrived at by replacing the arithmetic mean in C_{KL} by a geometric mean. The resulting kernel is

$$C_{GM}(\mathbf{a}_\alpha, \mathbf{a}_\beta) = (\mathbf{m}_\alpha - \mathbf{m}_{UBM})^T (\boldsymbol{\lambda}_\alpha^{1/2} \otimes I_n) \Sigma^{-1} (\boldsymbol{\lambda}_\beta^{1/2} \otimes I_n)(\mathbf{m}_\beta - \mathbf{m}_{UBM}) \tag{5.18}$$

where Σ is the block diagonal UBM covariances.

Fisher Kernel (C_F): The Fisher kernel specialized to the UBM case has several forms [31]. The main variations are the choice of covariance in the inner product and the choice of normalization of the gradient term. We took the best performing configuration for this chapter—we normalize the gradient by the number of frames which results in a mixture weight scaling of the gradient. We also use a diagonal data-trained covariance term. The resulting comparison function is

$$C_F(\mathbf{a}_\alpha, \mathbf{a}_\beta) = \left[(\boldsymbol{\lambda}_\alpha \otimes I_n)\Sigma^{-1}(\mathbf{m}_\alpha - \mathbf{m}_{UBM})\right]^T \Phi^{-1} \left[(\boldsymbol{\lambda}_\beta \otimes I_n)\Sigma^{-1}(\mathbf{m}_\beta - \mathbf{m}_{UBM})\right] \quad (5.19)$$

where Φ is a diagonal matrix acting as a variance normalizer.

Linearized Q-function (C_Q): Another form of inner product may be derived from the linear Q-scoring shown in [44]. In this case, the scoring is given as $(\mathbf{m}_{\text{TGT}} - \mathbf{m}_{UBM})^T\Sigma^{-1}(\mathbf{F} - \mathbf{N}\mathbf{m}_{UBM})$ where $\mathbf{N}$ and $\mathbf{F}$ are the zeroth and first order sufficient statistics of a test recording, $\mathbf{m}_{UBM}$ is the UBM means, $\mathbf{m}_{\text{TGT}}$ is the mean of the target model, and Σ is the block diagonal UBM covariances. A close approximation of this function can be made by using a small relevance factor in MAP adaptation of the means to obtain the following comparison function

$$C_Q(\mathbf{a}_\alpha, \mathbf{a}_\beta) = (\mathbf{m}_\alpha - \mathbf{m}_{UBM})^T\Sigma^{-1}(\boldsymbol{\lambda}_\beta \otimes I_n)(\mathbf{m}_\beta - \mathbf{m}_{UBM}). \quad (5.20)$$

Note that if we symmetrize C_Q, this gives us C_{KL}; this analysis ignores for a moment that in [44], compensation is also asymmetric.

KL Kernel (K_{KL}): By assuming the mixture weights are constant and equal to the UBM mixture in the comparison function C_{KL} (5.7), we obtain the KL kernel,

$$K_{KL}(\mathbf{a}_\alpha, \mathbf{a}_\beta) = (\mathbf{m}_\alpha - \mathbf{m}_{UBM})^T \, (\boldsymbol{\lambda} \otimes I_n) \, \Sigma^{-1}(\mathbf{m}_\beta - \mathbf{m}_{UBM}) \quad (5.21)$$

where $\boldsymbol{\lambda}$ are the UBM mixture weights. This kernel has been used extensively in SVM speaker recognition [35].

An analysis of the different inner products in the preceding sections shows that many of the methods presented in the literature have a similar form, but are interestingly derived with quite disparate techniques. Our goal in the experimental section is to understand how these comparison function perform and how they interact with compensation.

5.4 Compensation in IPDFs

Our next task is to explore compensation methods for IPDFs. Our focus will be on subspace-based methods. With these methods, the fundamental assumption is that either speakers and/or nuisances are confined to a small subspace in the parameter vector space. The problem is to use this knowledge to produce a higher signal (speaker) to noise (nuisance) representation of the speaker.

Standard notation is to use U to represent the nuisance subspace and to have V represent the speaker subspace. Our goal in this section is to recast many of the methods in the literature in a standard framework with oblique and orthogonal projections.

To make a cohesive presentation, we introduce some notation. We define an orthogonal projection with respect to a metric, $P_{U,D}$, where D and U are full rank matrices as

$$P_{U,D} = U(U^T D^2 U)^{-1} U^T D^2 \tag{5.22}$$

where DU is a linearly independent set, and the metric is $\|x - y\|_D = \|Dx - Dy\|_2$. The process of projection, e.g. $y = P_{U,D}b$, is equivalent to solving the least-squares problem, $\hat{x} = \operatorname{argmin}_x \|Ux - b\|_D$ and letting $y = U\hat{x}$. For convenience, we also define the projection onto the orthogonal complement of U, $U^{\perp}$, as $Q_{U,D} = P_{U^{\perp},D} = I - P_{U,D}$. Note that we can regularize the projection $P_{U,D}$ by adding a diagonal term to the inverse in (5.22); the resulting operation remains linear but is no longer a projection.

We also define the oblique projection onto V with null space $U + (U + V)^{\perp}$ and metric induced by D. Let QR be the (skinny) QR decomposition of the matrix $\begin{bmatrix} UV \end{bmatrix}$ in the D norm (i.e., $Q^T D^2 Q = I$), and Q_V be the columns corresponding to V in the matrix Q. Then, the oblique (non-orthogonal) projection onto V is

$$O_{V,U,D} = V(Q_V^T D^2 V)^{-1} Q_V^T D^2. \tag{5.23}$$

The use of projections in our development will add geometric understanding to the process of compensation.

5.4.1 Nuisance Attribute Projection (NAP)

A framework for eliminating nuisances in the parameter vector based on projection was shown in [35]. The basic idea is to assume that nuisances are confined to a small subspace and can be removed via an orthogonal projection, $\mathbf{m}_x \mapsto Q_{U,D}\mathbf{m}_x$. One justification for using subspaces comes from the perspective that channel classification can be performed with inner products along one-dimensional subspaces. Therefore, the projection removes channel specific directions from the parameter space.

The NAP projection uses the metric induced by a kernel in an SVM. For the GMM context, the standard kernel used is the approximate KL comparison (5.8) [35]. We note that since D is known *a priori* to speaker comparison, we can orthonormalize the matrix DU and apply the projection as a matrix multiply. The resulting projection has $D = \left(\boldsymbol{\lambda}^{1/2} \otimes I_n\right) \Sigma^{-1/2}$.

5.4.2 Factor Analysis and Joint Factor Analysis

The joint factor analysis (JFA) model assumes that the mean parameter vector can be expressed as

$$\mathbf{m}_{\text{s,sess}} = \mathbf{m} + U\mathbf{x} + V\mathbf{y} \tag{5.24}$$

where $\mathbf{m}_{\text{s,sess}}$ is the speaker and session-dependent mean parameter vector, U and V are matrices with small rank, and $\mathbf{m}$ is typically the UBM. Note that for this section, we will use the standard variables for factor analysis, $\mathbf{x}$ and $\mathbf{y}$, even though they conflict with our earlier development. The goal of joint factor analysis is to find solutions to the latent variables $\mathbf{x}$ and $\mathbf{y}$ given training data. In (5.24), the matrix U represents a nuisance subspace, and V represents a speaker subspace. Existing work on this approach for speaker recognition uses both maximum likelihood (ML) estimates and MAP estimates of $\mathbf{x}$ and $\mathbf{y}$ [48, 8]. In the latter case, a Gaussian prior with zero mean and diagonal covariance for $\mathbf{x}$ and $\mathbf{y}$ is assumed. For our work, we focus on the ML estimates [48] of $\mathbf{x}$ and $\mathbf{y}$ in (5.24), since we did not observe substantially different performance from MAP estimates in our experiments.

Another form of modeling that we will consider is factor analysis (FA). In this case, the term $V\mathbf{y}$ is replaced by a constant vector representing the true speaker model, $\mathbf{m}_s$; the goal is then to estimate $\mathbf{x}$. Typically, as a simplification, $\mathbf{m}_s$ is assumed to be zero when calculating sufficient statistics for estimation of $\mathbf{x}$ [49].

The solution to both JFA and FA can be unified. For the JFA problem, if we stack the matrices $[UV]$, then the problem reverts to the FA problem. Therefore, we initially study the FA problem. Note that we also restrict our work to only one EM iteration of the estimation of the factors, since this strategy works well in practice.

The standard ML solution to FA [48] for one EM iteration can be written as:

$$\left[U^T \Sigma^{-1} (\mathbf{N} \otimes I_n) U\right] \mathbf{x} = U^T \Sigma^{-1} \left[\mathbf{F} - (\mathbf{N} \otimes I_n)\mathbf{m}\right] \tag{5.25}$$

where $\mathbf{F}$ is the vector of first order sufficient statistics, and $\mathbf{N}$ is the diagonal matrix of zeroth order statistics (expected counts). The sufficient statistics are obtained from the UBM applied to an input set of feature vectors. We first let $N_{\text{t}} = \sum_{i=1}^{N} N_i$ and multiply both sides of (5.25) by $1/N_t$. Now we use relevance MAP with a small relevance factor and $\mathbf{F}$ and $\mathbf{N}$ to obtain $\mathbf{m}_s$; i.e., both $\mathbf{m}_s - \mathbf{m}$ and $\mathbf{F} - (\mathbf{N} \otimes I_n)\mathbf{m}$ will be nearly zero in the entries corresponding to small N_i. We obtain

$$\left[U^T \Sigma^{-1} (\boldsymbol{\lambda}_s \otimes I_n) U\right] \mathbf{x} = U^T \Sigma^{-1} \left(\boldsymbol{\lambda}_s \otimes I_n\right) \left[\mathbf{m}_s - \mathbf{m}\right] \tag{5.26}$$

where $\boldsymbol{\lambda}_s$ is the speaker dependent mixture weights. We note that (5.26) are the normal equations for the least-squares problem, $\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \|U\mathbf{x} - (\mathbf{m}_s - \mathbf{m})\|_D$ where D is given below in (5.28). This solution is not unexpected since ML estimates commonly lead to least-squares problems with GMM distributed data [50].

Once the solution to (5.26) is obtained, the resulting $U\mathbf{x}$ is subtracted from an estimate of the speaker mean, $\mathbf{m}_s$ to obtain the compensated mean. If we assume that $\mathbf{m}_s$ is obtained by a relevance map adaptation from the statistics $\mathbf{F}$ and $\mathbf{N}$ with a small relevance factor, then the FA process is well approximated by

$$\mathbf{m}_s \mapsto Q_{U,D} \mathbf{m}_s \tag{5.27}$$

where

$$D = \left(\boldsymbol{\lambda}_s^{1/2} \otimes I_n\right) \Sigma^{-1/2}. \tag{5.28}$$

JFA becomes an extension of the FA process we have demonstrated. One first projects onto the stacked UV space. Then another projection is performed to eliminate the U component of variability. This can be expressed as a single oblique projection; i.e., the JFA process is

$$\mathbf{m}_s \mapsto O_{V,U,I} P_{[UV],D} \mathbf{m}_s = O_{V,U,D} \mathbf{m}_s. \tag{5.29}$$

5.4.3 Comments and Analysis

Several comments should be made on compensation schemes and their use in speaker comparison. First, although NAP and ML FA (5.27) were derived in substantially different ways, they are essentially the same operation, an orthogonal projection. The main difference is in the choice of metrics under which they were originally proposed. For NAP, the metric depends on the UBM only, and for FA it is recording and UBM dependent.

A second observation is that the JFA oblique projection onto V has substantially different properties than a standard orthogonal projection. When JFA is used in speaker recognition [8, 44], typically JFA is performed in training, but the test recording is compensated only with FA. In our notation, applying JFA with linear scoring [44] gives

$$C_Q(O_{V,U,D_{TGT}}\mathbf{m}_{TGT}, Q_{U,D_{TST}}\mathbf{m}_{TST}) \tag{5.30}$$

where $\mathbf{m}_{TGT}$ and $\mathbf{m}_{TST}$ are the mean parameter vectors estimated from the target and test recordings of a trial, respectively; also, $D_{TGT} = (\boldsymbol{\lambda}_{TGT}^{1/2} \otimes I_n)\Sigma^{-1/2}$ and $D_{TST} = (\boldsymbol{\lambda}_{TST}^{1/2} \otimes I_n)\Sigma^{-1/2}$. Our goal in the experiments section is to disentangle and understand some of the properties of scoring methods such as (5.30). What is significant in this process—mismatched train/test compensation, data-dependent metrics, or asymmetric scoring?

A final note is that training the subspaces for the various projections *optimally* is not a process that is completely understood. One difficulty is that the metric used for the inner product may not correspond to the metric for compensation. As a baseline, we used the same subspace for all comparison functions. The subspace was obtained with an ML style procedure for training subspaces similar to [50] but specialized to the factor analysis problem as in [8].

5.5 Speaker Comparison Experiments

Experiments were performed on the NIST 2006 speaker recognition evaluation (SRE) data set. Enrollment/verification methodology and the evaluation criteria, equal error rate (EER) and minDCF, were based on the NIST SRE evaluation plan [51]. The main focus of our efforts was the one conversation enroll, one conversation verification task for telephone recorded speech. T-Norm models and Z-Norm [12] speech recordings were drawn from the NIST 2004 SRE corpus. Results were obtained for both the English only task (Eng) and for all trials (All) which includes speakers that enroll/verify in different languages.

Feature extraction was performed using HTK [52] with 20 MFCC coefficients, deltas, and acceleration coefficients for a total of 60 features. A GMM UBM with 512 mixture components was trained using data from NIST SRE 2004 and from Switchboard corpora. The dimension of the nuisance subspace, U, was fixed at 100; the dimension of the speaker space, V, was fixed at 300.

Results are in Table 5.1. In the table, we use the following notation,

$$D_{\mathrm{UBM}} = \left(\boldsymbol{\lambda}^{1/2} \otimes I_n\right)\Sigma^{-1/2},\ D_{TGT} = \left(\boldsymbol{\lambda}_{TGT}^{1/2} \otimes I_n\right)\Sigma^{-1/2},\ D_{TST} = \left(\boldsymbol{\lambda}_{TST}^{1/2} \otimes I_n\right)\Sigma^{-1/2} \tag{5.31}$$

Table 5.1: A comparison of baseline systems and different IPDF implementations

Comparison Function	Enroll Comp.	Verify Comp.	EER All (%)	minDCF All (×100)	EER Eng (%)	minDCF Eng (×100)
Baseline SVM	$Q_{U,D_{\mathrm{UBM}}}$	$Q_{U,D_{\mathrm{UBM}}}$	3.82	1.82	2.62	1.17
Baseline JFA, C_Q	$O_{V,U,D_{TGT}}$	$Q_{U,D_{TST}}$	3.07	1.57	2.11	1.23
C_{KL}	$O_{V,U,D_{TGT}}$	$Q_{U,D_{TST}}$	3.21	1.70	2.32	1.32
C_{KL}	$O_{V,U,D_{TGT}}$	$O_{V,U,D_{TST}}$	8.73	5.06	8.06	4.45
C_{KL}	$Q_{U,D_{TGT}}$	$Q_{U,D_{TST}}$	2.93	1.55	1.89	0.93
C_{KL}	$Q_{U,D_{\mathrm{UBM}}}$	$Q_{U,D_{\mathrm{UBM}}}$	3.03	1.55	1.92	0.95
C_{KL}	$I - O_{U,V,D_{TGT}}$	$I - O_{U,V,D_{TST}}$	7.10	3.60	6.49	3.13
C_{GM}	$Q_{U,D_{TGT}}$	$Q_{U,D_{TST}}$	2.90	1.59	1.73	0.98
C_{GM}	$Q_{U,D_{\mathrm{UBM}}}$	$Q_{U,D_{\mathrm{UBM}}}$	3.01	1.66	1.89	1.05
C_{GM}	$Q_{U,D_{\mathrm{UBM}}}$	I	3.95	1.93	2.76	1.26
K_{KL}	$Q_{U,D_{\mathrm{UBM}}}$	$Q_{U,D_{\mathrm{UBM}}}$	4.95	2.46	3.73	1.75
K_{KL}	$Q_{U,D_{TGT}}$	$Q_{U,D_{TST}}$	5.52	2.85	4.43	2.15
C_{GLDS}	Q_{U,D_L}	Q_{U,D_L}	3.60	1.93	2.27	1.23
C_G	Q_{U,D_G}	Q_{U,D_G}	5.07	2.52	3.89	1.87
C_F	Q_{U,D_F}	Q_{U,D_F}	3.56	1.89	2.22	1.12

where $\boldsymbol{\lambda}$ are the UBM mixture weights, $\boldsymbol{\lambda}_{TGT}$ are the mixture weights estimated from the enrollment recording, and $\boldsymbol{\lambda}_{TST}$ are the mixture weights estimated from the verification recording. We also use the notation D_L, D_G, and D_F to denote the parameters of the metric for the GLDS, Gaussian, and Fisher comparison functions from Sections 5.3.2, 5.3.3, and 5.3.4, respectively.

An analysis of the results in Table 5.1 shows several trends. First, the performance of the best IPDF configurations is as good or better than the state of the art SVM and JFA implementations. Second, the compensation method that dominates good performance is an orthogonal complement of the nuisance subspace, $Q_{U,D}$. Combining a nuisance projection with an oblique projection is fine, but using only oblique projections onto V gives high error rates. A third observation is that comparison functions whose metrics incorporate $\boldsymbol{\lambda}_{TGT}$ and $\boldsymbol{\lambda}_{TST}$ perform significantly better than ones with fixed $\boldsymbol{\lambda}$ from the UBM. In terms of best performance, C_{KL}, C_Q, and C_{GM} perform similarly. For example, the 95% confidence interval for 2.90% EER is $[2.6, 3.3]\%$.

We also observe that a nuisance projection with fixed D_{UBM} gives similar performance to a projection involving a "variable" metric, D_i. This property is fortuitous since a fixed projection can be precomputed and stored and involves significantly reduced computation. Table 5.2 shows a comparison of error rates and compute times normalized by a baseline system. For the table, we used precomputed data as much as possible to minimize compute times. We see that with an order of magnitude reduction in computation and a significantly simpler implementation, we can achieve the same error rate.

Table 5.2: Summary of some IPDF performances and computation time normalized to a baseline system. Compute time includes compensation and inner product only.

Comparison Function	Enroll Comp.	Verify Comp.	EER Eng (%)	minDCF Eng (×100)	Compute time
C_Q	$O_{V,U,D_{TGT}}$	$Q_{U,D_{TST}}$	2.11	1.23	1.00
C_{GM}	$Q_{U,D_{TGT}}$	$Q_{U,D_{TST}}$	1.73	0.98	0.17
C_{GM}	$Q_{U,D_{\mathrm{UBM}}}$	$Q_{U,D_{\mathrm{UBM}}}$	1.89	1.05	0.08
C_{GM}	$Q_{U,D_{\mathrm{UBM}}}$	I	2.76	1.26	0.04

5.6 Discussion

This chapter proposed the inner-product decision function (IPDF) framework for speaker comparison and compensation and showed that several recent systems in the speaker verification literature can be placed in this framework. We then used the framework to compare the different systems to one another to identify the key components required to achieve good performance. The results of this analysis showed that it is important to include mixture weights in the inner product, and that the more computational costly oblique compensations are not necessary for good performance. We then proposed a comparison function that combined these insights and had substantially reduced computation cost without sacrificing accuracy.

Chapter 6

Toward Reduced False Alarms Using Cohorts

In identification and verification tasks it is usually the case that the target prior is significantly lower than the non-target, thus when a system is deployed it is expected that the majority of test instances are non-targets which leads to a large number of false alarms. The NIST Speaker Recognition Evaluation (SRE) [1] takes this into consideration by setting the parameters of the detection cost function (DCF) to penalize false alarms more severely than misses. As shown in Table 2.1, the 2010 NIST SRE increased the cost of false alarms (FAs) by adjusting the DCF parameters: a typical system yields approximately 0.01% false alarms at the minimum DCF operating point. At that operating point the detection threshold falls in the tail of the non-target score distribution which is not a region that typical speaker verification and normalization algorithms optimize for. Typical algorithms focus on ensuring a large degree of separation between target and non-target score distributions and typical score normalization schemes attempt to reduce score distribution variability over different target models and test recordings.

This work examines the low false-alarm region and proposes algorithms that attempt to tackle it directly. The approaches leverage a large auxiliary set of unlabeled impostor (non-target) recordings to identify suspect false-alarm trials whose match score can then be penalized. Thus, the enabling factor in these algorithms is low-cost comparison functions, such as the TV system (Section 2.2.6) and the C_{GM} IPDF with orthogonal compensation (Section 5.3.4). The focus of this work will be on the one-conversation train one-conversation test scenario and the development set is an extended trial set drawn from the 2008 NIST SRE [19] English telephony data. Final performance will be measured on the extended condition 5 of the 2010 [20] NIST SRE which consists of normal vocal effort English telephony speech.

To motivate the approaches presented in this chapter, we sketch out an example scenario in Figure 6-1. The figure shows recordings in the speaker similarity space, where the distance between two points represents the speaker similarity between two recordings as computed by the comparison function. The target and test recordings of two trials 1 & 2 are shown along with impostor recordings. The distance between target and test is equivalent in both trials, and thus both would, conventionally, be considered equally likely to be true trials, where the target and test contain the same speaker. However, examining these trials within the context of the impostor recordings, one could argue that trial 2 is less likely to be a true trial: the target recording in 2 is closer to impostors than it is to the test recording, while the target and test recordings in 1 are closest to each other. This intuition leads to the algorithms presented in this chapter that identify and penalize suspect trials such as 2. It is also apparent, from this sketch, that for these to work they require a dense sampling of impostor recordings, which is why fast comparison functions are key enablers.

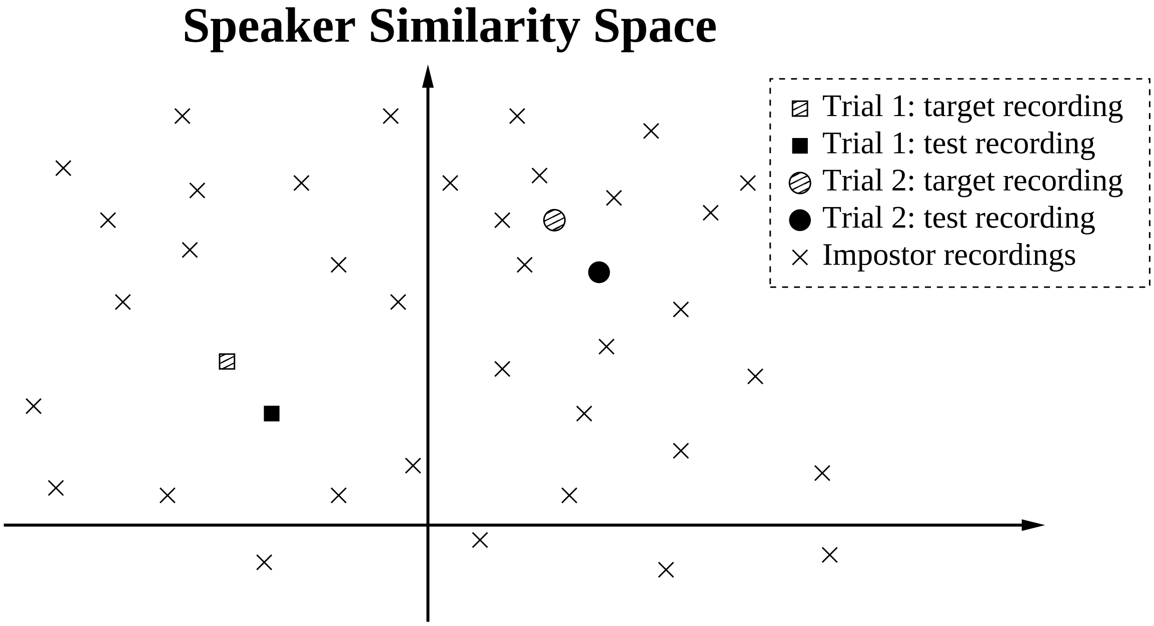
Speaker Similarity Space
▨ Trial 1: target recording
■ Trial 1: test recording
▨ Trial 2: target recording
● Trial 2: test recording
× Impostor recordings

Figure 6-1: Motivating Example

The chapter begins by briefly introducing the baseline system used in this work and highlighting the difficulty encountered by these systems in the low-FA region. The proposed methods to tackle this difficulty are then presented and evaluated on an extended English telephony development set from the 2008 NIST SRE with promising outcomes. The methods are then applied to the telephony condition of the 2010 NIST SRE with less favorable results. This unexpected discrepancy between the 2008 and 2010 evaluations is explored and the likely reason identified and fixed resulting in improved performance on the 2010 SRE.

6.1 Baseline System and The Problem

6.1.1 Baseline: TV and SNorm

The baseline system used in this work is the total variability (TV) system, as in Section 2.2.6. The particular configuration is presented in [53] and operates on cepstral features, extracted using a 25 ms Hamming window. 19 Mel frequency cepstral coefficients together with log energy are calculated every 10 ms. Delta and double delta coefficients were then calculated using a 5 frame window to produce 60-dimensional feature vectors. This 60-dimensional feature vector was subjected to feature warping using a 3 s sliding window. The UBMs used are gender dependent Gaussian mixture models containing 2048 Gaussians. The UBM and the LDA projection are trained on data from the Switchboard II, Switchboard cellular, and telephone recordings from the 2004/05/06 NIST SRE. The TV subspace is trained on these corpora as well as the Fisher English corpus. The WCCN matrix is computed using only the telephone recordings from the 2004/05/06 NIST SRE data sets.

It is common for speaker verification systems to be followed by a score normalization technique, the goal of which is to reduce within trial variability leading to improved performance, better calibration, and more reliable threshold setting. In this work symmetric score normalization (SNorm), Section 2.2.7, is used as the baseline with gender dependent impostor lists consisting of 614 female and 406 male English telephone recordings drawn from the 2005/06 NIST SRE data-sets.

6.1.2 The Problem

The 2010 NIST SRE set a very low prior of 0.001 on target trials in the detection cost function (DCF) which results in false alarms costing significantly more than misses. The minimum DCF threshold, therefore, falls in the tail of the non-target trial scores as can be seen in Figure 6-2. For the TV baseline with and without SNorm the figure shows the minimum DCF threshold and the overlap of the histograms of the target and non-target trial scores of the development set used. The low overlap between target and non-target trials in both plots and the reduced variance of the scores for the SNormed system highlight the efficacy of the TV system for speaker verification and SNorm

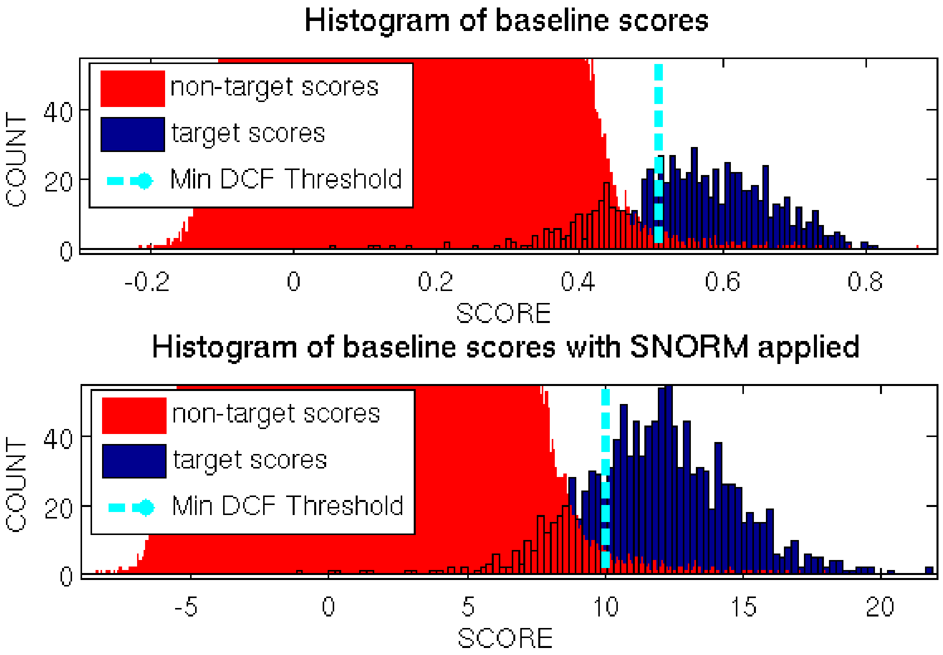

Figure 6-2: The Problem

for score normalization. However, TV and SNorm, though effective, do not specifically tackle the tails of the score distributions in the overlap region, which we will attempt to do in this work.

6.2 Proposed Systems

We tackle the problem by trying to identify the high scoring non-target trials, i.e. the trials in the tail. This is done by leveraging a wealth of data available as an impostor set, a set of recordings that do not share common speakers with the development or test set, and asking the question: “are the two recordings in the trial more similar to one another or to recordings in the impostor set?” Gender dependent impostor sets are used consisting of 9281 female and 6932 male telephony recordings from the 2004/05/06 NIST SREs excluding those used to perform SNorm. All match scores, between the trial recordings or a trial recording and an impostor recording, are computed using the symmetric equation (2.19).

In the proposed methods, one is not constrained to using a specific system to score trials. However, inner product scoring based systems, such as TV [28] and inner product decision functions (Chapter 5), are especially well suited because they allow for fast and efficient comparison of a large number of recordings, as is needed when scoring each trial recording against the thousands of impostor recordings.

6.2.1 False-Alarm Detectors

Nearest Neighbor AND/OR (NN-AND/NN-OR)

We begin with two strategies to detect whether a trial is likely a non-target trial, i.e. one that would contribute to false alarms. The first proposed strategy, called NN-OR, flags a trial as a non-target if *either* of the trial recordings, target or test, are closer, as indicated by a higher match score, to recordings in the impostor set than to the other trial recording. The second, called NN-AND, flags a trial as non-target if *both* trial recordings are closer to recordings in the impostor set.

We evaluate the two strategies on the development data-set by examining the percentage of target and non-target trials that get detected and labeled as non-target trials, a perfect detector being one that would have detected and flagged 100% of the non-target and 0% of the target trials. Table 6.1 shows that while the majority of the non-target trials were detected correctly, a significant number of target trials were falsely detected.

Table 6.1: Percent of trials flagged on the development set

$Strategy$	$\%$ target flagged	$\%$ non-target flagged
NN-OR	18.7	99.87
NN-AND	25.2	99.96

This observation suggests a strategy that, rather than making a hard decision to label all recordings flagged by these detectors as non-targets, penalizes those trials by subtracting an offset from the trial score. Figure 6-3 shows the minDCF and EER values on the development set as a function of the offset, and shows that both strategies perform better than the baseline SNorm system and that NN-AND with an offset of 2 yields the best performance.

Nearest Neighbor Difference (NN-DIFF)

In both NN-AND and NN-OR each trial is either flagged as a non-target or not flagged. We now propose to instead assign a confidence score $c_D(\mathbf{R}_{TGT}, \mathbf{R}_{TST})$, where $\mathbf{R}_{TGT}$ is the enrollment recording and $\mathbf{R}_{TST}$ is the test recording, to each trial based on how suspect it is, by:

$$\begin{aligned} c_D(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) &= \frac{1}{2}\{\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - \hat{s}(\mathbf{R}_{TGT}, NN_1(\mathbf{R}_{TGT}))\} \\ &+ \frac{1}{2}\{\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - \hat{s}(\mathbf{R}_{TST}, NN_1(\mathbf{R}_{TST}))\}. \end{aligned} \tag{6.1}$$

where $\hat{s}(.,.)$ is the SNormed TV match score, and $NN_1(utt)$ is the recording in the impostor set that is nearest, has highest match score, to utt. c_D will therefore take on a large negative value when we are highly confident that a trial is a non-target, and a large positive value when we are highly confident it is a target trial. The confidence score is then fused with the baseline SNorm score to obtain the final trial score

$$s_D(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) = (1-f)\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - f * c_D(\mathbf{R}_{TGT}, \mathbf{R}_{TST}), \tag{6.2}$$

where $f \in [0, 1]$. Figure 6-3 shows the minDCF and EER values on the development set as a function of the fusion parameter, with $f = 0$ being the baseline SNorm system and $f = 1$ using the confidence score as the trial score. The parameter sweep suggests that a good choice of f is in the range of .3 to .6. Also, setting the trial score to be the confidence score, i.e. $f = 1$, performs well at the minDCF point yet poorly at the EER.

6.2.2 K Nearest Neighbor Difference (KNN-DIFF) and Adaptive Symmetric Normalization (ASNorm)

The first set of proposed methods share a common shortcoming: they heavily rely on a single nearest neighbor from the impostor set. We therefore extend the NN-DIFF idea in an attempt to reduce this

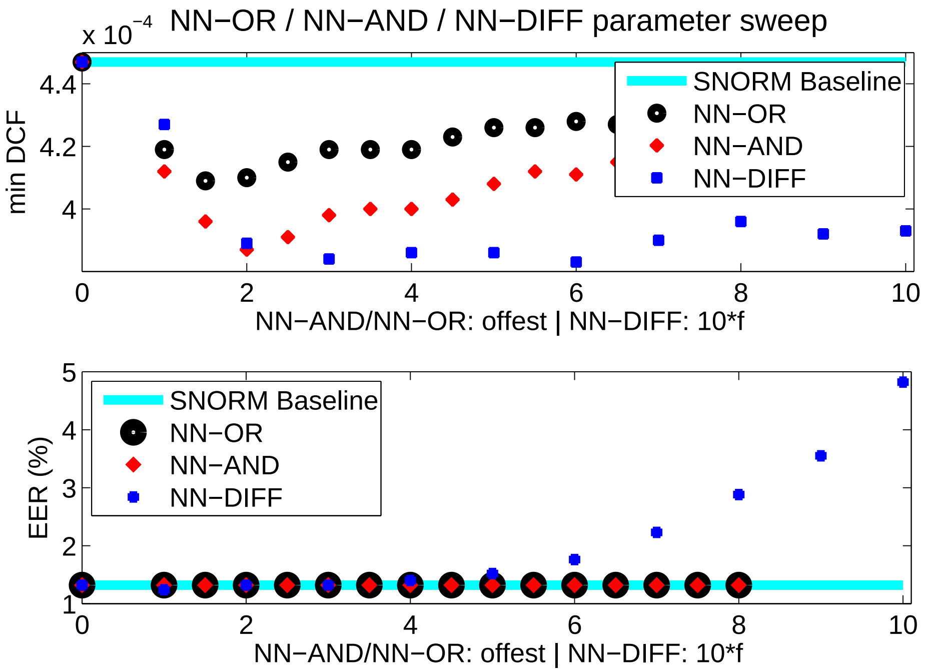

Figure 6-3: Offset penalty sweep for NN-AND, NN-OR, and NN-DIFF

reliance by averaging the scores of the top K NNs rather than just the first, and call it KNN-DIFF. The confidence score is now

$$\begin{aligned} c_{KD}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) &= \tfrac{1}{2}\{\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - \mu(\hat{s}(\mathbf{R}_{TGT}, NN_K(\mathbf{R}_{TGT})))\} \\ &+ \tfrac{1}{2}\{\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - \mu(\hat{s}(\mathbf{R}_{TST}, NN_K(\mathbf{R}_{TST})))\}, \end{aligned} \tag{6.3}$$

where $\mu(.)$ is the mean and $NN_K(.)$ is the set of the K NNs. As K gets large we can further divide out the standard deviation in the confidence score resulting in an adaptive symmetric normalization (ASNorm), similar to TopNorm [54] and ATNorm [30]:

$$\begin{aligned} c_{ASN}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) &= \frac{\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - \mu(\hat{s}(\mathbf{R}_{TGT}, NN_K(\mathbf{R}_{TGT})))}{\sigma(\hat{s}(\mathbf{R}_{TGT}, NN_K(\mathbf{R}_{TGT})))} \\ &+ \frac{\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - \mu(\hat{s}(\mathbf{R}_{TST}, NN_K(\mathbf{R}_{TST})))}{\sigma(\hat{s}(\mathbf{R}_{TST}, NN_K(\mathbf{R}_{TST})))}, \end{aligned} \tag{6.4}$$

where $\sigma(.)$ is the standard deviation. Figure 6-4 shows how increasing K affects each of the strategies. Notice that a lower number of cohorts, $K = 50$, is needed in KNN-DIFF, while $K = 1500$ is best for ASN.

We now choose the best performing confidence scores $c_{KD,K=50}$ and $c_{ASN,K=1500}$ and fuse

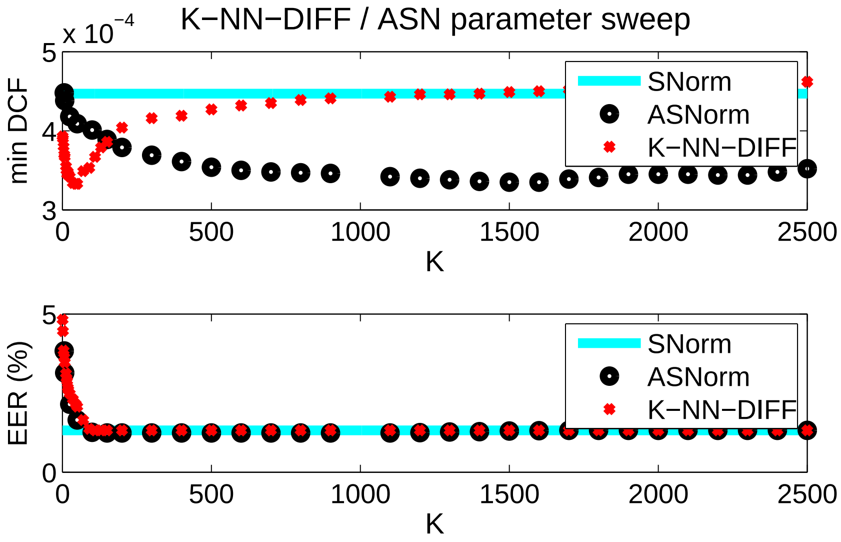

Figure 6-4: Offset penalty sweep for K-NN-DIFF and ASN

them with the baseline SNorm scores,

$$\begin{aligned} s_{KD}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) &= (1-f)\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - f c_{KD,K=50}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) \\ s_{ASN}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) &= (1-f)\hat{s}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}) - f c_{ASN,K=1500}(\mathbf{R}_{TGT}, \mathbf{R}_{TST}), \end{aligned}$$

and show the sweep of the fusion parameter f in Figure 6-5. The fusion shows that to optimize for minDCF f should be set to 0, meaning that the confidence score c_{KD} or c_{ASN} should be used rather than fusing with SNorm. However, the fusion does benefit EER specifically in the KNN-DIFF case, where $f = .7$ seems to be a reasonable trade-off between DCF and EER.

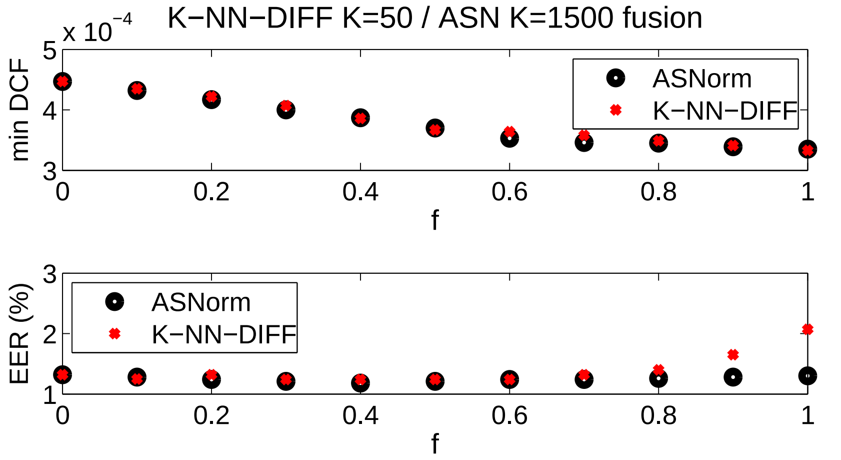

Figure 6-5: Fusion of KNN-DIFF and ASNorm with SNorm

6.2.3 Analysis

We first examine Table 6.2 and Figure 6-6 (A) and notice that even the simplest of the proposed strategies, that rely only on the first NN and make hard decisions to flag a trial as non-target, can yield overall improvement over SNorm and specifically a 13% relative improvement at minDCF. Using the confidence score in NN-DIFF as the trial score, however, aggressively targets the low-FA region of the DET curve at the expense of the rest. Fusing the confidence score with SNorm provides a less aggressive system that improves in the region of interest while performing reasonably elsewhere.

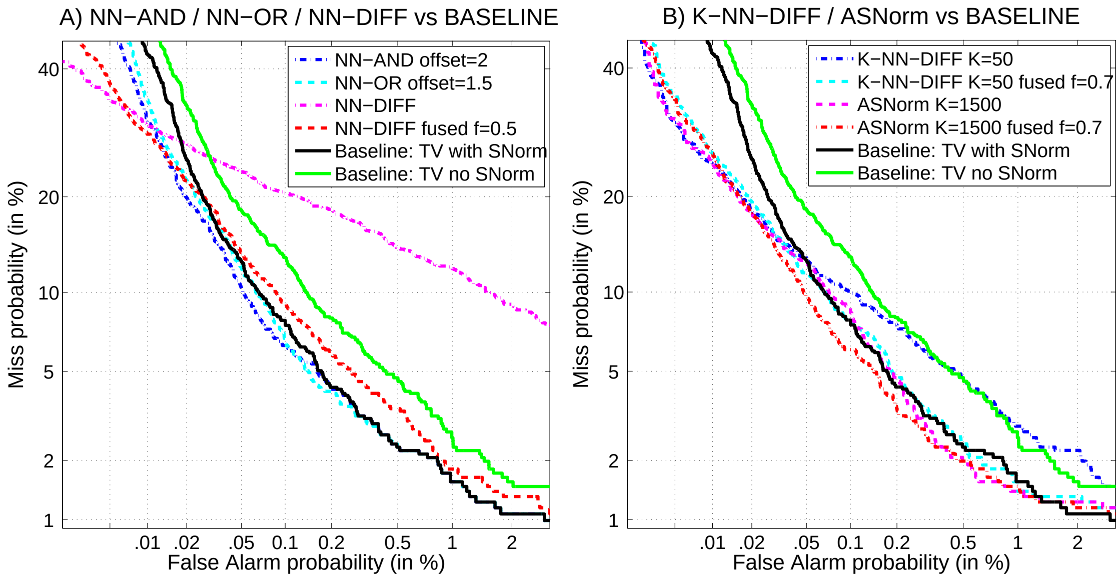

Figure 6-6: DET plots of the different systems on the development set.

Table 6.2: Percent of trials flagged on the development set

$Strategy$	DCF*1e4	EER (%)
Baseline: TV no SNorm	5.32	1.73
Baseline: TV with SNorm	4.47	1.32
NN-OR offset=1.5	4.09	1.32
NN-AND offset =2	3.87	1.32
NN-DIFF	3.93	4.82
NN-DIFF fused f=.5	3.86	1.52
KNN-DIFF K=50	3.33	2.07
KNN-DIFF K=50 fused f=.7	3.58	1.32
ASNorm K=1500	3.35	1.30
ASNorm K=1500 fused f=.7	3.46	1.24

The results of KNN-DIFF and ASNorm shown in Table 6.2 and Figure 6-6 (B) show that utilizing more than one NN in the confidence score further improves performance at minDCF, yielding a 25% relative improvement over SNorm. However, the two methods differ greatly in performance over the rest of the DET curve: KNN-DIFF only shows improvement in the low-FA region while ASNorm improves overall. Fusing the confidence score with the SNorm trial score trades off performance at the low-FA range for overall performance.

6.3 NIST SRE 2010 results

We now present in the first columns of Table 6.4 and Figure 6-7 the results of the proposed methods on condition 5 of the 2010 NIST SRE versus the baselines. It is apparent from the DET plot that the improvement in performance observed on the development data-set is not seen on the test set, specifically at the minDCF operating point.

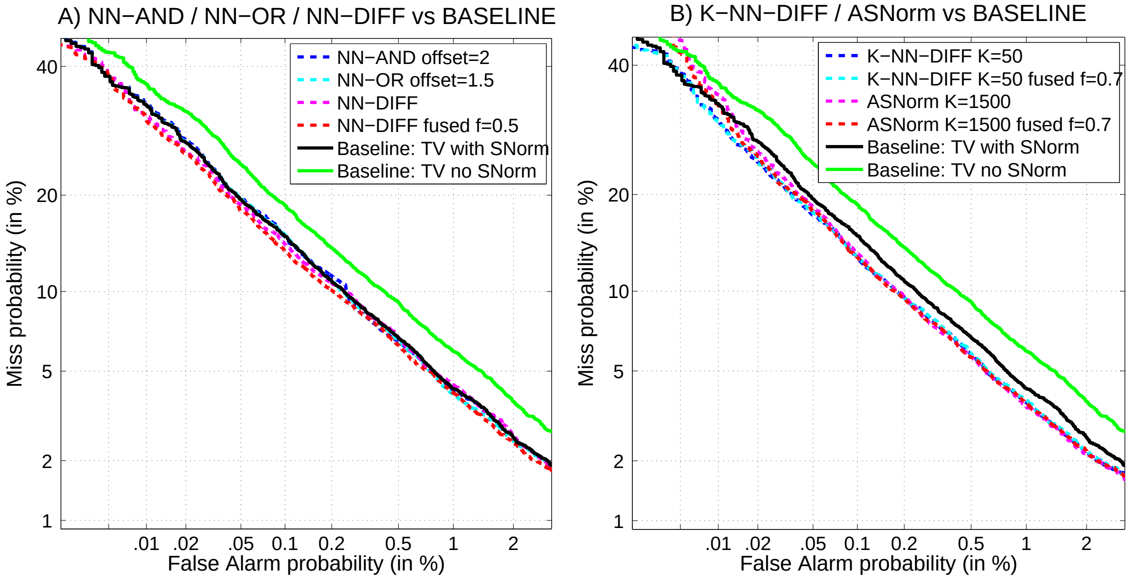

Figure 6-7: DET plots of the different systems on 2010 NIST SRE.

In an attempt to resolve this discrepancy we examine the percentage of trials being flagged as non-targets in the simple NN-AND and NN-OR algorithms, shown in the first two columns of Table 6.3. Comparing these percentages to those in Table 6.1 it is apparent that the test data-set is interacting with the impostor set in a different manner than the development set: specifically a significantly smaller percentage of trials are being flagged as non-targets. This could be for one of two reasons: either the within set variability is lower for the test set than the development set, or the impostor set is better matched to the development data.

Table 6.3: Percent of trials flagged on the test set

Strategy	% target flagged	% non-tar flagged	% target flagged+08	% non-tar flagged+08
NN-OR	5.7	99.32	8.38	99.71
NN-AND	10.7	99.76	16	99.92

As changing the within-set variability would require changing the system we are using to drive the experiments, we therefore attempt to better match the impostor set to the test set by including the 2008 NIST SRE English telephony recordings in the impostor set. The last two columns of Table 6.3 show that there is about a two-fold increase in the number of flagged recordings, indicating that the 2008 data is better matched to the 2010 data. The last two columns of Table 6.4 and Figure 6-8 show that augmenting the impostor set to better match the test data does improve performance over the original impostor set.

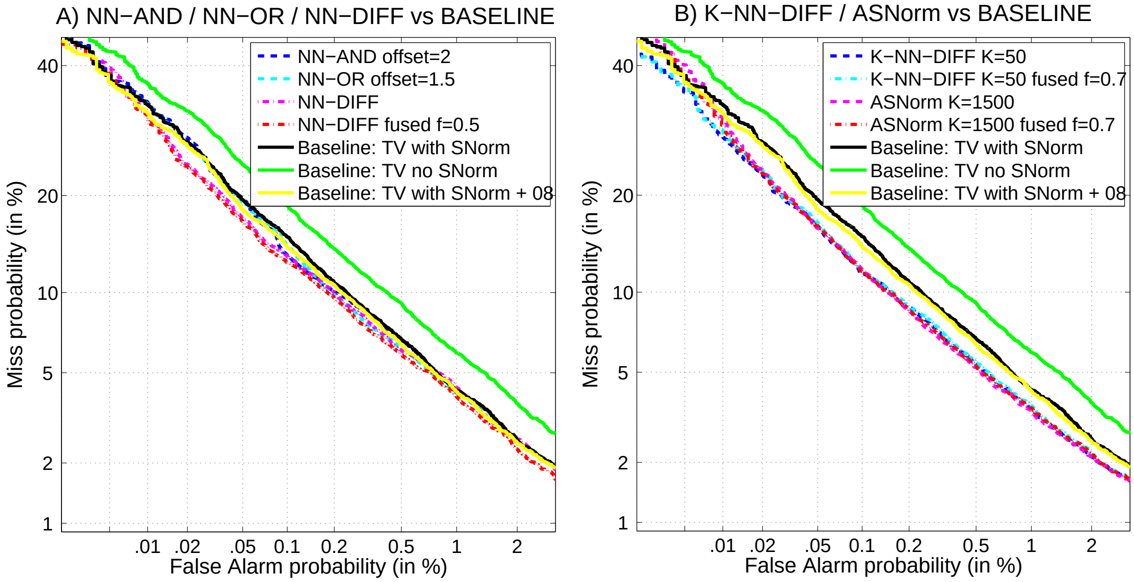

Figure 6-8: DET plots of the different systems with the augmented impostor set on 2010 NIST SRE.

To provide a fair comparison between our proposed systems and the SNorm baseline we augment the SNorm set with a uniformly selected subset of recordings from the 2008 data-set. The comparison with the baseline is presented in Table 6.4 and Figure 6-8 and, even though the improvement is not as dramatic as was seen on the development data, there is a consistent improvement in performance over the DET range between the minDCF point and the EER point. Specifically, a $5 - 10\%$ and $8 - 10\%$ relative improvement at the minDCF and EER points respectively for the KNN-DIFF and ASNorm systems. However, even though the performance did improve it still falls short of expectation. This may be because the percentage flagged in the last two columns of Table 6.3 are still lower than those in Table 6.1 indicating a likely persistent mismatch not addressed by augmenting the impostor set.

Table 6.4: minDCF and EER breakdown on test set

$Strategy$	DCF *1e4	EER (%)	DCFe4 with 08	EER (%) with 08
Baseline: TV no SNorm	4.62	2.82	4.62	2.82
Baseline: TV with SNorm	4.21	2.32	4.13	2.29
NN-OR offset=1.5	4.21	2.30	4.21	2.32
NN-AND offset =2	4.23	2.32	4.28	2.32
NN-DIFF	4.07	2.30	4.11	2.32
NN-DIFF fused f=.5	4.07	2.22	4.05	2.16
KNN-DIFF K=50	4.00	2.11	3.70	2.06
KNN-DIFF K=50 fused f=.7	4.01	2.13	3.80	2.09
ASNorm K=1500	4.33	2.09	4.02	2.08
ASNorm K=1500 fused f=.7	4.17	2.11	3.92	2.11

6.4 Discussion

The goal of this work was to attempt to directly tackle the newly proposed DCF with systems that leverage a large impostor set. Our results on the development set were very promising with even the simplest algorithms outperforming the baseline. However, performance on the test set was on-par with the baseline. Upon exploring this discrepancy, it became apparent that an impostor set that is well matched to the data of interest is crucial to the proposed algorithms. Augmenting the impostor to better satisfy this criterion led to better performance. However, performance still fell short of what was observed on the development set, most likely due to not addressing all of the mismatches. An avenue of future work is to explore techniques to identifying well matched impostor sets. It would also be of interest to further examine this apparent mismatch between the 2010 NIST SRE data-set and the NIST SRE data from previous years.

Chapter 7

Graph Embedding: Manifolds and Geodesics

The KL divergence approximations used in the derivations of the MAP and MLLR GSV kernels, Sections 2.2.3 & 3.3.2, hold locally, as is the case for linearized scoring of the JFA [33]. Though these approximations hold locally, they are applied globally, which raises the question of whether there is a more suitable global distance. This question, and the recent work on total variability [53], which suggests that the majority of the variability between recordings lies in significantly lower dimensional space, compel us to explore whether the variability, instead, lies on a low-dimensional non-linear manifold. There are several techniques in the literature to explore manifold structure and embed data onto manifolds, such as ISOMAP [55] and locally linear embedding [56], as well as techniques that incorporate the manifold structure into the classifier, such as manifold regularization of SVMs [10]. In this chapter we focus on manifold discovery and embedding and do so using ISOMAP.

The extension from linear subspaces to non-linear manifolds, though compelling, is not trivial, because unlike linear subspaces, manifolds cannot, in general, be parametrized by a set of basis vectors and do not have corresponding simple projection operators. Even though a global representation of the manifold may not be available, the distance along the manifold (geodesic distance) [55] between two points lying on it, can be approximated with graph geodesics. The graph-geodesic distance between two points, is the length of the shortest path connecting them along a graph embedding of the data. For the graph embedding to capture the global structure, and the graph geodesics to properly approximate the true geodesic, a large auxiliary data-set is needed to densely sample all the variability in the data. Given graph-geodesic distances, ISOMAP [55] can be used to explore the existence and dimension of a manifold and embed data in it.

The goal of this chapter is to explore the use of graph geodesics and manifolds in the context of speaker comparison. We will begin by describing the large-dimensional inner-product space we have chosen to base our exploration on. We then discuss embedding data on graphs and computing graph geodesics. Next, we briefly outline ISOMAP and apply it to the model-parameter space to explore the existence and size of the underlying manifold. We then present results on data-mining experiments, which show that the use of graph-geodesic distances can greatly improve classification. Finally, we propose a method to use graph geodesics in an evaluation scenario along with results.

7.1 Inner-product Space for Speaker Comparison

Graph embedding, which we will discuss in Section 7.2, requires computing the euclidean distance between each point in a large auxiliary data-set and all others. For speaker comparison, this translates to computing the speaker-similarity, or match score, between all the recordings in the data-set, thus making it more crucial to have a fast comparison function. In this chapter, we chose to use the C_{GM} IPDF, Section 5.3.4, with factor analysis (FA) orthogonal compensation, Section 5.4.2. This can be written, since the comparison function is an inner product, as:

$$s(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \mathbf{u}_\alpha^T \mathbf{u}_\beta, \tag{7.1}$$

where $\mathbf{u}_\alpha$ & $\mathbf{u}_\alpha$ are the compensated supervectors representing $\mathbf{R}_\alpha$ & $\mathbf{R}_\alpha$ in the speaker comparison space. The supervectors are further magnitude normalized $\bar{\mathbf{u}} = \mathbf{u}/\left|\left|\mathbf{u}\right|\right|_2^2$, as this was empirically shown to improve the result of the geodesic approximation, resulting in the following comparison:

$$\bar{s}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \bar{\mathbf{u}}_\alpha^t \bar{\mathbf{u}}_\beta. \tag{7.2}$$

The associated euclidean distance in this space is therefore,

$$d_{euc}(\mathbf{R}_\alpha, \mathbf{R}_\beta) = \sqrt{2 - 2\bar{\mathbf{u}}_\alpha^t \bar{\mathbf{u}}_\beta}. \tag{7.3}$$

For the experiments in this work, the frame level feature extraction was performed using HTK [57] with 20 MFCC coefficients, deltas, and acceleration coefficients for a total of 60 features, with speech activity detection and feature warping [23] applied. The UBM consists of a 512 mixture GMM and MAP adaptation of the means was performed with a relevance factor of 10^{-5}, while the mixture weights of the UBM were replaced by their maximum-likelihood (ML) estimates. The FA compensation was trained using speakers from the NIST 2004 SRE corpora [58]. The resulting euclidean space has dimension $30,720$.

7.2 Graph Embedding of Speech Recordings

Graph embedding of a data-set can help explore, visualize and uncover structure in the data, as we show in Chapter 9. It is also the first step to computing approximate geodesic distances between two recordings.

Nodes in the graph represent recordings while weighted and undirected edges represent speaker-similarity between a pair of recordings. To assess this notion of similarity, we first compute a large speaker-similarity matrix capturing the similarity between each recording in the data and all others; the i, j^{th} entry of the matrix is the euclidean distance, using equation (7.3), between the i^{th} &j^{th} recordings. An edge between two nodes exists if their corresponding recordings are deemed "similar enough", and in this chapter, this is decided using one of two ways: the first, connects two points if they lie within some "epsilon" euclidean distance of each other, and the second, connects two vertices if one is among the K-nearest neighbors (NN) of the other in the euclidean space. The weights of the edges are the euclidean distances between their recordings. We will refer to graphs built based on the epsilon distance as ϵ-graphs, and those based on NN as NN-graphs.

When performing the graph embedding, a summarized version of the similarity matrix is first computed, either based on epsilon distances or on K-nearest neighbors, with the only valid entries being those corresponding to the existing edges. Note that the summarized matrix and the resultant graph are two ways to represent the same information. Figure 7-1 sketches out the embedding process for four recordings $\{\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C, \mathbf{R}_D\}$.

To compare the two edge selection techniques and decide which is more suitable for speaker comparison, we compare the resultant node-degree (number of edges a node possesses) distribution of the graphs to the "correct" distribution. The correct degree distribution, is that of a graph in which all recordings of the same speaker are connected with one another and there are no edges between recordings of different speakers. Figure 7-2 shows histograms of the degree distributions of sample NN and ϵ-graphs as well as the correct graph on the NIST SRE Eval-04 data-set, which contains

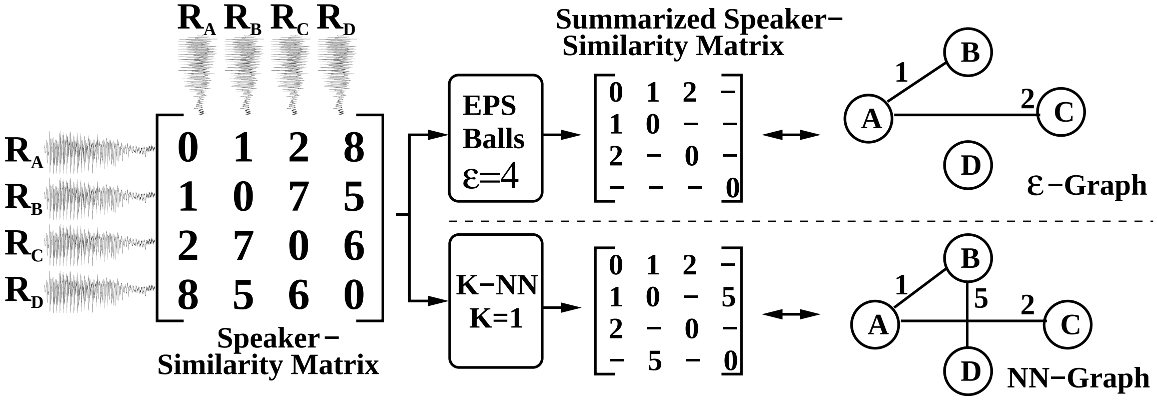

Figure 7-1: Sketch of graph embedding.

212 speakers and a total of 5213 recordings. We see that the degree distribution of a NN-graph with $K = 16$ has the same range and a similar trend as that of the correct graph. The ϵ-graph, however, is significantly different regardless of the choice of epsilon, this is because the variance within the speaker recordings is not consistent across speakers, the figure shows two choices of ϵ. Based on these observations, we choose to use NN-graphs in the rest of this chapter.

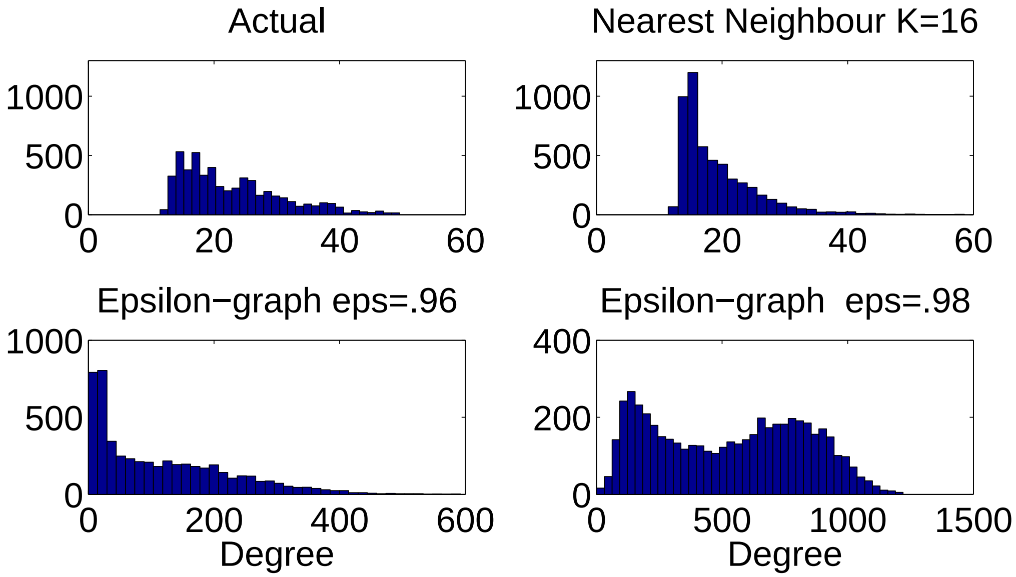

Figure 7-2: Histogram of degree distribution.

7.3 Geodesics

If we assume that the recordings lie on a low-dimensional manifold in the speaker-similarity space, then the euclidean distance between two recordings that are far apart may not a be a faithful representation of the speaker similarity. A better choice may be the geodesic distance, which is the length of the shortest path connecting them along the manifold, between the two recordings. Figure 7-3 sketches the difference between the two distances for a manifold with an intrinsic dimension of two in a three-dimensional euclidean space.

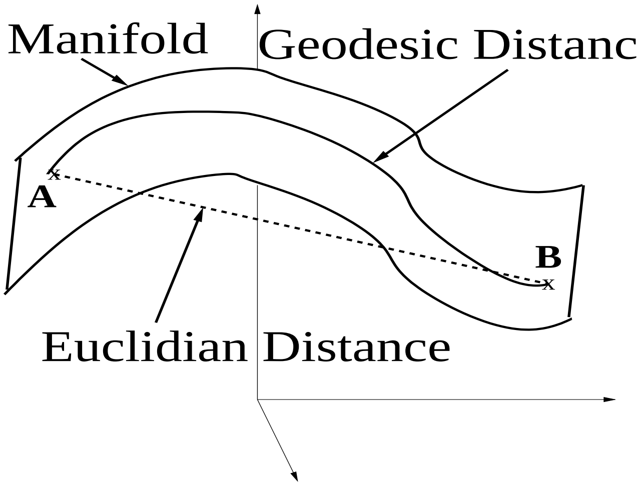

Figure 7-3: Geodesic and euclidean distances between A and B.

Though they differ over large distances, the euclidean and geodesic distances are approximately equivalent for arbitrarily short distances. This equivalence can be used to approximate the geodesic distance [55] as follows:

We first assume that enough recordings are available such that they densely sample the manifold in the euclidean space, and embed these recordings on a NN or ϵ-graph, as described in the previous section. The graph only connects nodes that are similar and if the space is densely sampled, we can assume the weight of the edge between two recordings is a faithful representation of how similar they are. Thus, the geodesic distance between two recordings can be approximated using the graph geodesic, which is computed by summing the weights of the edges along the shortest path in the graph connecting their corresponding nodes. Figure 7-4 sketches this approximation for a manifold with an intrinsic dimension of two in a three-dimensional euclidean space.

We will refer to the graph embedding of the recordings used to densely sample the manifold as the NN-background, and finding the graph-geodesic distance between any two points in the

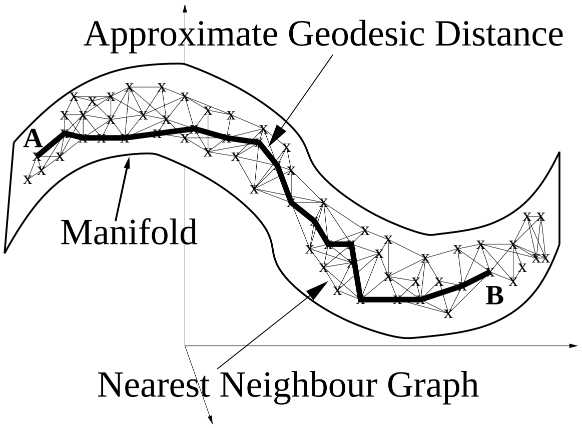

Figure 7-4: Approximate geodesic distance between A and B.

NN-background involves just finding the shortest path along the graph. However, to compute the graph-geodesic distance between two recordings not part of the NN-background, they must first be “connected” to the graph. This is done by adding the recordings as vertices in the graph, calculating the euclidean distance from them to the rest of the vertices, and modifying the edge connections to obtain the NN-graph one would have gotten had the two recordings been part of the NN-background. Once they are “connected” the graph-geodesic distance is again the length of the shortest path along the graph connecting the nodes. To compute the shortest path we use a Matlab implementation [55] of the Dijkstra algorithm [59].

In this chapter we will examine the use of geodesic distances in three speaker comparison scenarios:

- A data-mining scenario where the NN-background includes just the recordings of interest for comparison.
- A data-mining scenario where the NN-background includes the recordings of interest for comparison as well as additional recordings whose purpose is to attempt to more densely sample the manifold, the hope being that this would yield a more accurate approximate geodesic distance between the recordings of interest.
- An evaluation scenario where the NN-background does not include any of the recordings we wish to compare and includes only recordings that attempt to densely sample the manifold.

7.4 ISOMAP

ISOMAP [55] is a technique that is used to explore the existence and dimension of the manifold, as well as embed points into it [55]. The embedding uses the graph-geodesic distances, to map the data from the original high dimensional euclidean space into a lower dimensional space in which the euclidean distance is equivalent to the geodesic in the original space. We will refer to the euclidean distance in the embedded space as the ISOMAP distance. Multidimensional scaling (MDS), a technique used for dimensionality reduction and data visualization [60], is used to perform the embedding. The optimal size of the lower-dimensional coordinate space is, in general, not known a-priori and can be estimated by examining the decay of the residual variance, the variance in the data unaccounted for by the embedding. In this chapter we used the software package [61] to apply ISOMAP.

It is important to note that ISOMAP requires access to all the data, one wishes to embed, a-priori to estimate the manifold and embed the points in it. This requirement prohibits ISOMAP from being used in an evaluation scenario where one does not have access to the testing recordings to train the classifier.

7.4.1 ISOMAP Applied to Speech Recordings

The speaker-similarity euclidean space which we have chosen to represent speech recordings described in Section 7.1 has a dimension of 30720, however, previous work [53] had shown that good speaker separation can be done in a significantly smaller space of dimensionality 400. This smaller space is essentially the subspace of largest variability in the original space. In this section, we attempt to uncover whether the data lies near a non-linear manifold and if so what its dimension is:

We apply ISOMAP with $K = 6$, the parameter used to build the NN-graph, to three NN-backgrounds:

- 5213 recordings of the NIST SRE Eval-04 data-set, which contain 212 speakers from both genders.
- 5742 recordings, of both genders, from the 1 and 3 conversation enroll and 1 conversation test tasks of the NIST SRE Eval-06.
- 23000 recordings, of both genders, sub-selected from the NIST SRE 04/06/08 evaluations as well as the Fisher corpora.

Figure 7-5 examines the decay of the residual error as the embedding dimension is increased. Note that most of the variability in the Eval-04 data-set can be captured by a 50 dimensional manifold, and similarly for the Eval-06 data-set. However, when the NN-background includes speech from multiple sources the intrinsic dimension is closer to 100 with an overall higher residual error, which seems to indicate a lack of consistency in the manifold across the data-sets.

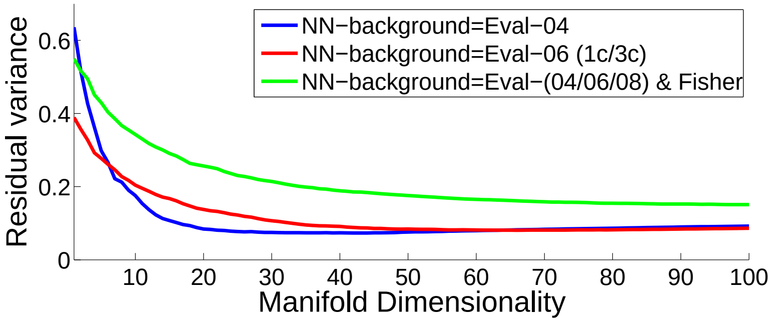

Figure 7-5: Decay of residual error with increasing embedding dimension.

To further highlight the existence of an underlying manifold of speaker variability, Figure 7-6 shows the two-dimensional embedding, with Eval-04 as the NN-background, of 5 recordings from 10 male and 10 female speakers randomly selected from the 212 speakers from the Eval-04 data-set. Each set of similarly colored "o"s represents recordings from a male speaker, and the set of similarly colored "x"s represents recordings from a female speaker. It is interesting to note that both speaker and gender separation can be observed in this two-dimensional embedding.

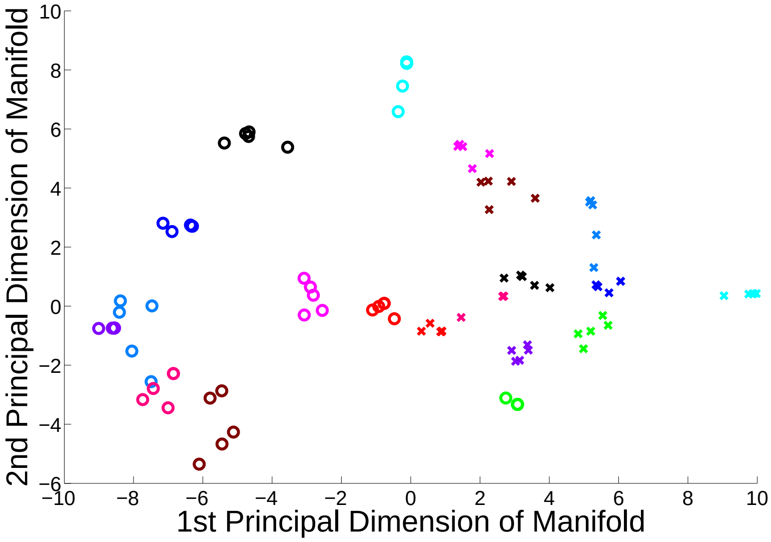

Figure 7-6: 5 recordings each from 20 speakers embedded on the estimated two-dimensional manifold. "o" for males, and "x" for females.

7.5 Graph Geodesics for Speaker Recognition

In this section we examine the possibility of using approximate-geodesics and manifold distances to perform speaker verification. We do this by comparing three classifiers:

C_E Labels two recordings as belonging to the same speaker if the euclidean distance between them in the original speaker-similarity space is below a threshold.

C_G Labels two recordings as belonging to the same speaker if the graph-geodesic distance between them is below a threshold.

C_I Labels two recordings as belonging to the same speaker if the ISOMAP distance is below a threshold.

For C_G and C_I, we use a $K = 6$ NN-graph and will explicitly state what NN-background was used in each of the results presented below. For C_I, the dimensionality of the manifold is fixed at 50.

7.5.1 Data-Mining Task

The previous section showed that indeed speech recordings lie near a low-dimensional manifold in the model parameter space. One would therefore expect that using graph-geodesic distances rather than euclidean distances will yield more accurate speaker comparisons. We explore this expectation using data-mining experiments, where it is assumed that all enroll and test data is available, though unlabeled, to the classifier.

Figure 7-7 shows a detection error trade-off (DET) plot that compares the three classifiers on the NIST SRE Eval-04 data-set, where pair-wise comparisons between all the recordings were performed. For C_G and C_I, the NN-background consisted of the Eval-04 data-set itself. Note the large improvement in classification when the manifold is taken into account, either by using graph geodesics (C_G) or the ISOMAP distance (C_I). It is also important to note that the 50 dimensional embedding performed by the ISOMAP algorithm does not completely characterize the manifold, thus resulting in the performance of C_I being poorer than that of C_G.

Figure 7-8 shows the DET plot of the classifier performance for all trials of the NIST SRE Eval-06 1 conversation train 1 conversation test ($1c$) task [18]. Two C_G and C_I classifiers were trained, the first used only the $1c$ data as the NN-background, while the second also included the enrollment recordings from the NIST SRE Eval-06 3 conversation train 1 conversation test ($3c$) task. Similarly to the results on Eval-04 the C_G and C_I classifiers outperform C_E, with C_G performing better than

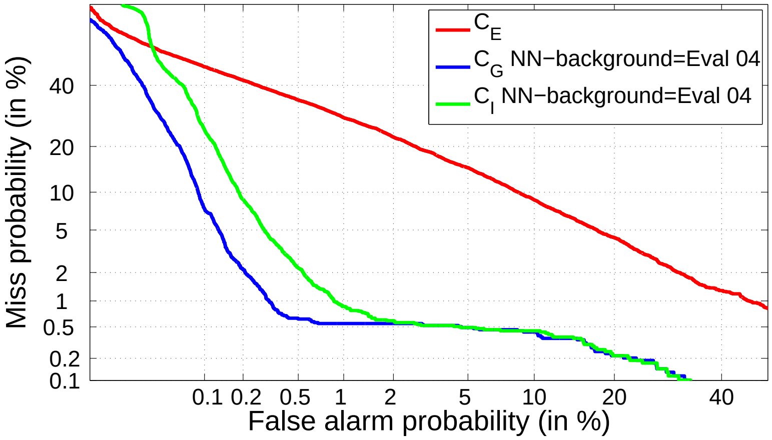

Figure 7-7: DET plot of classifiers using euclidean, geodesic and ISOMAP distances for the NIST SRE Eval-04 data-set.

the corresponding C_I. The DET-plot also shows the performance of a C_G classifier whose NN-background contains a total of 23000 recordings from NIST SRE Eval-(04/06/08) and the Fisher database. One would expect extending the NN-background beyond the Eval-06 $1c$ and $3c$ will yield improvement across the whole DET curve as the additional data will result in denser sampling of the manifold yielding more accurate geodesic distances. However, as seen in the figure, performance is improved in the low false-alarm regime and worsened at the low probability-of-miss regime. This lack of overall improvement may be due to a miss-match in the underlying manifold on which the Eval-06 and the Fisher data lie.

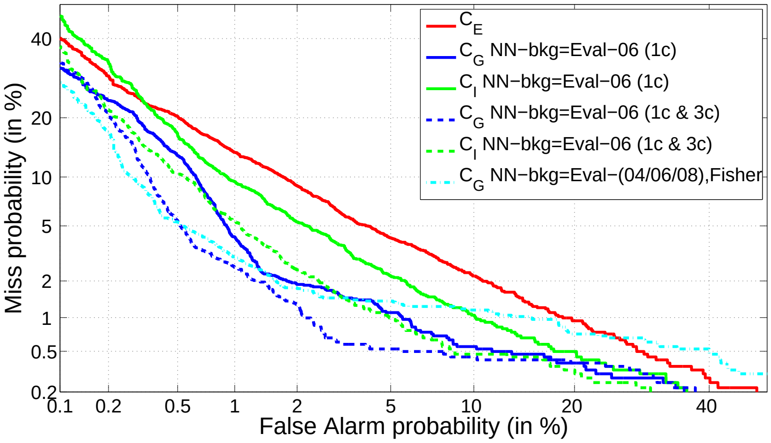

Figure 7-8: DET plot of classifiers using euclidean, geodesic and ISOMAP distances on All trials of the Eval-06 $1c$ task.

7.5.2 NIST SRE Task

The data-mining results showed that, if the evaluation data is available a-priori, the graph-geodesic distances can greatly improve classification results, and that the choice of the NN-background in computing the graph geodesics is important since it essentially defines the manifold. In an evaluation scenario, such as the NIST SRE, the classifier does not have a-priori access to the evaluation data and thus the NN-background cannot include the data on which the classifier will be evaluated, as was done in the data-mining experiments. This restriction also prohibits us from using ISOMAP to perform the embedding, since it requires the train and test data to be part of the NN-background. Therefore, in this section we focus on comparing graph geodesics to the euclidean distance on all trials of the Eval-06 1c task:

For C_G, the NN-background used was the Fisher data-set and the number of nearest neighbors (K) used to create the NN-graph was varied from 2 to 25. Figure 7-9 shows the effect of varying K on the detection cost function point (DCF) and the equal error rate (EER) point, with the minimum DCF occurring at $K = 3$ and min EER occurring at $K = 23$. In Figure 7-10, we show the DET plot for the C_G classifiers for $K = 3$ & 23 as well as the C_E baseline. These, figures show that the performance of the geodesic distance classifier is based on the choice of K and only yields an improvement over the baseline in certain regimes of the DET plot. The discrepancy between these results and the significant improvements seen in the data-mining experiments is perhaps due to a miss-match in the underlying manifolds of the Fisher data and the Eval-06 data.

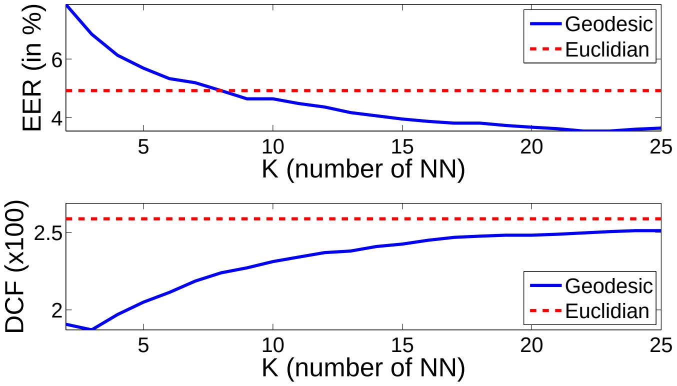

Figure 7-9: DCF and EER vs K of C_G on All trials of the Eval-06 $1c$ task.

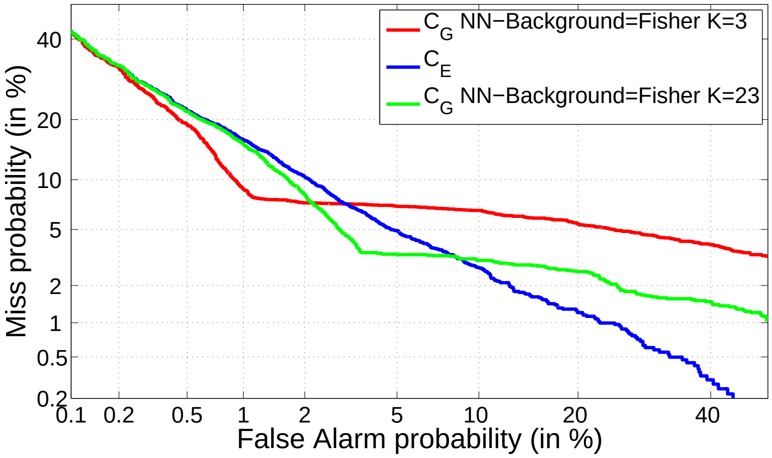

Figure 7-10: DET plot for C_G with $K = 3$ and $K = 6$ as well as C_E.

7.6 Discussion

Using the ISOMAP algorithm, we have empirically shown that there exists an underlying manifold on which speech-recordings live in the speaker-similarity space. We used NN-graph embedding as a proxy for the manifold, which allowed for computing graph-geodesic distances. Using the graph-geodesic distance and the ISOMAP distance in the manifold embedding greatly improves classification, over the euclidean baseline, in data-mining experiments. Results on NIST-SRE Eval-06 core task show that this improvement is only observed in some regimes of the DET plot at the cost of degradation in others. Future work could examine this discrepancy in performance improvement between the data-mining experiments and the NIST SRE experiments, with the ultimate goal being a competitive classifier that fully exploits the structure of the manifold.

Chapter 8

Graph Embedding: Graph-Relational Features

In this thesis we've already explored two ways to leverage comparisons between the trial recordings and a large auxiliary set to improve speaker comparison: In Chapter 6, we used a large set of impostor recordings to reduce false-alarms, by performing adaptive score normalization based on the immediate neighborhood around the trial recordings. In Chapter 7, we used the scores between the trial and background recordings to embed the trial recordings as nodes in a graph and used the graph-geodesic distance between them as a speaker-match score. In this chapter, we combine the local neighborhood around the trial recordings with the geodesic distance between them and other relational features to perform speaker comparison.

Motivated by the link prediction problem [62], this work embeds the trial recordings along with the background set in a graph and, in addition to using the shortest path as a match score, extracts several other features that capture the interconnection between the trial recordings and the background. We will refer to these as graph-relational features and use them to represent each trial. These features are used in a classifier, e.g. linear SVM, to separate between true trials, where the trial recordings correspond to the same speaker, and false ones.

We will begin with a description of the total variability system which we will use both as a baseline and for graph construction. We then discuss the graph construction and embedding, followed by the relational features we'll extract from the graph. Next, we present the classifier used along with the train and test setup. We conclude with results that show the efficacy of these features and suggestions for future work.

8.1 Total Variability (TV) and Graph Embedding

The baseline system used in this work, and the one used to build the graph, is the total variability (TV) system, as in Section 2.2.6, followed by SNorm score normalization, Section 2.2.7. The particular configuration is presented in [53] and operates on cepstral features, extracted using a 25 ms Hamming window. 19 Mel frequency cepstral coefficients together with log energy are calculated every 10 ms. Delta and double delta coefficients were then calculated using a 5 frame window to produce 60-dimensional feature vectors. This 60-dimensional feature vector was subjected to feature warping using a 3 s sliding window. The UBMs used are gender dependent Gaussian mixture models containing 2048 Gaussians. The UBM and the LDA projection are trained on data from the Switchboard II, Switchboard cellular, and telephone recordings from the 2004/05/06 NIST SRE. The TV subspace is trained on these corpora as well as the Fisher English corpus. The WCCN matrix is computed using only the telephone recordings from the 2004/05/06 NIST SRE data sets. The gender dependent impostor lists used for SNorm consisted of 614 female and 406 male English telephone recordings drawn from the 2005/06 NIST SRE data-sets. We will use $\hat{s}(\mathbf{R}_\alpha, \mathbf{R}_\beta)$ and $s(\mathbf{R}_\alpha, \mathbf{R}_\beta)$ to refer to the TV and TV combined with SNorm symmetric scoring functions between two recordings $\mathbf{R}_\alpha$ and $\mathbf{R}_\beta$.

$s(\mathbf{R}_\alpha, \mathbf{R}_\beta)$ is used to compute a pair-wise match score between each pair of recordings in the set consisting of the background and trial recordings, resulting in a square and symmetric match-score matrix. The score matrix encodes not only the direct comparison between the trial recordings, but also how they interact with the background set. This information can be leveraged to improve on the direct match score. Motivated by the link prediction problem [62], we generate a relational graph that summarizes the score matrix and extract graph-relational features. These features combined with the direct match score are combined to train a classifier that discriminates between true and false trials.

Section 7.2 of the previous chapter describes how the relational graph can be constructed. However, unlike the previous chapter, we do not restrict ourselves to just NN-graphs, and allow for ϵ-graphs as well. The choice of graph construction method, and the parameters K and ϵ, will result in very different graphs. These differences allow us to examine the match-score matrix from different perspectives which we speculate would yield somewhat complementary graph-relational features. We therefore include both construction methods and several parameter choices in the feature extraction process.

Another choice in graph construction is whether the edges of the graph are weighted or not. Weighted graphs, like those of used in the previous chapter, use the pair-wise score between two recordings for the weight of the edge connecting them. Binary graphs on the other hand have all their edge weights set to unity, therefore all the information is encoded in whether an edge exists between two nodes or not. In the next section, we propose several graph-relational features, some applicable to both binary and weighted graphs, others to only one.

8.2 Graph-Relational Features

Once the trial and background recordings are embedded in a graph we can extract several features that capture the interaction between the trial recordings via the graph. These features are split into two main classes: those that examine only the immediate neighborhood of the trial recordings and those that extend beyond that. To simplify the presentation of the features we first present some notation:

- The nodes in the graph, representing trial and background recordings, are indexed from 1 to T, where T is the total number of nodes in the graph.
- Each trial consists of a target and test recording TGT and TST respectively.
- NN_x is the set of neighbors of node x, i.e. the nodes connected to x by an edge. For example NN_{TGT} is the set of neighbors of TGT.
- $|X|$ is the cardinality of the set X .
- $||x||$ is the 2-norm of the vector x .
- The vectors v_x are typically sparse vector, of size Tx1, that capture the interaction of x with the remaining graph nodes:

  - Zero valued entries in the vectors indicate the lack of an edge between the recording x and the nodes corresponding to the zero locations.

  - For weighted graphs, the value of the non-zero vector entries indicates the weight of the edge between x and the corresponding graph nodes.

  - For binary graphs, all non-zero entries have a value of one and indicate edges between x and the corresponding graph nodes.

8.2.1 Neighborhood Features

The premise of these neighborhood features is that if TGT and TST are recordings of the same speaker then their match scores with the background recordings should be similar indicating they lie within the same neighborhood of the graph.

Binary graph

We adopt the following features, which were proposed in [62] for link prediction:

- *Common neighbors*=$|NN_{TGT} \cap NN_{TST}|$ counts the number of common neighbors between TGT and TST.
- *Jaccard's coefficient*=$\frac{|NN_{TGT} \cap NN_{TST}|}{|NN_{TGT} \cup NN_{TST}|}$ normalizes the common neighbor score by the total number of nodes connected to both TGT and TST. An example scenario where the normalization would be useful, is where a particular target recording TGT shares the same number of common neighbors with two separate test recordings TST_1 and TST_2, however $|NN_{TST_2}| \gg |NN_{TST_1}|$ and thus the Jaccard coefficient would penalize TST_2.
- *Adamic*=$\Sigma_{z \in NN_{TGT} \cap NN_{TST}} \frac{1}{log|NN_z|}$ a measure that combines the size of the intersection set with how highly connected the nodes in the intersection are. This could be thought of as another form of normalized common neighbors.

Weighted graph

The features in this section are inspired by those of the binary graph.

- *Inner product*=$v_{TGT}^T.v_{TST}$ is based on the common neighbors measure.
- *Normalized inner products*=$\frac{v_{TGT}^T.v_{TST}}{||v_{TGT}||.||v_{TST}||}$ and $\frac{v_{TGT}^T.v_{TST}}{||v_{TGT}||+||v_{TST}||}$ which are inspired by Jaccard's coefficient.
- *Adamic Weighted*=$\Sigma_{z \in NN_{TGT} \cap NN_{TST}} \frac{1}{log||v_z||}$, based on the binary Adamic feature.
- *Landmark Euclidean distance*=$||v_{TGT} - v_{TST}||$, a measure that considers the recordings in the graph as landmarks and that the vectors v_{TGT} and v_{TST} represent the coordinates of TGT and TST in the space defined by the landmarks.

8.2.2 Paths Features

In the previous sections our discussion has focused on graphs constructed based on match scores. One can also create graphs based on the Euclidean distance between the TV representation of the recordings. In the K-NN version of the distance based graphs the NN are selected to be the closest ones to a recording in the Euclidean space. And in the epsilon version of the graphs, edges exist between nodes that are less than ϵ apart from one another. Given the normalization of the match score presented in Section 2.2.6 the euclidean distance between two recordings is just

$$e(\mathbf{R}_a, \mathbf{R}_b) \quad = \quad \sqrt{2 - 2\hat{s}(\mathbf{R}_a, \mathbf{R}_b)}. \tag{8.1}$$

These distance graphs allow for extracting paths based features that go beyond the immediate neighborhoods of the trial recordings:

Shortest path

- *Shortest path*=$2^{-SP(TGT,TST)}$, where $SP(TGT, TST)$ is the value of the shortest path from node TGT to TST, which we compute using a Matlab implementation of the Dijkstra algorithm [9].
- *Number of hops*=$2^{-NH(TGT,TST)}$, where $NH(TGT, TST)$ is the number of edges traversed along the shortest path from TGT to TST.

N-Step Markov (NSM):

NSM is a feature used to quantify the relative importance of TGT to TST [63] by computing the probability that a random walk started at TGT will visit TST after N steps are taken. This can be computed as the value at the index of the TST vector:

$$NSM(TGT, .) = \mathbf{A}i_{TGT} + \mathbf{A}^2 i_{TGT} + \mathbf{A}^3 i_{TGT} + ... + \mathbf{A}^N i_{TGT}, \tag{8.2}$$

where i_{TGT} is a vector of size Tx1 of all zeros except for 1 at the index of the TGT, and $\mathbf{A}$ is an TxT matrix representing transition probabilities from one node to another. We obtain $\mathbf{A}$ from the distance graph by dividing each outward edge from a node by the sum of all outward edges from that node. In this work we choose to set $N = 15$ since beyond that the contribution of $\mathbf{A}^N i_{TGT}$ to the NSM score is minimal.

Table 8.1: The graph-relational features used in classification

	K used in K-NN	ϵ used in Epsilon Graph
BGN	5, 10, 20, 50, 100, 250, 500, 750, 1000	.35, .4, .45
WGN	5, 10, 20, 50, 100, 250, 500, 750, 1000	-.4, -.3, -.2, -.1, 0, .1, .2, .3, .4
Paths		1.1, 1.2, 1.3

8.3 Classifier

Section 8.1 presented two graph embedding techniques, K-NN and epsilon graphs, each with a parameter that can be varied to obtain different resultant graphs. These graphs are then used in Section 8.2 to extract three categories of features: binary graph neighborhood (BGN), weighted graph neighborhood (WGN) and paths features. Combining the different graph construction with the different feature extraction techniques results in a large set of features to represent each trial. We narrow the set down to 135 features according to the efficacy of each individual feature on the development set. Table 8.1 lists the resulting set.

These relational features combined with the baseline match-score result in a 136-dimensional feature vector that represents each trial of interest. The features are individually normalized to have zero mean and unit variance across the training set. A linear SVM classifier is then trained, per gender, on the development set to separate between true and false trials. This is done using the LibSVM toolbox [64] with five fold cross-validation to set the regularization parameter c. Once trained, the SVM is used to classify test trials as true or false. The next section presents the results of our approach on speaker recognition and speaker-mining tasks.

8.4 Results

We evaluate the proposed algorithms on the one-conversation train one-conversation test scenario, where each trial contains one target recording and one test. All the experiments use the 2008 NIST SRE English telephony data as a training/development set. And final performance is measured on condition 5 of the 2010 NIST SRE which consists of normal vocal effort English telephony speech.

8.4.1 Speaker Recognition Task

The speaker recognition task follows the standard NIST SRE task requiring that each trial be considered independently of all other trials in the evaluation. Therefore, the auxiliary set used to build the graph and extract graph-relational features for a given trial consisted of only impostor recordings. The background sets used are of size 6932 for males and 9281 for females and consist of recordings from the 2004/05/06 NIST SREs.The regularization parameter c was set via cross-validation to 5 for males and 15 for females. Figure 8-1 shows the detection error trade-off (DET) curves of the baseline, in blue, and our proposed algorithm, in red, on the NIST SRE 08 data, which was also used to train the SVM classifier. When examining this plot it is important to keep in mind that we are testing on the SVM training data, however the plot does highlight the potential of the graph-relational features.

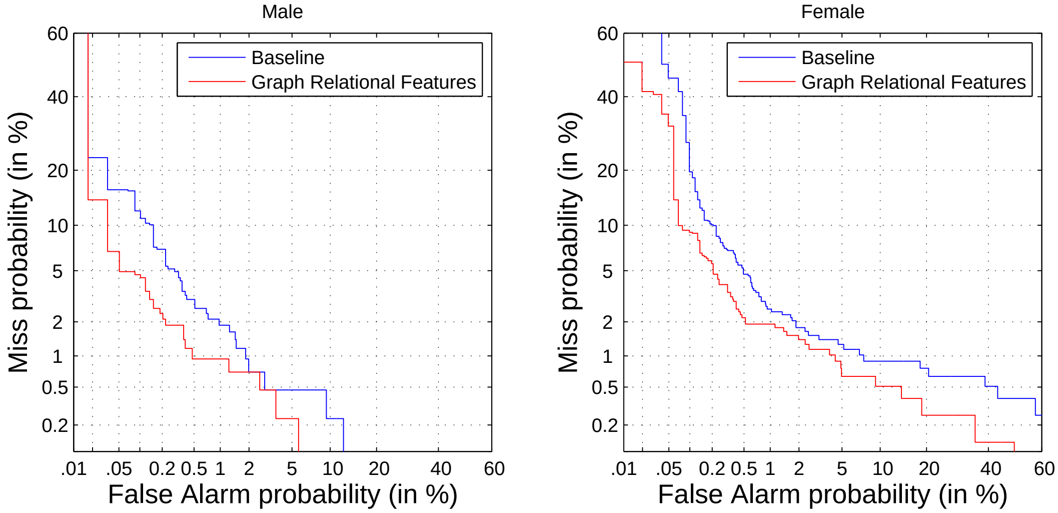

Figure 8-1: Speaker recognition DET plots of the baseline and proposed system on the training set (NIST SRE 08).

Figure 8-2 shows the DET curves of the baseline, in blue, and our proposed algorithm, in red, on the held out test set, NIST SRE 10. Note that our algorithm yields moderate improvement over the baseline.

8.4.2 Speaker-Mining Task

In the speaker-mining task, we relax the constraint requiring each trial to be considered independently and include all the trials of the particular evaluation in the graph background set along with recordings from the 2004/05/06 NIST SREs. This yielded background sets of size 8475 for males and 12099 for females on the development set and 9868 and 13209 for males and females on the held out test set. We note that in this task the background set is not only comprised of impostor

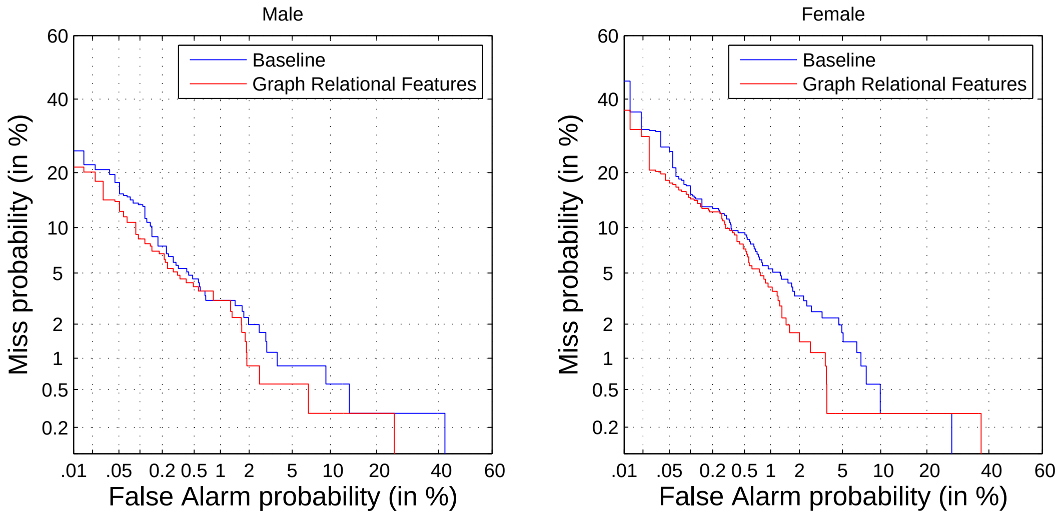

Figure 8-2: Speaker recognition DET plots of the baseline and proposed system on the held out test set (NIST SRE 10).

recordings and may have speaker overlap with the trial of interest. During SVM training the regularization parameter c was set via cross-validation to 3 for males and 2 for females. Figure 8-3 shows the DET curves of the baseline, in blue, and our proposed algorithm, in red, on the NIST SRE 08 data used to train the SVM classifier. Keeping in mind that we are testing on the training data, it is still worthwhile to note the potential of the graph-relational features for speaker mining.

Figure 8-4 shows the DET curves of the baseline, in blue, and our proposed algorithm, in red, on the held out test set, and clearly shows the improvement of our algorithm over the baseline.

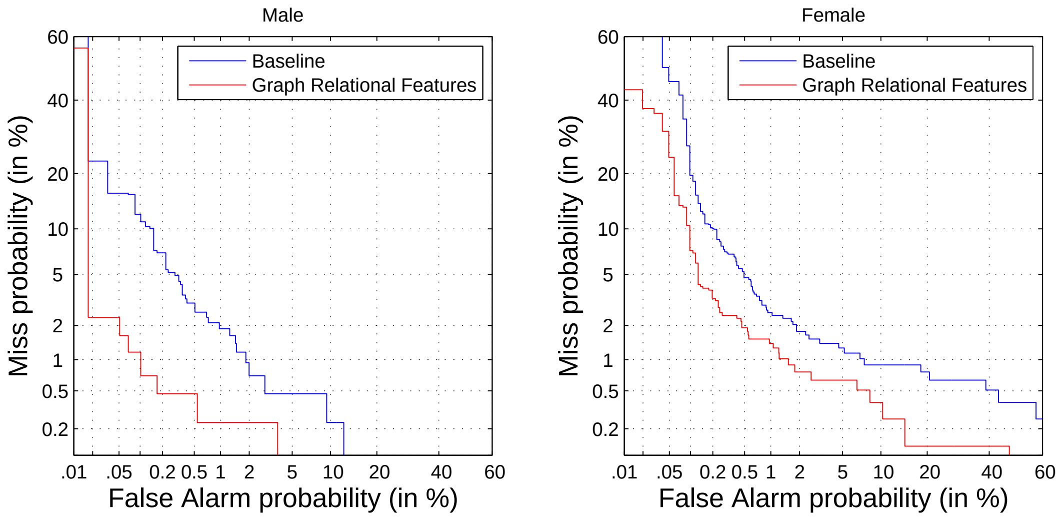

Figure 8-3: Speaker mining DET plots of the baseline and proposed system on the training set (NIST SRE 08).

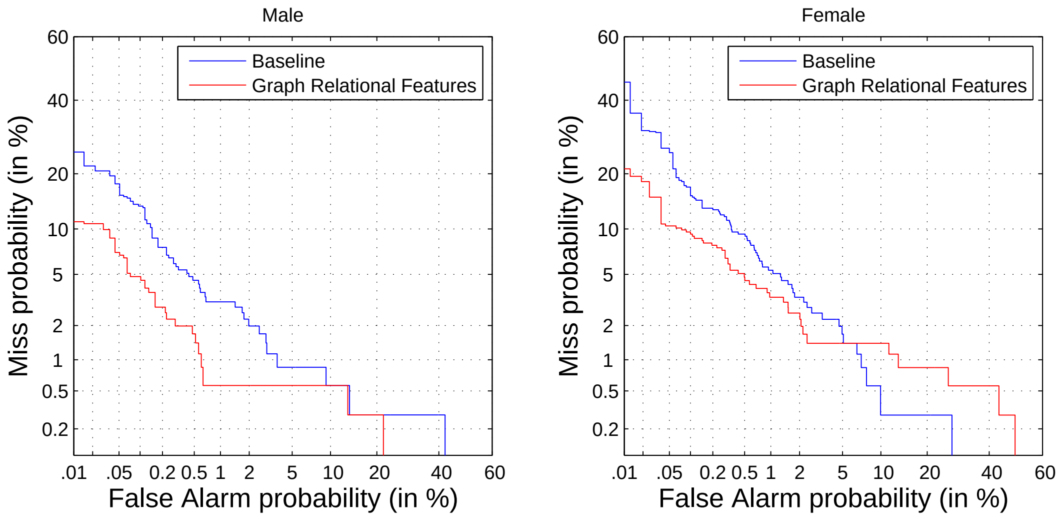

Figure 8-4: Speaker mining DET plots of the baseline and proposed system on the held out test set (NIST SRE 10).

8.5 Discussion

In this chapter, we presented a framework to use graph-relational features extracted from speaker similarity graphs for improved speaker comparison. We applied this framework to two speaker comparison tasks, speaker recognition and mining. In both tasks, our proposed system outperformed the baseline, with significant improvement observed in the speaker-mining task. We also present results from test-on-train scenarios to highlight the potential of the features. There was a noticeable discrepancy between the test-on-train results and testing on the held-out set which is a concern that should be addressed in future work.

The goal of this chapter was to highlight the benefit of using graph-relational features in speaker verification. It, however, does not fully explore this topic and leaves many issues to be addressed in future work, some examples are:

- It is expected that there is significant correlation in the different graph-relational features, specifically between those of the same class (neighborhood or path) and those extracted from the same type of graph ($K - NN$ or ϵ). It would be of interest to understand this correlation and compensate for it in the classifier.

- In this work we chose to use a linear SVM for the classifier, however other classifiers should be considered.

- The set of graph-relational features used in this work is not an exhaustive one and there may be other better or complementary ones we have not considered.

- We considered $K - NN$ and ϵ graph construction techniques, yet there are other choices that may be useful.

- When constructing the graphs we used unlabeled auxiliary recordings, it may however be beneficial to use speaker labeled recordings along with graph construction techniques that exploit the labels.

Chapter 9

Graph Embedding: Data Visualization

The NN-graph of speech recordings, Section 7.2, can serve as a good method to visualize the effects of the algorithms on the data-sets. In the NN-graph the location of the vertices is not important, only the existence and weights of the edges between them. The graph can, therefore, be “laid out” (the process of choosing vertex locations) in a manner that would result in good visualization. We use the GUESS [65] software package to perform both the visualization and the layout using the GEM algorithm [66]. An example of such a layout is presented in Figure 9-1 which shows the layout of the $K = 6$ NN-graph of the Eval-04 telephony data, where the system used was the one in Section 7.1. Male and female recordings are represented by red and green nodes respectively, and the visualization clearly shows the gender separation. This data visualization technique can be used as both an exploratory and a visual analysis tool. In this chapter we present a brief case study showing how this could be done.

In [67] a channel-blind system was proposed that could be used across the different tasks in the NIST 2010 Evaluation [20]. These include recordings of telephony speech as well as various microphone recordings collected from interviews conducted in two separate rooms. This system is based on the TV system, Section 2.2.6 with WCCN and LDA supposedly performing the crucial role of removing channel variability. We use the data visualization technique to examine both the efficacy of the channel compensation in the system as well as to explore the full NIST 2010 evaluation recordings. We present only male recordings since similar results are observed with female recordings. The graphs show all male recordings of the core conditions of the 2010 extended NIST SRE, and the number of NNs is set to $K = 3$.

We begin by showing the efficacy of the channel-blind system by using the system in building

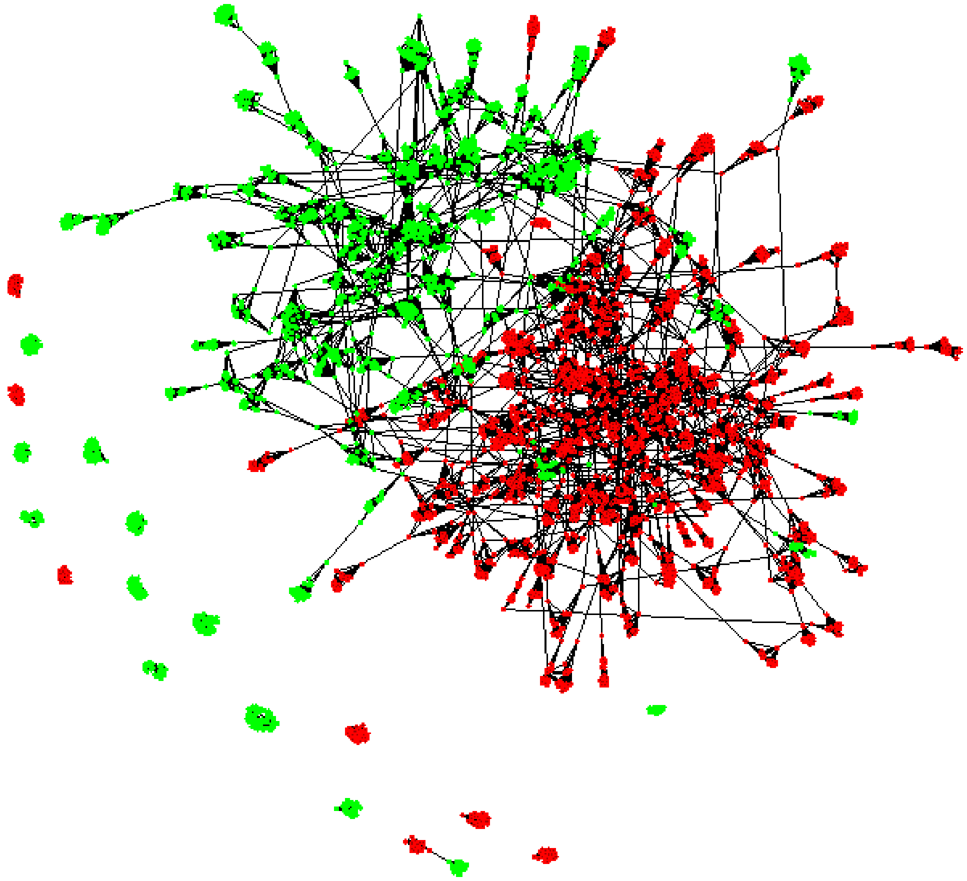

Figure 9-1: Eval-04 NN-graph $K = 6$ male (red) and female (green) recordings.

the NN-graph. Figure 9-2 shows the resultant visualization with speaker meta-data overlaid such that recordings of the same speaker are colored alike. The clusters of similar color, representing clusters of recordings of the same speaker, show that the system is indeed assigning lower cosine distance scores to pairs of recordings of the same speaker.

Next, we examine the importance of the channel compensation performed by the combination of WCCN/LDA. To do this, we build a NN-graph using the channel-blind system without the WCCN/LDA step, the corresponding visualization is in Figure 9-3. We notice that the speaker clustering observed with the full channel-blind system is no longer visible, however, there does seem to be some structure to the graph.

Further exploration, by overlaying channel meta-data, shows that the structure can be attributed to channel variability. Figure 9-4 shows the layout of the NN-graph using the channel-blind system without WCCN/LDA with: colors representing different telephone and microphone channels, the node shape representing the two different rooms the interview data was collected in. Upon careful

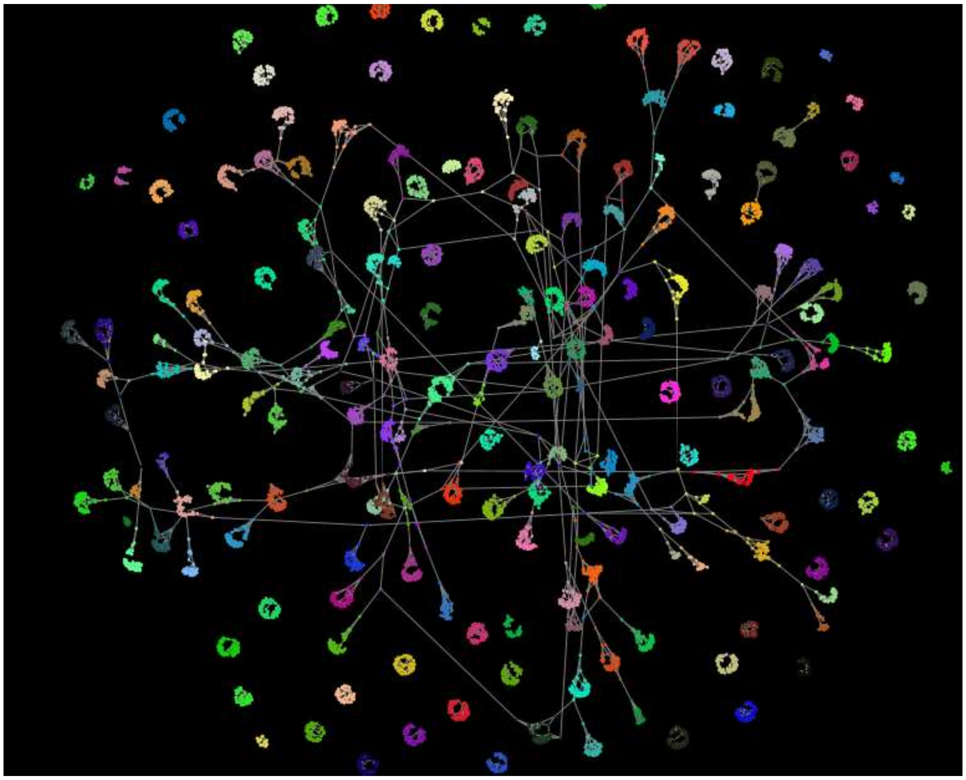

Figure 9-2: Graph visualization of all NIST SRE 2010 Male recordings using the full channel-blind system with speaker meta data overlaid.

inspection of the graph, one notices that the room accounted for more variability than the interview microphones, specifically for the far-talking microphones: MIC CH 05/07/08/12/13. Another worthwhile observation, is that the two phone numbers (215573qqn and 215573now) which are land-line phones located in each of the rooms, cluster near the interview data of the corresponding room, and more specifically near the close-talking and desk microphones: MIC CH 02/04.

This ability to visualize and explore the dominant variability within a data-set may prove to be a useful tool when dealing with newly collected data-sets. In this particular case study, the greater effect of the room variability over that of the microphones, seems to suggest that future NIST SREs should include tasks that test for robustness over varying recording rooms.

Another useful aspect of visualization, which we will only mention here, is to help identify key errors in a data-set. For example, a speaker or gender key error would show up as a node or group of nodes not clustering with their same speaker/gender labeled counterparts.

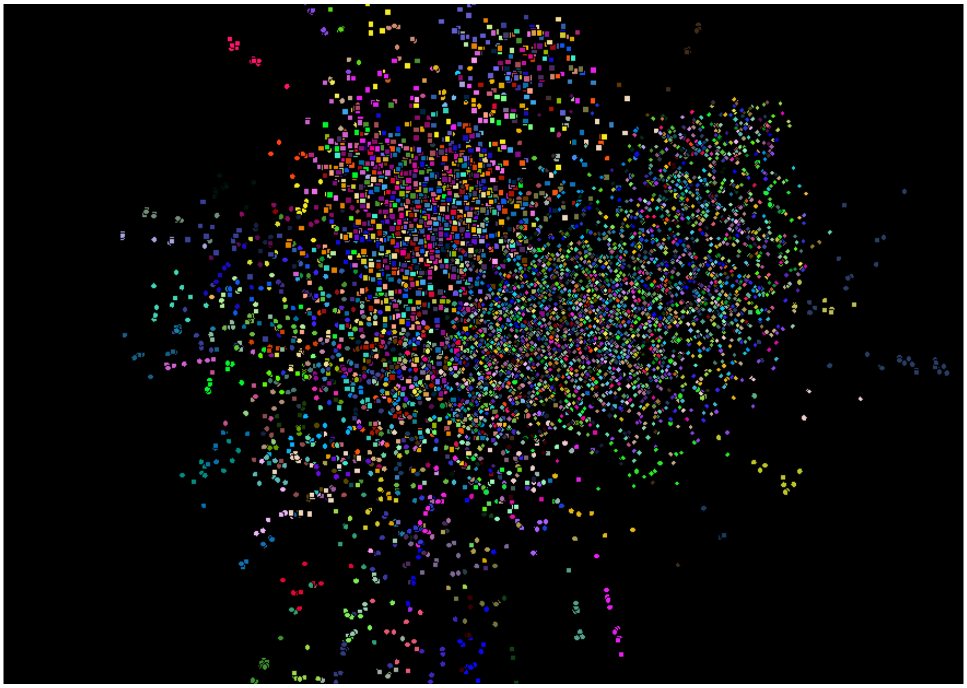

Figure 9-3: Graph visualization of all NIST SRE 2010 Male recordings using the channel-blind system without WCCN/LDA channel compensation with speaker meta data overlaid.

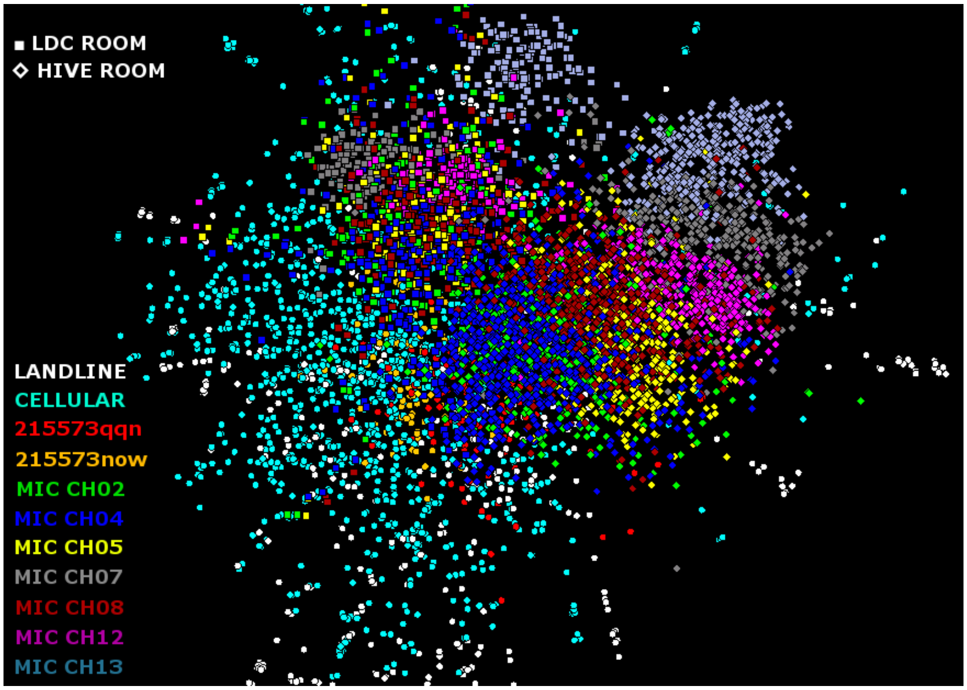

Figure 9-4: Graph visualization of all NIST SRE 2010 Male recordings using the channel-blind system without WCCN/LDA channel compensation with channel meta data overlaid.

Appendix A

Useful Machine Learning Concepts

A.1 Support Vector Machines (SVMs)

An SVM [25] is a two-class classifier constructed from sums of a kernel function $K(\cdot,\cdot)$,

$$f(\mathbf{x}) = \sum_{s=1}^{L} \gamma_s y_s K(\mathbf{x}, \mathbf{x}_s) + b, \tag{A.1}$$

where the y_s are the ideal outputs, $\sum_{s=1}^{L} \gamma_s y_s = 0$, and $\gamma_s > 0$. The vectors $\mathbf{x}_s$ are support vectors (a subset of the training data) and obtained from the training set by an optimization process [37]. The ideal outputs are either 1 or -1, depending upon whether the corresponding support vector is in class 0 or class 1, respectively. For classification, a class decision is based upon whether the value, $f(\mathbf{x})$, is above or below a threshold (usually 0).

The kernel $K(\cdot,\cdot)$ is constrained to have certain properties (the Mercer condition), so that $K(\cdot,\cdot)$ can be expressed as

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z}), \tag{A.2}$$

where $\phi(\mathbf{x})$ is a mapping from the input space (where $\mathbf{x}$ lives) to a possibly infinite-dimensional SVM feature space. We will refer to the $\phi(\mathbf{x})$ as the SVM features.

The focus of the SVM training process is to model the boundary between classes: the boundary is a hyperplane in the SVM feature space defined by the vector $\mathbf{w}$ normal to it:

$$\mathbf{w} = \sum_{s=1}^{L} \gamma_s y_s \phi(\mathbf{x}) + b \tag{A.3}$$

The training process identifies the subset of the training data which are the support vectors $\mathbf{x}_s$ (data that if removed from training set would lead to a different classifier) and associated weights γ_s. Figure A-1 shows the in-class ($+$) and out-of-class ($-$) training points in SVM feature space, the support vectors (circled points), the linear decision boundary, and the normal ($\mathbf{w}$) to it. We will refer to the support vectors, their associated weights, and discriminating direction ($\mathbf{w}$) as the "bi-products" of the SVM training process.

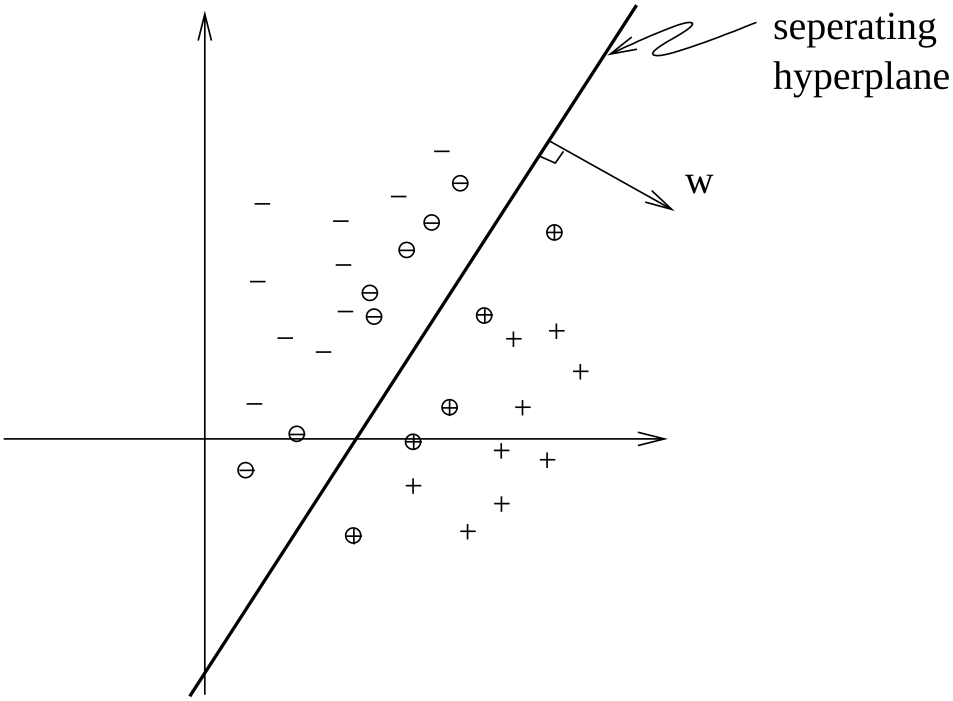

Figure A-1: Example of separating hyperplane

A.2 Gaussian Mixture Models (GMMs)

A Gaussian mixture model (GMM) is a probability density function comprised of a mixture of Gaussian density functions [68]. It models the probability density of a vector $\mathbf{r}$ of size D as:

$$g(\mathbf{r}) = \sum_{i=1}^{M} w_i \mathcal{N}(\mathbf{r}; \mathbf{m}_i, \Sigma_i), \tag{A.4}$$

where w_i is the mixing weight of the i^{th} mixture, $\mathbf{m}_i$ is the mean vector of the i^{th} mixture, Σ_i is the covariance matrix of the i^{th} mixture, and

$$\mathcal{N}(\mathbf{r};\mathbf{m}_i,\Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp\{-\frac{1}{2}(\mathbf{r}-\mathbf{m}_i)^T\Sigma_i^{-1}(\mathbf{r}-\mathbf{m}_i)\}. \tag{A.5}$$

Maximum likelihood (ML) training is typically used to fit the model parameters of the GMM, and is done using expectation maximization (EM) [68]. In this thesis we only consider GMMs with diagonal covariance matrices.

A.3 Maximum A Posteriori (MAP) Adaptation of GMMs

Gaussian mixture models (GMMs) are used throughout this thesis to model features extracted from a recording. This is typically done by adapting the parameters of a universal background model (UBM), a GMM trained to model features extracted from a large and diverse set of recordings:

$$g_{UBM}(\mathbf{r}) = \sum_{i=1}^{M} w_{UBM,i}\mathcal{N}(\mathbf{r};\mathbf{m}_{UBM,i},\Sigma_{UBM,i}). \tag{A.6}$$

In this section we present maximum a posteriori (MAP) adaptation of the means of the UBM to a recording $\mathbf{R}_\alpha = \{\mathbf{r}_{\alpha,1},\mathbf{r}_{\alpha,2},...,\mathbf{r}_{\alpha,N_\alpha}\}$ [2]. MAP adaptation uses the UBM means ($\mathbf{m}_{UBM,i}$) as a prior and moves the means in the direction of the α ML estimate of the means ($\hat{\mathbf{m}}_{\alpha,i}$), which maximizes the likelihood of $\mathbf{r}_\alpha$ being generated by the GMM $\hat{g}_\alpha(\mathbf{r}) = \sum_{i=1}^{M} w_{UBM,i}\mathcal{N}(\mathbf{r};\hat{\mathbf{m}}_{\alpha,i},\Sigma_{UBM,i})$. The amount of movement towards the ML means is based on the amount of adaptation data: the more data available the more the adapted means ($\mathbf{m}_{(\alpha,i)}$) move away from the UBM means and closer to the ML means. Specifically the adapted means are:

$$\mathbf{m}_{\alpha,i} = \frac{\mathbf{N}_i}{\mathbf{N}_i+\tau}\hat{\mathbf{m}}_{\alpha,i} + \frac{\tau}{\mathbf{N}_i+\tau}\mathbf{m}_{UBM,i} \qquad \forall i \tag{A.7}$$

$$\mathbf{N}_i = \sum_{n=1}^{N_\alpha}\frac{w_i\mathcal{N}(\mathbf{r}_n;\mathbf{m}_i,\Sigma_i)}{\sum_{j=1}^{M} w_j\mathcal{N}(\mathbf{r}_n;\mathbf{m}_j,\Sigma_j)} \qquad \& \qquad \hat{\mathbf{m}}_{\alpha,i} = \frac{1}{N_\alpha}\mathbf{N}_i\mathbf{r}_n, \tag{A.8}$$

where τ is a relevance factor that is empirically chosen.

Note that, if a single Gaussian were used instead of GMMs, i.e. $g_{UBM}(\mathbf{r}) = \mathcal{N}(\mathbf{r};\mathbf{m}_{UBM},\Sigma_{UBM})$, then $\hat{\mathbf{m}}_\alpha$ would just be the sample mean of $\mathbf{r}_\alpha$:

$$\hat{\mathbf{m}}_\alpha = \frac{1}{N_\alpha} \sum_{n=1}^{N_\alpha} \mathbf{r}_{\alpha,n}. \tag{A.9}$$

Another important observation is that as $\mathbf{N}_i$ increases (i.e. as more adaptation data is available) the adapted mean approaches the ML mean, with $\hat{\mathbf{m}}_\alpha = \mathbf{m}_\alpha$ when an infinite amount of adaptation data is available.

In a similar manner the covariance matrices of the GMM can also be adapted by MAP adaptation, the details of which can be found in [2].

Bibliography

[1] "NIST speaker recognition evaluation," http://www.nist.gov/speech/tests/spk/.

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

[3] W. M. Campbell, D. E. Sturim, D. A. Reynolds, and A. Solomonoff, "SVM based speaker verification using a GMM supervector kernel and NAP variability compensation," in *Proc. ICASSP*, 2006.

[4] M. Ferras, C. Leung, C. Barras, and J. Gauvain, "Constrained MLLR for speaker recognition," in *Proc. ICASSP*, 2007.

[5] A. Stolcke, L. Ferrer, and S. Kajarekar, "Improvements in MLLR-transform-based speaker recognition," in *Proc. Odyssey*, 2006.

[6] A. Solomonoff, W. M. Campbell, and I. Boardman, "Advances in channel compensation for SVM speaker recognition," in *Proc. ICASSP*, 2005.

[7] A. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for SVM-based speaker recognition," in *Proc. Interspeech*, 2006.

[8] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of inter-speaker variability in speaker verification," *IEEE Transactions on Audio, Speech and Language Processing*, 2008.

[9] J. B. Tenenbaum, V. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, 2000.

[10] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 1, 2006.

[11] S. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised speaker adaptation based on the cosine similarity for text-independent speaker verification," in *Proc. Odyssey*, 2010.

[12] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, pp. 42–54, 2000.

[13] K. P. Li and J. E. Porter, "Normalizations and selection of speech segments for speaker recognition scoring," in *Proc. ICASSP*, 1988.

[14] H. Hermansky, "Perceptual linear prediction (PLP) analysis for speech," *Journal of the Acoustic Society of America*, vol. 87, pp. 1738–1752, 1990.

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

[16] A. Orebaugh and J. Allnutt, “Classification of instant messaging communications for forensics analysis,” *The International Journal of Forensic Computer Science*, vol. 1, pp. 22–28, 2009.

[17] T. K. Moon, P. Howland, and J. H. Gunther, “Document author classification using generalized discriminant analysis,” in *Proc. Workshop on Text Mining, SIAM Int’l Conf. on Data Mining*, 2006.

[18] “The NIST year 2006 speaker recognition evaluation plan,” http://www.nist.gov/speech/tests/spk/2006/index.html, 2006.

[19] “The NIST year 2008 speaker recognition evaluation plan,” http://www.nist.gov/speech/tests/spk/2008/index.html, 2008.

[20] “The NIST year 2010 speaker recognition evaluation plan,” http://www.itl.nist.gov/iad/mig/tests/sre/2010/index.html, 2010.

[21] S. Furui, “Cepstral analysis technique for automatic speaker verification,” *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 29, no. 2, 1981.

[22] H. Hermansky and N. Morgan, “RASTA processing of speech,” *IEEE Trans. on Speech and Audio Processing*, vol. 2, no. 4, 1994.

[23] J. Pelecanos and S. Sridharan, “Feature warping for robust speaker verification,” in *Proc. Odyssey*, 2001.

[24] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, “Support vector machines using GMM supervectors for speaker verification,” *IEEE Signal Processing Letters*, 2005.

[25] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines*, Cambridge University Press, Cambridge, 2000.

[26] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, “Factor analysis simplified,” in *Proc. ICASSP*, 2009.

[27] N. Dehak, P. Kenny, R. Dehak, O. Glembek, P. Dumouchel, L. Burget, V. Hubeika, and F. Castaldo, “Support vector machines and joint factor analysis for speaker verification,” in *Proc. ICASSP*, 2009.

[28] N. Dehak, P. Kenny, R. Dehak, P. Ouellet, and P. Dumouchel, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech and Language Processing*, 2010.

[29] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[30] D. E. Sturim and D. A. Reynolds, “Speaker adaptive cohort selection for tnorm in text-independent speaker verification,” in *Proc. ICASSP*, 2005, vol. I, pp. 741–744.

[31] C. Longworth and M. J. F. Gales, “Derivative and parametric kernels for speaker verification,” in *Proc. Interspeech*, 2007, pp. 310–313.

[32] W. M. Campbell, “Generalized linear discriminant sequence kernels for speaker recognition,” in *Proc. ICASSP*, 2002, pp. 161–164.

[33] O. Glembek, L. Burget, and P. Kenny N. Dehak, N. Brummer, “Comparison of scoring methods used in speaker recognition with joint factor analysis,” in *Proc. ICASSP*, 2009.

[34] M. J. F. Gales and P. C. Woodland, “Mean and variance adaptation within the MLLR framework,” *Computer Speech and Language*, vol. 10, no. 4, pp. 249–264, 1996.

[35] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, “Support vector machines using GMM supervectors for speaker verification,” *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, 2006.

[36] S. Young et al, “The HTK book,” http://htk.eng.cam.ac.uk, 2005.

[37] R. Collobert and S. Bengio, “SVMTorch: Support vector machines for large-scale regression problems,” *Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.

[38] M. Ferras, C. Barras, and J-L. Gauvain, “Lattice-based MLLR for speaker recognition,” in *Proc. ICASSP*, 2009.

[39] L. Ferrer, K. Sonmez, and E. Shriberg, “A smoothing kernel for spatially related features and its application to speaker verification,” in *Proc. Interspeech*, 2007.

[40] M. Brookes, “The matrix reference manual,” http://www.ee.ic.ac.uk/hp/staff/www/matrix/intro.html.

[41] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. I: Theory*, Birkhuser Boston, 1984.

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

[43] P. Sollich, “Probabilistic interpretation and bayesian methods for support vector machines,” in *Proceedings of ICANN*, 1999.

[44] O. Glembek, L. Burget, N. Dehak, N. Brummer, and P. Kenny, “Comparison of scoring methods used in speaker recognition with joint factor analysis,” in *Proc. ICASSP*, 2009.

[45] P. J. Moreno, P. P. Ho, and N. Vasconcelos, “A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications,” in *Adv. in Neural Inf. Proc. Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, Cambridge, MA, 2004.

[46] A. J. Laub, *Matrix Analysis for Scientists and Engineers*, SIAM, 2004.

[47] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990.

[48] S. Lucey and T. Chen, “Improved speaker verification through probabilistic subspace adaptation,” in *Proc. Interspeech*, 2003, pp. 2021–2024.

[49] R. Vogt, B. Baker, and S. Sriharan, “Modelling session variability in text-independent speaker verification,” in *Proc. Interspeech*, 2005, pp. 3117–3120.

[50] M. J. F. Gales, “Cluster adaptive training of hidden markov models,” *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 4, pp. 417–428, 2000.

[51] M. A. Przybocki, A. F. Martin, and A. N. Le, “NIST speaker recognition evaluations utilizing the Mixer corpora—2004,2005,2006,” *IEEE Trans. on Speech, Audio, Lang.*, vol. 15, no. 7, pp. 1951–1959, 2007.

[52] J. Odell, D. Ollason, P. Woodland, S. Young, and J. Jansen, *The HTK Book for HTK V2.0*, Cambridge University Press, Cambridge, UK, 1995.

[53] N. Dehak, R. Dehak, P. Kenny, N. Brummer, P. Ouellet, and P. Dumouchel, “Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification,” in *Proc. ICASSP*, 2009.

[54] Y. Zigel and M. Wasserblat, “How to deal with multiple-targets in speaker identification systems,” in *Proc. Odyssey*, 2006.

[55] J. B. Tenenbaum, V. Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[56] S. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, 2000.

[57] S. Young, Gunnar Evermann, Thomas Hain, D. Kershaw, Gareth Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK book*, Entropic, Ltd., Cambridge, UK, 2002.

[58] “The NIST year 2004 speaker recognition evaluation plan,” http://www.nist.gov/speech/tests/spk/2004/index.html, 2004.

[59] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[60] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling, Second Edition*, Chapman and Hall, 2000.

[61] J. Tenenbaum, “Matlab package for a global geometric framework for nonlinear dimensionality reduction,” http://isomap.stanford.edu/.

[62] D. Liben-Nowell and J. Kleinberg, “The link prediction problem for social networks,” in *Proc. 12th International Conference on Information and Knowledge Management*, 2003.

[63] S. White and P. Smyth, “Algorithms for estimating relative importance in networks,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[64] C.-C. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines,” http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[65] E. Adar, “Guess: A language and interface for graph exploration,” in *CHI*, 2006.

[66] D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for Visualization of Graphs*, Prentice Hall, 2002.

[67] N. Dehak, Z. Karam, D. Reynolds, R. Dehak, W. Campbell, and J. Glass, “A channel-blind system for speaker verification,” in *Proc. ICASSP*, 2011.

[68] G. J. McLachlan and K. E. Basford, *Mixture models : inference and applications to clustering*, M. Dekker, New York, N.Y., 1988.