38. Text-Independent Speaker Recognition

D. A. Reynolds, W. M. Campbell

In this chapter, we focus on the area of text-independent speaker verification, with an emphasis on unconstrained telephone conversational speech. We begin by providing a general likelihood ratio detection task framework to describe the various components in modern text-independent speaker verification systems. We next describe the general hierarchy of speaker information conveyed in the speech signal and the issues involved in reliably exploiting these levels of information for practical speaker verification systems. We then describe specific implementations of state-of-the-art text-independent speaker verification systems utilizing low-level spectral information and high-level token sequence information with generative and discriminative modeling techniques. Finally, we provide a performance assessment of these systems using the National Institute of Standards and Technology (NIST) speaker recognition evaluation telephone corpora.

38.1 Introduction

With the merging of telephony and computer networks, the growing use of speech as a modality in man–machine communication, and the need to manage ever increasing amounts of recorded speech in audio archives and multimedia applications, the utility of recognizing a person from his or her voice is increasing. While the area of speech recognition is concerned with extracting the linguistic message underlying a spoken utterance, speaker recognition is concerned with extracting the identity of the person speaking the utterance. Applications of speaker recognition are wide ranging, including: facility or computer access control [38.1,2], telephone voice authentication for long-distance calling or banking access [38.3], intelligent answering machines with personalized caller greetings [38.4], and automatic speaker labeling of recorded meetings for speaker-dependent audio indexing (speech skimming) [38.5, 6].

Depending upon the application, the general area of speaker recognition is divided into two specific tasks: identification and verification. In speaker identification, the goal is to determine which one of a group of known voices best matches the input voice sample. This is also referred to as *closed-set* speaker identification. Applications of pure closed-set identification are limited to cases where only enrolled speakers will be encountered, but it is a useful means of examining the separability of speakers' voices or finding similar sounding speakers, which has applications in speaker-adaptive speech recognition. In verification, the goal is to determine from a voice sample if a person is who he or she claims to be. This is sometimes referred to as the *open-set* problem, because this task requires distinguishing a claimed speaker's voice known to the system from a potentially large group of voices unknown to the system (i. e., impostor speakers). Verification is the basis

for most speaker recognition applications and the most commercially viable task. The merger of the closed-set identification and open-set verification tasks, called open-set identification, performs like closed-set identification for known speakers but must also be able to classify speakers unknown to the system into a *none of the above* category.

These tasks are further distinguished by the constraints placed on the speech used to train and test the system and the environment in which the speech is collected [38.7]. In a text-dependent system, the speech used to train and test the system is constrained to be the same word or phrase. In a text-independent system, the training and testing speech are completely unconstrained. Between text dependence and text independence, a vocabulary-dependent system constrains the speech to come from a limited vocabulary, such as the digits, from which test words or phrases (e.g., digit strings) are selected. Furthermore, depending upon the amount of control allowed by the application, the speech may be collected from a noise-free environment using a wide-band microphone or from a noisy, narrow-band telephone channel.

In this chapter, we focus on the area of text-independent speaker verification, with an emphasis on unconstrained telephone conversational speech. While many of the underlying algorithms employed in text-independent and text-dependent speaker verification are similar, text-independent applications have the additional challenge of operating unobtrusively to the user with little to no control over the user's behavior (i. e., the user is speaking for some other purpose, not to be verified, so will not cooperate to speak more clearly, use a limited vocabulary or repeat phrases). Further, the ability to apply text-independent verification to unconstrained speech encourages the use of audio recorded from a wide variety of sources (e.g., speaker indexing of broadcast audio or forensic matching of law-enforcement microphone recordings), emphasizing the need for compensation techniques to handle variable acoustic environments and recording channels.

This chapter is organized as follows. We begin by providing a general likelihood ratio detection task framework to describe the various components in modern text-independent speaker verification systems. We next describe the general hierarchy of speaker information conveyed in the speech signal and the issues involved in reliably exploiting these levels of information for practical speaker verification systems. We then describe specific implementations of state-of-the-art text-independent speaker verification systems utilizing low-level spectral information and high-level token sequence information with generative and discriminative modeling techniques. Finally we provide a performance assessment of these systems using the NIST speaker recognition evaluation telephone corpora.

38.2 Likelihood Ratio Detector

Given a segment of speech, Y, and a hypothesized speaker, S, the task of speaker detection, also referred to as verification, is to determine if Y was spoken by S. An implicit assumption often used is that Y contains speech from only one speaker. Thus, the task is better termed single-speaker detection. If there is no prior information that Y contains speech from a single speaker, the task becomes multispeaker detection. In this paper we will focus on the core single-speaker detection task. Discussion of systems that handle the multispeaker detection task can be found in [38.8].

The single-speaker detection task can be restated as a basic hypothesis test between

H_0 : Y is from the hypothesized speaker S

H_1 : Y is *not* from the hypothesized speaker S .

From statistical detection theory, the optimum test to decide between these two hypotheses is a likelihood ratio test given by

$$\frac{p(Y \mid H_0)}{p(Y \mid H_1)} \begin{cases} \geq \theta & \text{accept } H_0 \\ < \theta & \text{reject } H_0 \end{cases}, \tag{38.1}$$

where $p(Y \mid H_i), i = 0, 1$, is the probability density function for the hypothesis H_i evaluated for the observed speech segment Y, also referred to as the *likelihood* of the hypothesis H_i given the speech segment ($p(A \mid B)$ is referred to as a likelihood when B is considered the independent variable in the function). Strictly speaking, the likelihood ratio test is only optimal when the likelihood functions are known exactly, which is rarely the case. The decision threshold for accepting or rejecting H_0 is θ. Thus, the basic goal of a speaker detection system is to determine techniques to compute the likelihood ratio between the two likelihoods, $p(Y \mid H_0)$ and $p(Y \mid H_1)$. Depending upon the techniques used, these likelihoods

are explicitly or implicitly modeled and applied in the speaker detection systems.

Figure 38.1 shows the basic components found in speaker detection systems. The role of the front-end processing is to extract features from the speech signal that convey speaker-dependent information. In addition, techniques to minimize non-speaker-dependent effects from these features, such as linear filtering or additive noise, are also employed in the front-end processing. The output of this stage is a sequence of feature vectors representing the test segment, $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, where $\boldsymbol{x}_t$ is a feature vector indexed at discrete time $t \in [1, 2, \ldots, T]$. Features can be continuous or discrete with multiple dimensions, may be extracted at asynchronous time instances and can cover variable temporal durations. For example, the sequence of words spoken as estimated by a speech recognizer (series of one-dimensional discrete, asynchronous, variable temporal duration values), or the overall average pitch and energy in an utterance (single two-dimensional continuous-value vector) could be used as features. Thus, features can be extracted that capture short- and long-span speaker characteristics. Common features used in text-independent speaker detection systems are described in Sect. 38.3.

These feature vectors are then used to compute a score to evaluate the likelihood ratio test between H_0 and H_1. The exact way in which the hypothesis likelihood functions are represented and scored against the feature vectors depends on the classifier employed. The parameters of the classifier are obtained from the speaker enrollment or training phase, which is discussed in Sect. 38.4.

In generative modeling approaches, the likelihoods are explicitly represented by probability density models. Mathematically, H_0 is represented by a model denoted λ_{hyp}, which characterizes the hypothesized speaker S in the feature space of $\boldsymbol{x}$. For example, one could assume that the feature vectors for H_0 were generated from a Gaussian distribution so that λ_{hyp} would denote the mean vector and covariance matrix parameters of a Gaussian distribution. The alternative hypothesis, H_1, is similarly represented by the model $\lambda_{\overline{\text{hyp}}}$. The likelihood ratio statistic (or score) is then $p(X \mid \lambda_{\text{hyp}})/p(X \mid \lambda_{\overline{\text{hyp}}})$. Often, the logarithm of this statistic is used giving the log-likelihood ratio (LLR) score for the utterance,

$$\Lambda(X) = \log p(X \mid \lambda_{\text{hyp}}) - \log p(X \mid \lambda_{\overline{\text{hyp}}}) \,. \qquad (38.2)$$

In discriminative modeling approaches such as support vector machines (SVMs), the two likelihoods are not explicitly modeled separately, but a detection score similar to the log likelihood ratio score is computed (see Sect. 38.4.2). Although not precise mathematically, it is useful to use this likelihood ratio framework for both generative and discriminative approaches since it emphasizes the important roles of the speaker and alternative hypotheses in the modeling and detection process.

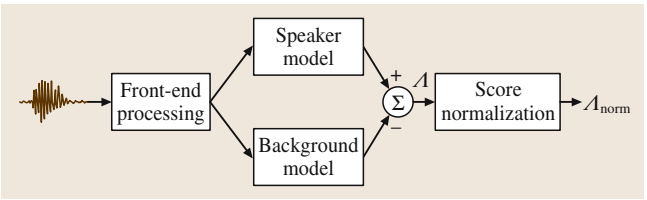

Fig. 38.1 Likelihood-ratio-based speaker detection system

After a likelihood ratio score is produced for the test speech and hypothesized speaker, it is often further processed by some form of score normalization and/or calibration. Despite mitigation steps taken during feature extraction (see Sect. 38.3.1) and model parameter estimation (see Sect. 38.4), the LLR score produced will be influenced by several nonspeaker factors, such as the spoken text in the test speech, differences in the microphones used and acoustic environment of the enrollment and test speech, and inaccuracies and assumptions in the models applied. Score normalization aims to compensate for these variabilities by removing score biases and scale factors estimated during enrollment. Several approaches have been proposed for score normalization, for example, Z-norm [38.9], H-norm [38.10], and T-norm [38.11], which compensate the LLR score as

$$\Lambda^{\text{norm}}(X) = \frac{\Lambda(X) - \mu(X, S)}{\sigma(X, S)} \,. \qquad (38.3)$$

The main difference in these score normalization techniques is how the biases, $\mu(X, S)$, and scale factors, $\sigma(X, S)$, are estimated. For example, in the H-norm, handset-microphone-dependent values are estimated, while in the T-norm scores from other speaker models are used to estimate these bias and scale factors.

A further step applied after or as part of score normalization is that of score calibration or confidence estimation. The aim in this process is to convert the LLR score values into *human-interpretable* values, such as posterior probabilities, with a defined range of, for example, (0–1). Confidence estimation techniques range from simple formula-based approaches that merely convert the LLR score to a posterior probability using a given prior probability, to more-sophisticated ap-

proaches that integrate in external signal measurement information, such as signal-to-noise measures, with the LLR score using artificial neural networks [38.12].

With this framework, we next describe specific examples of these components for modern text-independent speaker detection systems.

38.3 Features

One of the fundamental components in any speaker detection system is the features extracted from the speech signal that convey information about the speaker's identity. Unfortunately there is no single attribute of speech that conveys identity, rather it is conveyed through several different types or levels of information in the speech signal. These levels can roughly be categorized into a hierarchy running from low-level information, such as the sound of a person's voice, related to physical traits of the vocal apparatus, to high-level information, such as particular word usage (idiolect), related to learned habits, dialect, and style.

Features related to low-level information based on short-term spectral analysis are the dominant type used in text-independent speaker detection systems. This is primarily due to the computational ease of extracting these features and their proven effectiveness in terms of detection performance.

In recent years, there has been increased interest in exploiting high-level features for text-independent speaker detection systems. This has been spurred on by the continual advancement of phone and speech recognition systems used to extract features for high-level characterization. Work done at the 2002 SuperSID workshop [38.13] and subsequent related efforts demonstrated how systems using high- and low-level features could be successfully fused to improve overall accuracy. While high-level features are a promising new area of research in text-independent speaker recognition, they often have substantial computational costs (e.g., running a large-vocabulary speech recognizer), which need to be balanced against relatively modest accuracy gains.

38.3.1 Spectral Features

Several processing steps occur in the front-end analysis to extract spectral-based features. First, the speech is segmented into frames by a 20 ms window progressing at a 10 ms frame rate. A speech activity detector is then used to discard silence/noise frames. Typically, the speech detector discards 20–25% of the signal from conversational telephone speech.

Next, mel-scale cepstral feature vectors are extracted from the speech frames. The mel-scale cepstrum is the discrete cosine transform of the log-spectral energies of the speech segment Y. The spectral energies are calculated over logarithmically spaced filters with increasing bandwidths (*mel-filters*). A detailed description of the feature extraction steps can be found in [38.14]. For bandlimited telephone speech, cepstral analysis is usually performed only over the mel-filters in the telephone pass band (300–3400 Hz). All cepstral coefficients except the zeroth value (the DC level of the log-spectral energies) are retained in the processing. Lastly, delta-cepstra values are computed using a first-order derivative of an orthogonal polynomial temporal fit over ± 2 feature vectors (two to the left and two to the right over time) from the current vector [38.15].

Finally, the feature vectors are channel-normalized to remove linear channel convolutional effects. With cepstral features, convolutional effects appear as additive biases. Both cepstral mean subtraction (CMS) and RASTA filtering [38.16] have been used successfully and, in general, both methods have comparable performance for single-speaker detection tasks. When training and recognition speech are collected from different microphones or channels (e.g., different telephone handsets and/or lines), this is a crucial step for achieving good recognition accuracy. However, this linear compensation does not completely eliminate the performance loss under mismatched microphone conditions, so more-sophisticated compensation techniques such as feature mapping [38.17], where transformations are trained to map features coming from particular channels (e.g., microphones) into channel independent features, are also applied.

There are many variants of these cepstral features, such as using linear prediction coding (LPC) spectral analysis instead of fast Fourier transform (FFT)-based spectral analysis, but the basic steps are similar and performance does not vary significantly.

38.3.2 High-Level Features

For high-level feature extraction, input speech is converted into a series of tokens, $T = \{t_i\}$. The tokens are time-ordered discrete symbols and represent linguistically significant interpretations of the input signal.

Examples of token types are words, phones, and pitch gestures.

High-level feature extraction involves many tradeoffs. First, designing token extraction systems is, in general, a difficult process. If the high-level features are specific to the language of interest, then significant effort may be required to implement the system. For example, speech-to-text (STT) systems such as the BBN Byblos system [38.18] require training using large corpora in the language of interest and significant engineering effort to produce high-accuracy output. A second consideration in token extraction is how it will be used with other speaker recognition recognition systems. Some token systems, such as STT, have been shown to provide significant fusion gains when combined with standard acoustic systems [38.19], but have low speaker recognition accuracy alone. Other token systems, such as phone token [38.20], have excellent standalone accuracy, but do not provide significant complimentary information to standard spectral methods. A third consideration for token system selection is length of enrollment and verification. A token system that produces tokens at a low rate, e.g., words, requires multi-conversation enrollment to produce good results. Other higher rate tokens, e.g., phones, require less enrollment for equivalent accuracy.

Modeling of the token stream is usually accomplished by computing probabilities of n-grams of token contexts [38.21] and then applying a classifier. We discuss two methods based on SVMs and standard likelihood ratio techniques in Sect. 38.4.3.

38.4 Classifiers

In this section, we describe popular and successful generative and discriminative classifiers used for both high- and low-level features. First we present spectral-feature-based classifiers using adapted Gaussian mixture models and sequence-based support vector machines. This is followed by high-level (token sequence) feature based n-gram and SVM classifiers.

38.4.1 Adapted Gaussian Mixture Models

Gaussian Mixture Models

An important step in the implementation of the likelihood ratio detector of (38.2) is selection of the actual likelihood function, $p(X|\lambda)$. The choice of this function is largely dependent on the features being used as well as specifics of the application. For text-independent speaker recognition using continuous features, where there is no prior knowledge of what the speaker will say, the most successful likelihood function has been Gaussian mixture models (GMMs). In text-dependent applications, where there is strong prior knowledge of the spoken text, additional temporal knowledge can be incorporated by using hidden Markov models (HMMs) as the basis for the likelihood function. To date, however, use of more-complicated likelihood functions, such as those based on HMMs, have shown no advantage over GMMs for text-independent speaker detection tasks. As more-accurate knowledge of the spoken text is gained via speech recognition, lessening the distinction between text-dependent and text-independent tasks, this may change.

For a D-dimensional feature vector, $\boldsymbol{x}$, the mixture density used for the likelihood function is defined as

$$p(\boldsymbol{x}|\lambda) = \sum_{i=1}^{M} w_i \; p_i(\boldsymbol{x}) \; . \tag{38.4}$$

The density is a weighted linear combination of M unimodal Gaussian densities, $p_i(\boldsymbol{x})$, each parameterized by a mean $D \times 1$ vector, $\boldsymbol{\mu}_i$, and a $D \times D$ covariance matrix, Σ_i;

$$p_i(\boldsymbol{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} \times \exp\left[-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_i)^{\mathrm{T}} \; \Sigma_i^{-1} \; (\boldsymbol{x} - \boldsymbol{\mu}_i) \right] . \tag{38.5}$$

The mixture weights, w_i, furthermore satisfy the constraint $\sum_{i=1}^{M} w_i = 1$. Collectively, the parameters of the density model are denoted by $\lambda = \{w_i, \boldsymbol{\mu}_i, \Sigma_i\}$, where $i = 1, \ldots, M$.

While the general model form supports full covariance matrices, i. e., a covariance matrix with all its elements, usually only diagonal covariance matrices are used. This is done for three reasons. First, the density modeling of an M^{th}-order full covariance GMM can equally well be achieved using a larger-order diagonal covariance GMM. GMMs with $M > 1$ using diagonal covariance matrices can model distributions of feature vectors with correlated elements. Only in the degenerate case of $M = 1$ is the use of a diagonal covariance matrix incorrect for feature vectors with correlated elements. Second, diagonal-matrix

GMMs are more computationally efficient than full-covariance GMMs for training since repeated inversions of a $D \times D$ matrix are not required. Third, empirically we have observed that diagonal-matrix GMMs outperform full-matrix GMMs.

Given a collection of training vectors, maximum-likelihood model parameters are estimated using the iterative expectation-maximization (EM) algorithm [38.22]. The EM algorithm iteratively refines the GMM parameters to increase the likelihood of the estimated model for the observed feature vectors monotonically, i. e., for iterations k and $k+1$, $p(X|\lambda^{(k+1)}) > p(X|\lambda^{(k)})$. Generally, five iterations are sufficient for parameter convergence. The EM equations for training a GMM can be found in [38.23].

The feature vectors of X are assumed independent, so the log-likelihood of a model λ for a sequence of feature vectors, $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, is computed as

$$\log p(X \mid \lambda) = \sum_{t=1}^{\mathrm{T}} \log p(\boldsymbol{x}_t \mid \lambda) \,, \tag{38.6}$$

where $p(\boldsymbol{x}_t \mid \lambda)$ is computed as in (38.4). Often, the average log-likelihood value is used, by dividing $\log p(X \mid \lambda)$ by T. This is done to normalize out duration effects from the log-likelihood value. Since the incorrect assumption of independence is underestimating the actual likelihood value with dependencies, this scaling factor can also be considered a rough compensation factor to the likelihood value in (38.6).

The GMM can be viewed as a hybrid between a parametric and nonparametric density model. Like a parametric model it has structure and parameters that control the behavior of the density in known ways, but without constraints that the data must be of a specific distribution type, such as Gaussian or Laplacian. Like a nonparametric model, the GMM has many degrees of freedom to allow arbitrary density modeling, without undue computation and storage demands. It can also be thought of as a single-state HMM with a Gaussian mixture observation density, or an ergodic Gaussian observation HMM with fixed, equal transition probabilities. Here, the Gaussian components can be considered to be modeling the underlying broad phonetic sounds that characterize a person's voice. A more-detailed discussion of how GMMs apply to speaker modeling can be found in [38.24].

The advantages of using a GMM as the likelihood function are that it is computationally inexpensive, is based on a well-understood statistical model, and, for text-independent tasks, is insensitive to the temporal aspects of the speech, modeling only the underlying distribution of acoustic observations from a speaker. The latter is also a disadvantage in that higher-levels of information about the speaker conveyed in the temporal speech signal are not used.

Universal Background Model

We next describe how the GMM described above is used to build the likelihood ratio detector in (38.2). While the model for H_0, λ_{hyp}, is well defined and is estimated using training speech from S, the model for the alternative hypothesis, $\lambda_{\overline{\mathrm{hyp}}}$, is less well defined since it potentially must represent the entire space of possible alternatives to the hypothesized speaker. Two main approaches have been taken for this alternative hypothesis modeling. The first approach is to use a set of other speaker models to cover the space of the alternative hypothesis. In various contexts, this set of other speakers has been called likelihood ratio sets [38.2], cohorts [38.25], and background speakers [38.23]. Given a set of N background speaker models $\{\lambda_1, \ldots, \lambda_N\}$, the alternative hypothesis model is represented by

$$p(X \mid \lambda_{\overline{\mathrm{hyp}}}) = \mathcal{F}[p(X \mid \lambda_1), \ldots, p(X \mid \lambda_N)], \tag{38.7}$$

where $\mathcal{F}()$ is some function, such as the average or maximum, of the likelihood values from the background speaker set. The selection, size, and combination of the background speakers has been the subject of much research. In general, it has been found that to obtain the best performance with this approach requires the use of speaker-specific background speaker sets. This can be a drawback in an applications using a large number of hypothesized speakers, each requiring their own background speaker set.

The second major approach to alternative hypothesis modeling is to pool speech from several speakers and train a single model. Various terms for this single model are a general model [38.26], a world model, and a universal background model [38.27]. Given a collection of speech samples from a large number of speakers representative of the population of speakers expected during recognition, a single model λ_{bkg} is trained to represent the alternative hypothesis. Research on this approach has focused on selection and composition of the speakers and speech used to train the single model. The main advantage of this approach is that a single speaker-independent model can be trained once for a particular task and then used for all hypothesized speakers in that task. It is also possible to use multiple background models tailored to specific sets of speakers. Overall,

the use of a single background model [which we will refer to as the universal background model (UBM)], is the dominant approach used in text-independent speaker detection systems.

The UBM is a large GMM trained to represent the speaker-independent distribution of features. Specifically, we want to select speech that is reflective of the expected alternative speech to be encountered during recognition. This applies to both the type and quality of speech, as well as the composition of speakers. For example, in the NIST speaker recognition evaluation (SRE) single-speaker detection tests, it is known a priori that the speech comes from local and long-distance telephone calls, and that male hypothesized speakers will only be tested against male speech. In this case, we would train the UBM used for male tests using only male telephone speech. In the case where there is no prior knowledge of the gender composition of the alternative speakers, we would train using gender-independent speech.

Other than these general guidelines and experimentation, there is no objective measure to determine the right number of speakers or amount of speech to use in training a UBM. Empirically, from the NIST SRE we have observed no performance loss using a UBM trained with one hour of speech compared to one trained using six hours of speech. In both cases, the training speech was extracted from the same speaker population.

Adaptation of Speaker Model

In the GMM UBM system, the speaker model is derived by adapting the parameters of the UBM using the speaker's training speech and a form of Bayesian adaptation [38.28]. This is also known as Bayesian learning or *maximum a posteriori* (MAP) estimation. We use the term Bayesian adaptation since, as applied to the speaker-independent UBM to estimate the speaker-dependent model, the operation closely resembles speaker adaptation used in speech recognition applications. Unlike the standard approach of maximum-likelihood training of a model for the speaker independently of the UBM, the basic idea in the adaptation approach is to derive the speaker's model by updating the well-trained parameters in the UBM via adaptation. This provides a tighter coupling between the speaker's model and UBM, which not only produces better performance than decoupled models, but as discussed later in this section, also allows for a fast scoring technique. Like the EM algorithm, the adaption is a two-step estimation process. The first step is identical to the *expectation* step of the EM algorithm, where estimates of the sufficient statistics of the speaker's training data are computed for each mixture in the UBM. For a GMM mixture, these are the count, and the first and second moments required to compute the mixture weight, mean and variance. Unlike the second step of the EM algorithm, for adaptation these *new* sufficient statistic estimates are then combined with the *old* sufficient statistics from the UBM mixture parameters using a data-dependent mixing coefficient. The data-dependent mixing coefficient is designed so that mixtures with high counts of data from the speaker rely more on the new sufficient statistics for final parameter estimation and mixtures with low counts of data from the speaker rely more on the old sufficient statistics for final parameter estimation.

While it is possible to adapt all the parameters of the GMM during speaker model training, it has been widely shown that speaker detection performance is best when only the mean parameters are adapted [38.11, 29, 30]. Here, we will focus only on mean adaptation but details of adapting all parameters can be found in [38.30].

Given a UBM and training vectors from the hypothesized speaker, $X = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_T\}$, we first determine the probabilistic alignment of the training vectors into the UBM mixture components (Fig. 38.2a). That is, for the mixture i in the UBM, we compute

$$\Pr(i|\boldsymbol{x}_t) = \frac{w_i p_i(\boldsymbol{x}_t)}{\sum_{j=1}^{M} w_j p_j(\boldsymbol{x}_t)} \,. \tag{38.8}$$

We then use $\Pr(i|\boldsymbol{x}_t)$ and $\boldsymbol{x}_t$ to compute the sufficient statistics for the mean parameter,

$$n_i = \sum_{t=1}^{\mathrm{T}} \Pr(i|\boldsymbol{x}_t), \tag{38.9}$$

$$E_i(\boldsymbol{x}) = \frac{1}{n_i} \sum_{t=1}^{\mathrm{T}} \Pr(i|\boldsymbol{x}_t)\boldsymbol{x}_t \,. \tag{38.10}$$

This is the same as the *expectation* step in the EM algorithm.

Lastly, these new sufficient statistics from the training data are used to update the old UBM sufficient statistics for the mixture i to create the adapted mean parameter for the mixture i (Fig. 38.2b):

$$\hat{\boldsymbol{\mu}}_i = \alpha_i E_i(\boldsymbol{x}) + (1-\alpha_i)\boldsymbol{\mu}_i \,. \tag{38.11}$$

The data-dependent adaptation coefficients controlling the balance between old and new estimates per mixture are $\{\alpha_i\}$, defined as

$$\alpha_i = \frac{n_i}{n_i + r} \,. \tag{38.12}$$

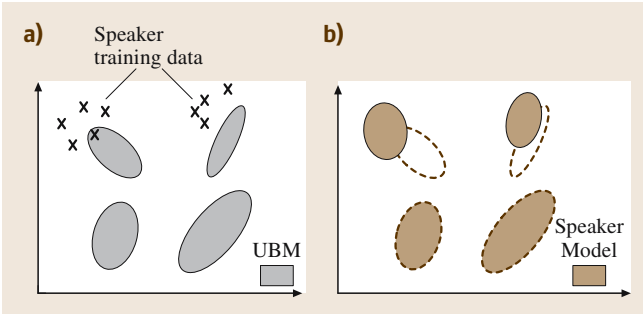

Fig. 38.2a,b Pictorial example of two steps in adapting a hypothesized speaker model. **(a)** The training vectors (*crosses*) are probabilistically mapped into the UBM mixtures. **(b)** The adapted mixture parameters are derived using the statistics of the new data and the UBM mixture parameters. The adaptation is data dependent, so UBM mixture parameters are adapted by different amounts

Here r is a fixed *relevance* factor that controls the amount of adaption from the UBM (e.g., higher values of r require more speaker data in a mixture component to adapt from the prior UBM mean).

The parameter updating as described in (38.11–38.12) can be derived from the general MAP estimation equations for a GMM using constraints on the prior distribution described in [38.28].

Using a data-dependent adaptation coefficient allows mixture-dependent adaptation of parameters. If a mixture component has a low probabilistic count, n_i, of new data, then $\alpha_i \to 0$ causing the de-emphasis of the new (potentially undertrained) parameters and the emphasis of the old (better-trained) parameters. For mixture components with high probabilistic counts, $\alpha_i \to 1$, causing the use of the new speaker-dependent parameters. The relevance factor is a way of controlling how much new data should be observed in a mixture before the new parameters begin replacing the old parameters. Values of r in the range 8–20 have been shown to work well for several speaker recognition tasks.

Comparison of published results strongly indicate that the adaptation approach provides superior performance over a decoupled system where the speaker model is trained independently of the UBM. One possible explanation for the improved performance is that the use of adapted models in the likelihood ratio is not affected by *unseen* acoustic events in recognition speech. Loosely speaking, if one considers the UBM as covering the space of speaker-independent, broad acoustic classes of speech sounds, then adaptation is the speaker-dependent *tuning* of those acoustic classes observed in the speaker's training speech. Mixture parameters for those acoustic classes not observed in the training speech are merely copied from the UBM. This means that, during recognition, data from acoustic classes unseen in the speaker's training speech produce approximately zero log-likelihood ratio values that contribute evidence neither toward nor against the hypothesized speaker. Speaker models trained using only the speaker's training speech will have low likelihood values for data from classes not observed in the training data, thus producing low likelihood ratio values. While this is appropriate for speech not from the speaker, it clearly can cause incorrect values when the unseen data occurs in test speech from the speaker.

Log-Likelihood Ratio Computation

The log-likelihood ratio for a test sequence of feature vectors X is computed as $\Lambda(X) = \log p(X|\lambda_{\text{hyp}}) - \log p(X|\lambda_{\text{ubm}})$. The fact that the hypothesized speaker model was adapted from the UBM, however, allows a faster scoring method than merely evaluating the two GMMs as in (38.6). This fast scoring approach is based on two observed effects. The first is that, when a large GMM is evaluated for a feature vector, only a few of the mixtures contribute significantly to the likelihood value. This is because the GMM represents a distribution over a large space but a single vector will be near only a few components of the GMM. Thus, likelihood values can be approximated very well using only the top-C best scoring mixture components.

The second observed effect, is that the components of the adapted GMM retain a correspondence with the mixtures of the UBM, so that vectors close to a particular mixture in the UBM will also be close to the corresponding mixture in the speaker model. Using these two effects, a fast scoring procedure operates as follows: for each feature vector, determine the top-C scoring mixtures in the UBM and compute the UBM likelihood using only these top-C mixtures. Next, score the vector against only the corresponding C components in the adapted speaker model to evaluate the speaker's likelihood. For a UBM with M mixtures, this requires only $M + C$ Gaussian computations per feature vector compared to $2M$ Gaussian computations for normal likelihood ratio evaluation. When there are multiple hypothesized speaker models for each test segment, the savings become even greater. Typical values of C are in the range 1–5.

GMM Compensation

In addition to compensation techniques applied in the feature and score domains, there are also many effec-

tive compensation techniques that can be applied in the model domain. For GMM-based classifiers, techniques that treat the undesired variability as a bias to the mean vectors have been shown to be very promising in improving the robustness of speaker recognition system. If we stack the means from a GMM into a *supervector* this can be written as

$$\boldsymbol{m}_j(s) = \boldsymbol{m}(s) + \boldsymbol{c}(s), \tag{38.13}$$

where $\boldsymbol{m}_j(s)$ is the supervector from speaker s's j-th enrollment session, $\boldsymbol{m}(s)$ is the desired compensated supervector for speaker s and $\boldsymbol{c}(s)$ is the undesired variability supervector. The main difference in the compensation techniques is in how they estimate and remove the variability vector $\boldsymbol{c}(s)$.

In speaker model synthesis (SMS) [38.31], the difference between bias vectors from a set of predefined channel types is used to synthetically generate a library of channel-dependent speaker models so as to allow matched-channel likelihood ratio scoring during recognition. Feature mapping [38.17], which is applied in the feature domain, is similar to SMS, but instead of synthetically creating unseen channel-dependent speaker models, acts to subtract off the channel-dependent biases in training and testing from the feature vectors directly.

More-recent latent factor analysis (LFA)-based techniques [38.32, 33], model the supervector bias as a low-dimensional normal bias,

$$\boldsymbol{c}(s) = U\boldsymbol{n}(s)\,, \tag{38.14}$$

where U is the low-rank session loading matrix. The LFA techniques are aimed specifically at compensation of session variability and do not require prior channel detectors or parameters. During training, the session bias is removed from each enrollment session and the session-independent supervectors are combined to produce the final speaker model. During recognition, the session bias for the test utterance is estimated and added to the speaker model to allow for matched session likelihood ratio scoring. A feature-space dual of LFA has also been developed and used successfully [38.34].

38.4.2 Support Vector Machines

A support vector machine (SVM) is a powerful classifier that has gained considerable popularity in recent years. An SVM is discriminative – it models the boundary between a speaker and a set of impostors. This approach contrasts to traditional methods for speaker recognition which separately model the probability distributions of the speaker and the general population. SVMs are based upon the structural risk minimization principal of *Vapnik* [38.35].

Two broad approaches using support vector machines have been proposed in the literature for speech applications. The first set of methods attempts to model emission probabilities for hidden Markov models [38.36–38]. This approach has been moderately successful in reducing error rates, but suffers from several problems. First, large training sets result in long training times for support vector methods. Second, the emission probabilities must be approximated [38.39], since the output of the support vector machine is not a probability. This approximation is needed to combine probabilities using the standard frame-independence assumption used in HMM speaker recognition.

A second set of methods is based upon comparing speech utterances using sequence kernels. Rather than model features from individual frames of speech, these methods instead model the entire sequence of feature vectors. Approaches include the generalized linear discriminant sequence kernel [38.40], Fisher kernel methods [38.41, 42], n-gram kernels [38.20], maximum-likelihood linear regression (MLLR) transform kernels [38.43], and GMM supervector kernels [38.44]. We focus on these latter methods, which have been successful in speaker recognition because of their accuracy and simplicity of implementation.

Basic SVM Theory

An SVM [38.35] models two classes using sums of a kernel function $K(\cdot,\cdot)$,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i t_i K(\boldsymbol{x}, \boldsymbol{x}_i) + d, \tag{38.15}$$

where the t_i are the ideal outputs, $\sum_{i=1}^{N} \alpha_i t_i = 0$, and $\alpha_i > 0$. The vectors $\boldsymbol{x}_i$ are support vectors and obtained from the training set by an optimization process [38.45]. 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, $f(\boldsymbol{x})$, is above or below a threshold.

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

$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{b}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{y}), \tag{38.16}$$

where $\boldsymbol{b}(\boldsymbol{x})$ is a mapping from the input space (where $\boldsymbol{x}$ lives) to a possibly infinite-dimensional *expansion*

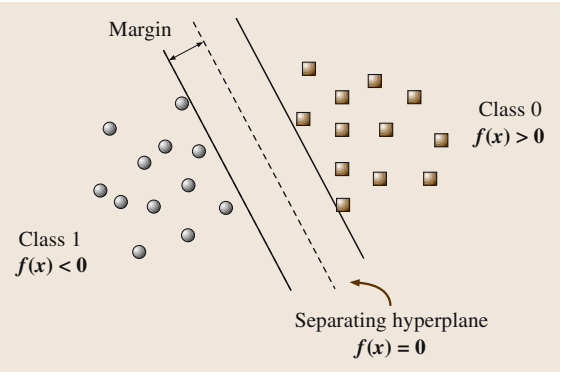

Fig. 38.3 Support vector machine concept

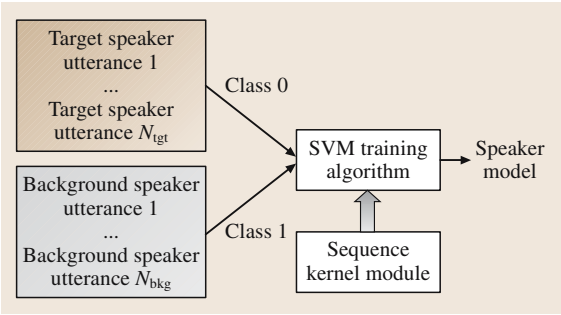

Fig. 38.4 SVM training strategy

space. The kernel is required to be positive semidefinite. The Mercer condition ensures that the margin concept is valid, and the optimization of the SVM is bounded.

The optimization condition relies upon a maximum margin concept (Fig. 38.3). For a separable data set, the system places a hyperplane in a high dimensional space so that the hyperplane has maximum margin. The data points from the training set lying on the boundaries (as indicated by solid lines in the figure) are the support vectors in (38.15). The focus, then, of the SVM training process is to model the boundary, as opposed to a traditional GMM UBM, which would model the probability distributions of the two classes.

Application of Support Vector Machines to Speaker Recognition

Discriminative training of an SVM for speaker recognition is straightforward, but several basic issues must be addressed: handling multiclass data, *world* modeling, and sequence comparison. The first two topics are handled in this section.

Since the SVM is a two-class classifier, we must recast speaker verification and identification in this structure. For speaker verification, a target model is trained for the speaker against a set of example speakers that have characteristics of the impostor population (Fig. 38.4). The set of example impostors is called a background set. In the figure, class 0 consists of all of the target speaker's utterances, and class 1 consists of the example impostors in the background set. The background speaker set is kept the same as we enroll different target speakers.

Figure 38.4 indicates the basic training strategy for SVMs using sequence kernels. Each utterance from a target or background speaker becomes a point in the SVM expansion space, see (Fig. 38.3). A sequence kernel module for comparing two utterances and producing a kernel value is implemented. The kernel module is connected into a standard SVM training tool that then produces a speaker model.

For speaker identification, a *one-versus-all* strategy is used. For a given target speaker, we pool all of the remaining nontarget speakers into class 1 and then train a speaker model. This operation is performed for all models to obtain our set of speaker identification models.

For speaker verification, the support vectors have an interesting interpretation. If $f(\boldsymbol{x})$ is an SVM for a target speaker, then we can write

$$f(\boldsymbol{x}) = \sum_{i\in\{i|t_i=1\}} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) - \sum_{i\in\{i|t_i=-1\}} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + d. \quad (38.17)$$

The first sum can be interpreted as a per-utterance-weighted target score. The second sum has many of the characteristics of a cohort score [38.25] with some subtle differences. First, utterances rather than speakers are used as cohorts. Second, the weighting on these *cohort* utterances is not equal.

The interpretation of the SVM score as a cohort-normalized score suggests how the *world* of impostors should be modeled. Our background should have a rich speaker set, so that we can always find speakers close to the target speaker. The speakers in the background should have similar attributes – language, accent, channel type, etc. – for best performance. Note that this interpretation distinguishes the SVM approach from a universal background model method [38.30], which tries to model the impostor set with one model. Other methods for GMMs including cohort normalization [38.25] and T-norm [38.11] are closer to the SVM method; although, the latter method (T-norm) typ-

ically uses a fixed set of cohorts rather than picking out individual speakers.

Sequence Kernels for Speaker Recognition – General Structure

To apply an SVM, $f(\boldsymbol{x})$, to a speaker recognition application, we need a method for calculating kernel values from speech inputs. For recognition, a way is needed for taking a sequence of input feature vectors from an utterance, $\{\boldsymbol{x}_i\}$, and computing the SVM output, $f(\{\boldsymbol{x}_i\})$. Typically, each vector $\boldsymbol{x}_i$ would be the cepstral coefficients and deltas for a given frame of speech. One way of handling this situation is to assume that the kernel, $K(\cdot,\cdot)$, in the SVM (38.15) takes sequences as inputs, i. e., we can calculate $K(\{\boldsymbol{x}_i\},\{\boldsymbol{y}_j\})$ for two input sequences $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_j\}$. This is called a sequence kernel method.

A challenge in applying the sequence kernel method is deriving a function for comparing sequences. A function is needed that, given two utterances, produces a measure of similarity of the speakers or languages. Also, a method that is efficient computationally is needed, since we will be performing many kernel inner products during training and scoring. Finally, the kernel must satisfy the Mercer condition.

One idea for constructing a sequence kernel is illustrated in Fig. 38.5. The basic approach is to compare two speech utterances, *utt 1* and *utt 2* by training a model on one utterance and then scoring the resulting model on another utterance. This process produces a value that measures the similarity between the two utterances. Two questions that follow from this approach are:

1. Can the train/test process be computed efficiently?
2. Is the resulting comparison a kernel (i. e., does it satisfy the Mercer condition)?

The answer to both of these questions is yes if the appropriate classifier is chosen – see next section.

Another idea for constructing sequence kernels is shown in Fig. 38.6. In this setup, a base model is adapted to obtain probability distributions which represent the utterances. A model comparison algorithm is then applied to get a measure of similarity. This approach has the useful property that it is naturally symmetric as long as the comparison calculation is symmetric.

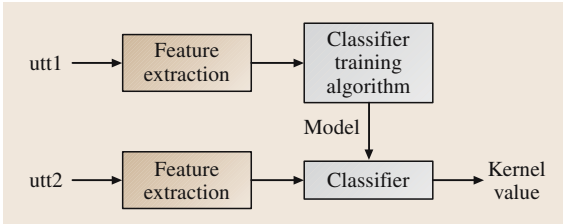

Fig. 38.5 General train/test sequence kernel

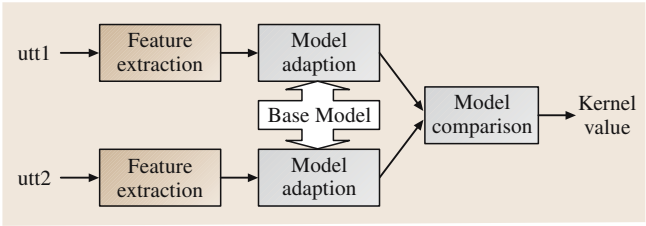

Fig. 38.6 Generative model sequence kernel

Sequence Kernels for Speaker Recognition – Specific Examples

For the train/test kernel shown in Fig. 38.5, a typical approach is the generalized linear discriminant sequence (GLDS) kernel [38.40]. In this method, the classifier is taken to be a polynomial discriminant function consisting of the monomials up to a certain degree; e.g., for two features, x_1 and x_2, and degree 2 the monomials are,

$$\boldsymbol{b}(\boldsymbol{x}) = \left(1, x_1, x_2, x_1^2, x_1x_2, x_2^2\right)^{\mathrm{T}}. \tag{38.18}$$

Suppose we have two sequences of feature vectors, $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_j\}$. If a polynomial discriminant is trained using mean-squared error, then the resulting kernel is given by

$$K(\{\boldsymbol{x}_i\},\{\boldsymbol{y}_j\}) = \bar{b}_x^{\mathrm{T}}\bar{R}^{-1}\bar{b}_y. \tag{38.19}$$

In (38.19),

$$\bar{b}_x = \frac{1}{N_x}\sum_i \boldsymbol{b}(\boldsymbol{x}_i)\,, \tag{38.20}$$

i. e., $\bar{b}_x$ is the average expansion over all frames. The quantity $\bar{b}_y$ is defined similarly for the sequence, $\{\boldsymbol{y}_j\}$. The matrix, $\bar{R}$ is the correlation matrix of a background data set; typically, it is approximated with only diagonal terms. For details on the derivation of these equations, refer to [38.40].

An interesting generalization of the GLDS kernel is to replace the polynomial expansion by a general kernel using the kernel trick [38.46]. This idea leads to a method of transforming frame-level kernels into sequence kernels.

The GLDS kernel in (38.19) has the properties expected for a sequence kernel. The kernel is a single value from a sequence of vectors, symmetric, and positive definite. An interesting property of the kernel is that the kernel itself is a classifier. In a certain sense, the

SVM method acts as a method of strengthening a weaker classifier.

For the generative model sequence kernel, several methods have been proposed. Current methods are based upon adapting from a GMM or HMM base model. In [38.47], GMMs are adapted to the utterances, and an exponentiated Kullbach Leibler (KL) divergence is used to compare the utterances. In [38.43], adaptation of an HMM from a speech-to-text system is performed using maximum-likelihood linear regression (MLLR). The MLLR adaptation parameters are then compared with a weighted linear inner product. In [38.44], the adaptation is performed via MAP adaptation of a GMM. In this GMM supervector (GSV) system, the GMMs are compared using either an approximation to the KL divergence or an integral inner product. Finally [38.42] also fits loosely into this structure. A GMM model is trained from all utterance data. Then, the comparison is performed using both the base model and the target model via a modified Fisher kernel method [38.42, 48].

Nuisance Attribute Projection (NAP)

As with the GMM, compensations with SVM classifiers can also be applied directly in the model domain. The SVM nuisance attribute projection (NAP) method [38.49] works by removing subspaces that cause variability in the kernel. NAP constructs a new kernel,

$$\begin{aligned} K(\{\boldsymbol{x}_i\},\{\boldsymbol{y}_j\}) &= \left[P\bar{b}_x\right]^{\mathrm{T}}\left[P\bar{b}_y\right] \\ &= \bar{b}_x^{\mathrm{T}} P \bar{b}_y \\ &= \bar{b}_x^{\mathrm{T}}(I-\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}})\bar{b}_y, \end{aligned} \tag{38.21}$$

where P is a projection ($P^2 = P$), $\boldsymbol{v}$ is the direction being removed from the SVM expansion space, $\boldsymbol{b}(\cdot)$ is the SVM expansion, and $\|\boldsymbol{v}\|_2 = 1$. The design criterion for P and correspondingly $\boldsymbol{v}$ is

$$\boldsymbol{v}^* = \underset{\boldsymbol{v},\|\boldsymbol{v}\|_2=1}{\operatorname{argmin}} \sum_{i,j} W_{i,j} \left\| P\bar{b}_{z,i} - P\bar{b}_{z,j} \right\|_2^2 , \tag{38.22}$$

where $\bar{b}_{z,i}$ is the average expansion (38.20) of the i-th sequence in a background data set. Here, $W_{i,j}$ can be selected in several different ways. If we have channel nuisance variables (e.g., electret, carbon button, cell) and a labeled background set, then we can pick $W_{i,j} = 0$ when the channels of $\bar{b}_{z,i}$ and $\bar{b}_{z,j}$ are the same, and $W_{i,j} = 1$ otherwise. Another criterion is to design the projection based on session variability. In this case, we pick $W_{i,j} = 1$ if $\bar{b}_{z,i}$ and $\bar{b}_{z,j}$ correspond to the same speaker, and $W_{i,j} = 0$ otherwise. The idea in both cases is to reduce variability in the SVM kernel distance with respect to nuisances – channel or session. See [38.44] for more details and the relationship to LFA compensation for GMMs.

Note that, while NAP is described above in the context of spectral based features, it can also be applied when using high-level token features described in the next section.

38.4.3 High-Level Feature Classifiers

In this section, we consider methods for modeling a token stream as shown in Fig. 38.7. Two standard approaches have emerged – log-likelihood ratio and support vector machine methods.

Log-Likelihood Ratio Classifier

A token sequence can be modeled using a standard *bag of n-grams* approach [38.21, 50]. For a token sequence, n-grams are produced by the standard transformation of the token stream; e.g., for bigrams, the sequence of symbols from a conversation $t_1, t_2, \ldots, t_n$ is transformed to the sequence of bigrams of symbols $t_1\_t_2, t_2\_t_3 \ldots t_{n-1}\_t_n$. Probabilities of n-grams are then found with n fixed. That is, suppose unigrams and bigrams of symbols are considered, and the unique unigrams and bigrams of the corpus are designated $d_1 \ldots d_M$ and $d_1\_d_1, \ldots, d_M\_d_M$, respectively; then calculate probabilities

$$\begin{aligned} p(d_i) &= \frac{\#(t_k = d_i)}{\sum_l \#(t_k = d_l)}, \\ p(d_i\_d_j) &= \frac{\#(t_k\_t_{k+1} = d_i\_d_j)}{\sum_{l,m} \#(t_k\_t_{k+1} = d_l\_d_m)}, \end{aligned} \tag{38.23}$$

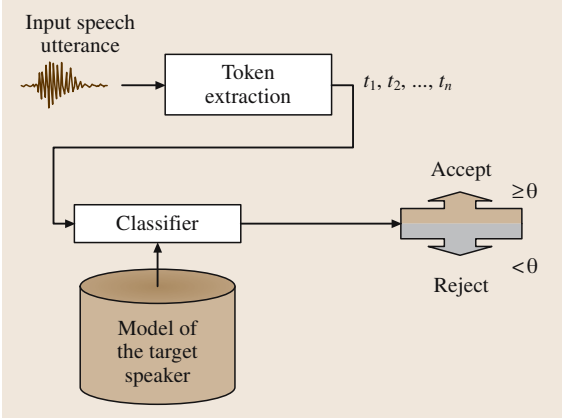

Fig. 38.7 High-level speaker recognition: basic setup

where $\#(t_k = d_i)$ indicates the number of symbols in the conversation equal to d_i, and an analogous definition is used for bigrams.

For simplicity, suppose that we have two utterances from two speakers, spk_1 and spk_2. Further suppose that the sequence of n-grams (for fixed n) in each conversation is $t_1, t_2 \ldots t_n$ and $u_1, u_2 \ldots u_m$ respectively. Denote the unique set of n-grams in the corpus as $d_1, \ldots, d_M$. A model can be built based upon the conversations for each speaker consisting of the probability of n-grams, $p(d_i|\text{spk}_j)$. The average log-likelihood ratio of the first conversation is computed as,

$$\begin{aligned}\text{score} &= \frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{p(t_i|\text{spk}_2)}{p(t_i|\text{bkg})}\right)\\ &= \sum_{j=1}^{M}\frac{\#(t_i = d_j)}{n}\log\left(\frac{p(d_j|\text{spk}_2)}{p(d_j|\text{bkg})}\right)\\ &= \sum_{j=1}^{M} p(d_j|\text{spk}_1)\log\left(\frac{p(d_j|\text{spk}_2)}{p(d_j|\text{bkg})}\right), \end{aligned} \quad (38.24)$$

where $p(d_j|\text{bkg})$ is the probability of an n-gram in some large background data set. This scoring (38.24) can be easily generalized to multiple utterances per speaker; we just estimate the probabilities $p(d_j|\text{spk}_i)$ from a larger data set. Further details on this scoring method can be found in [38.21, 50].

SVM-Based Scoring

The score (38.24) can be transformed into a sequence kernel via a simple linearization; using $\log(x) \approx x - 1$ (the Taylor series expansion around $x = 1$), gives

$$\begin{aligned}\text{score} &\approx \sum_{j=1}^{M} p(d_j|\text{spk}_1)\frac{p(d_j|\text{spk}_2)}{p(d_j|\text{bkg})} - 1\\ &= \sum_{j=1}^{M}\frac{p(d_j|\text{spk}_1)}{\sqrt{p(d_j|\text{bkg})}}\frac{p(d_j|\text{spk}_2)}{\sqrt{p(d_j|\text{bkg})}} - 1. \end{aligned} \quad (38.25)$$

Once we have a sequence kernel, an SVM is trained in the same manner as the spectral-based SVMs in Fig. 38.4.

In analogy with the information retrieval literature [38.51], the result in (38.25) can be expressed in vector form. First, a vector $\boldsymbol{v}$ is constructed describing the conversation

$$\boldsymbol{v}_i = \begin{bmatrix} p(d_1|\text{spk}_i) \\ \vdots \\ p(d_M|\text{spk}_i) \end{bmatrix}. \quad (38.26)$$

In general, the vector $\boldsymbol{v}$ will be sparse since the conversation will not contain all potential unigrams, bigrams, etc. The entries of $\boldsymbol{v}$ are then weighted with a diagonal matrix, $\mathbf{D}$, with entries $D_{j,j} = 1/\sqrt{p(d_j|\text{bkg})}$. The final kernel is an inner product,

$$K(t_1^n, u_1^m) = (\mathbf{D}\boldsymbol{v}_1)^{\mathrm{T}}(\mathbf{D}\boldsymbol{v}_2)^{\mathrm{T}}. \quad (38.27)$$

The kernel (38.25) is referred to as the term frequency log-likelihood ratio (TFLLR) kernel [38.20].

The kernel in (38.25) can be generalized in a straightforward way by analyzing its components. In (38.25), the background weighting term, $1/p(d_j|\text{bkg})$, acts as a *commonality normalization*. That is, the resulting ratios in the vector (38.26) are approximately on the same scale. This normalization can be excessive if the probability estimate $p(d_j|\text{bkg})$ is small because n-gram is infrequent. To trade off between these extremes, it is natural to use a diagonal weight of the form

$$D_{j,j} = \min\left[C_j, g_j\left(\frac{1}{p(d_j|\text{bkg})}\right)\right], \quad (38.28)$$

where $g_j(\cdot)$ is a function which shrinks the dynamic range.

The general token sequence kernel obtained by combining (38.27) and (38.28) encompasses several useful weighting types. Setting $g_j(x) = \sqrt{x}$, TFLLR kernel is obtained. Second, similar to term frequency inverse document frequency (TFIDF) in information retrieval [38.51], we can use $g_j(x) = \log(x) + 1$; we refer to this weighting as a term frequency logarithmic (TFLOG) kernel. Third, an extreme case for (38.28) is to set $C_j = 1$. This results in weighting all terms equally, $D_{j,j} = 1$ for all j. Fourth, we note that both C_j and the function g_j are dependent on j. This variable weighting may make sense if we have some idea of the confidence of our estimates in the probabilities of n-grams in the background data set.

38.4.4 System Fusion

The fusion of outputs from systems modeling low- and high-level speaker information [38.19] as well as generative- and discriminative-based classifiers [38.52] have demonstrated large accuracy improvements over the constituent input systems. If the recognizers do not behave uniformly (i. e., their errors are not completely correlated) then combining the scores may make it possible to exploit complementary information that exists in the scores.

Fusion is yet another classifier that is trained on feature vectors consisting of scores from the input systems.

Ideally, a fusion classifier would be trained for each speaker, but in practice there is often limited enrollment speech per speaker that is best utilized for training the speaker model. So speaker recognition fusion systems are typically trained in a speaker-independent fashion to optimize classification of input scores as coming from the detector hypothesis H_0 (true trial) or H_1 (false trial). Vectors of system scores are computed on a development data set consisting of true and false trials from a large population of development speaker models, aggregated, and used to train a fusion classifier.

Many different fusion classifiers have been used. At the simplest level, the fusion consists of a linear combination of the system scores. The weights used in the combination can be equal, if it is assumed that each system contributes equally to overall performance and have common numeric bias and scale ranges. The weights are usually empirically adjusted on the development data set to minimize the detection error rate.

Another common and successful fusion classifier is a simple logistic regression or single-layer perceptron. This classifier is similar to the linear combination, but has a sigmoid compression function applied to the linear combination. The weights are trained using a minimum mean-squared error criteria for the target values of 0 and 1 via a form of gradient descent [38.53] on the development data. A benefit of this type of fusion classifier is that the fusion scores are often reasonable posterior probabilities estimates that can be used as confidence values.

Fusion classifiers can also use information in addition to the system scores. For example, indicators of training and/or testing data conditions, such as speech durations, spoken language or signal-to-noise measures, can be used for explicit or implicit fusion guidance. Explicit guidance is when the indicator information is used to train and apply conditional fusion classifiers, such as fusion classifiers trained on scores from speaker models trained with specific durations of enrollment speech or for speech coming from particular languages. Conditional fusion classifiers can use different system scores, for example high-level system scores depending on an English speech recognizer may be excluded when the train and/or test speech is known to not be spoken in English. Implicit guidance is when the external information is used as another element with the system scores in the fusion vector. This approach relies on the fusion classifier to learn correlations of external measures with system scores to optimize accuracy. Implicit guidance is often inferior to explicit guidance since the fusion classifiers do not have enough complexity or enough training data to learn and represent the desired conditions.

38.5 Performance Assessment

The speaker recognition approaches described in this chapter have been subject to extensive development, and this section describes their evaluation under conditions that follow protocols established by the US National Institute of Standards and Technology (NIST). Since 1995, NIST has coordinated several evaluations with the goal of assessing the existing state of the art in speaker recognition technology. The most recent speaker recognition evaluation (SRE 2006) protocol will form the basis of the performance results presented in this section. A description of the 2006 NIST speaker recognition evaluation plan can be found in [38.54].

38.5.1 Task and Corpus

The task in the NIST SRE is the basic speaker detection task described in Sect. 38.2: determine if the speaker in the training speech is the same as the speaker in the test speech. This task is evaluated for various conditions of amount of training speech, amount of test speech, recording source (telephone or auxiliary microphones), and presence of other speakers in the speech. In all there were 15 conditions in the 2006 SRE. More details of these conditions can be found in [38.55].

The data comes from the Mixer telephone speech corpus collected by the Linguistic Data Consortium (LDC) [38.56] which consists of thousands of telephone conversations between hundreds of speakers within the US. The speakers cover a distribution of age, gender, location, and native languages. The participants in a telephone call are given general topics to discuss, but the conversations are unscripted and about five minutes in duration. Participants were encouraged to make many calls to the system over several weeks and to use varied telephone instruments and locations to provide large session and handset variability in the data. Participants

that have a common native non-English language are also instructed to conduct conversations in their native language.

For results presented here we focus on the one and eight conversation-side train and one conversation-side test telephone speech conditions. Each conversation-side provides nominally 2.5 minutes of speech. To induce mismatch, training data for a speaker comes from a single telephone number and the test speech comes from a different telephone number. Results from other conditions can be found in [38.55, 57].

Performance is shown in terms of equal error rate (EER), minimum decision cost function (minDCF) [38.54] and detection error tradeoff (DET) curves [38.58]. The minDCF is defined as

$$\text{minDCF} = \min_{\theta} 0.1\,P_{\text{miss}}(\theta) + 0.99\,P_{\text{fa}}(\theta), \qquad (38.29)$$

where $P_{\text{miss}}(\theta)$ and $P_{\text{fa}}(\theta)$ are the probability of miss and false alarm, respectively, at the detection threshold θ.

38.5.2 Systems

Below are some speaker detection systems evaluated on the 2006 SRE that exemplify the features and classifiers detailed in this chapter. More details and other system used in the 2006 SRE can be found in [38.59].

GMM UBM
Features. Mel cepstra plus delta cepstra
Classifier. 2048 mixture GMM UBM
Compensations. RASTA and mean/variance normalization in feature domain, LFA in model domain, Z-norm, and T-norm in score domain

SVM-GLDS
Features. Mel cepstra and LPC cepstra plus delta cepstra
Classifier. SVM with GLDS kernel
Compensations. RASTA and mean/variance normalization in feature domain, NAP in model domain, T-norm in score domain

SVM-GSV
Features. Mel cepstra plus delta cepstra
Classifier. SVM with generative model sequence kernel using GMM UBM mean supervectors
Compensations. NAP in model domain, T-norm in score domain

LLR-WORD
Features. Speech recognition system word sequences from word lattices
Classifier. n-gram log-likelihood ratio
Compensations. Z-norm and T-norm in score domain

SVM-WORD
Features. Speech recognition system word sequences from word lattices
Classifier. SVM TFLOG kernel using n-gram vectors
Compensations. None

38.5.3 Results

In Fig. 38.8, we show performance for the one conversation train and one conversation test condition in terms of minDCF and EER for the above individual systems, fusion of all systems and fusion of the spectral-based systems only (GMM-UBM, SVM-GLDS, and SVM-GSV). Figure 38.9 shows the same for the eight conversation train and one conversation test condition. Results for both figures use all trials from the train/test condition (e.g., both English and non-English conversations).

These results demonstrate some general observations about the various speaker recognition systems. First, we see that the spectral based systems have comparable performance to each other and significantly better performance than systems using word-based tokens. We also see that increased training data improves performance for all systems. For word-token-based systems this gain is attributable to increased instances of sparse speaker-dependent idiolect events. The improvement in

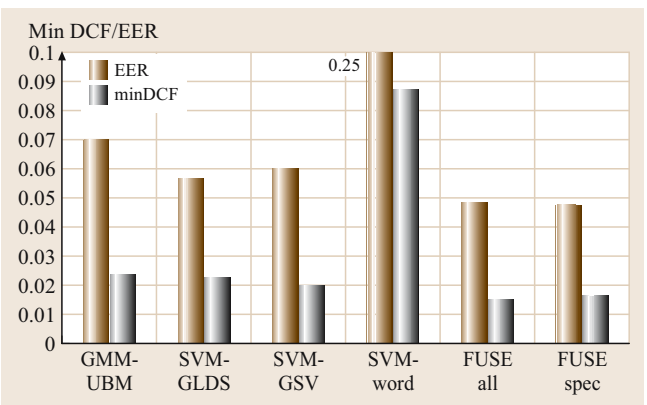

Fig. 38.8 Performance in terms of minDCF and EER for one conversation train and one conversation test for low-level, high-level, and fusion systems

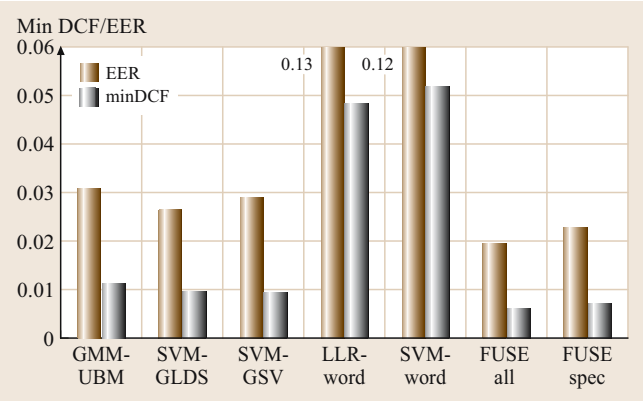

Fig. 38.9 Performance in terms of minDCF and EER for eight conversation trains and one conversation test for low-level, high-level, and fusion systems

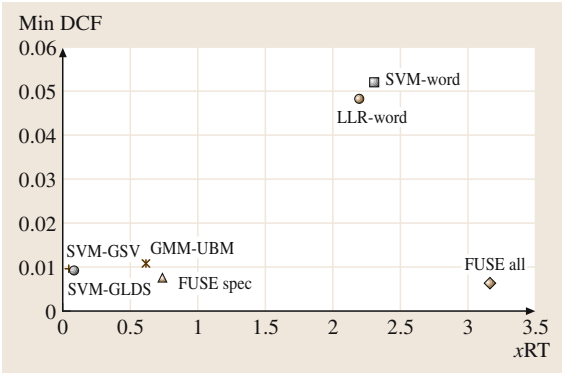

Fig. 38.10 MinDCF versus real-time factor for eight conversation trains and one conversation test for low-level, high-level, and fusion systems

spectral based systems is primarily due to observing more session variability of the speaker's speech. Lastly we observe that system fusion provides further gains in performance. This gain is seen when fusing only spectral based systems as well as when fusing spectral and word-token-based system. The extra gain from fusing in the word-token-based systems is virtually zero for the one conversation train condition and about 16% relative at EER for the eight conversation train condition.

While not shown, the results when restricting the trials to English-only conversations are only slightly better than using all trials. On average the EER is decreased by about 0.75%. An analysis of cross-lingual train/test condition indicates that spectral-based speaker recognition systems are relatively robust to the spoken language. High-level system that depend on matching the input language to the language of the speech recognizer are much more susceptible to degradations under cross-lingual conditions.

38.5.4 Computational Considerations

Although the focus of the discussion of speaker recognition systems has been performance, any practical implementation of such a system needs to account for computational complexity. A tradeoff plot of minDCF versus real-time processing factor for the above systems is shown in Fig. 38.10. The real-time processing factor is the multiple of input speech duration a system requires to produce a speaker detection score. (e.g., 2xRT means a 1 minute of speech requires 2 minutes of processing time. Real-time factors were computed using common input and CPU.) As exemplified here, since high-level systems often rely on the output of a speech recognition system, there is a significant computational cost for the incremental gains in fusing high-level systems with low-level systems. However, in applications where a speech recognition system is already being used for other purposes, reusing the output for high-level speaker recognition systems may be practical.

38.6 Summary

This chapter has presented a variety of methods for implementing automatic text-independent speaker verification systems. The speaker verification problem was cast as a general likelihood ratio detection task where features conveying speaker information from low-level spectral information to high-level word sequences are extracted and compared to models of speaker-dependent and general speaker-independent characteristics. Two of the most widely used classifiers, the generative GMM-UBM and the discriminative SVM, were presented as methods for performing the model generation and likelihood ratio comparison. To handle degradations due to channel and session variability compensation techniques in the feature, model, and score domains were discussed. Finally results from the 2006 NIST speaker recognition evaluation showed the relative performance of systems built using low- and high-level features with generative and discriminative classifiers and well as

fusions of these systems. Overall performance trends indicate that text-independent speaker verification on conversational telephone speech has shown significant improvement over years of research with state-of-the-art systems producing EERs below 2%. For the latest advances in the field of automatic language recognition, the reader is referred to the most recent proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), the Odyssey Speaker and Language Recognition Workshop, and the International Conference on Spoken Language Processing (Interspeech – ICSLP).

References

38.1 J. Naik, G. Doddington: Evaluation of a high performance speaker verification system for access Control, Proc. ICASSP (1987) pp. 2392–2395

38.2 A. Higgins, L. Bahler, J. Porter: Speaker verification using randomized phrase prompting, Digital Signal Process. **1**, 89–106 (1991)

38.3 J. Naik, L. Netsch, G. Doddington: Speaker verification over long distance telephone lines, Proc. ICASSP (1989) pp. 524–527

38.4 C. Schmandt, B. Arons: A conversational telephone messaging system, IEEE Trans. Consumer Electron. **30**(3), xxi–xxiv (1984)

38.5 L. Wilcox, F. Chen, D. Kimber, V. Balasubramanian: Segmentation of speech using speaker identification, Proc. ICASSP (1994) pp. I.161–I.164

38.6 B.M. Arons: *Interactively Skimming Recorded Speech*, Ph.D. Thesis (MIT Press, Cambridge 1994)

38.7 G. Doddington: Speaker recognition – identifying people by their voices, Proc. IEEE **73**(11), 1651–1664 (1985)

38.8 R.B. Dunn, D.A. Reynolds, T.F. Quatieri: Approaches to speaker detection and tracking in multi-speaker audio, Digital Signal Process. **10**(1), 93–112 (2000)

38.9 K. Li, J. Porter: Normalizations and selection of speech segments for speaker recognition scoring, Proc. ICASSP (1988) pp. 595–598

38.10 D.A. Reynolds, T.F. Quatieri, R.B. Dunn: Speaker verification using adapted Gaussian mixture models, Digital Signal Process. **10**(1), 19–41 (2000)

38.11 R. Auckenthaler, M. Carey, H. Lloyd-Thomas: Score Normalization for Text-Independent Speaker Verification Systems, Digital Signal Process. **10**, 42–54 (2000)

38.12 W.M. Campbell, D.A. Reynolds, J.P. Campbell, K. Brady: Estimating and evaluating confidence for forensic speaker recognition, Proc. ICASSP (2005) pp. 18–23

38.13 D. Reynolds, W. Andrews, J. Campbell, J. Navratil, B. Peskin, A. Adami, Q. Jin, D. Klusacek, J. Abramson, R. Mihaescu, J. Godfrey, D. Jones, B. Xiang: The SuperSID project: Exploiting high-level information for high-accuracy speaker recognition, Proc. ICASSP (2003) pp. 784–787

38.14 F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-Garcia, D. Petrovska-Delacretaz, D.A. Reynolds: A tutorial on text-independent speaker verification, EURASIP J. Appl. Signal Process. **4**, 430–451 (2004)

38.15 F. Soong, A. Rosenberg: On the use of instantaneous and transitional spectral information in speaker recognition, Proc. ICASSP (1986) pp. 877–880

38.16 H. Hermansky, N. Morgan, A. Bayya, P. Kohn: RASTA-PLP speech analysis technique, Proc. ICASSP (1992) pp. I.121–I.124

38.17 D.A. Reynolds: Channel robust speaker verification via feature mapping, Proc. ICASSP (2003) pp. I.53–I.56

38.18 L. Nguyen, S. Matsoukas, J. Davenport, F. Kubala, R. Schwartz, J. Makhoul: Progress in transcription of broadcast news using Byblos, Speech Commun. **38**(1-2), 213–230 (2002)

38.19 J.P. Campbell, D.A. Reynolds, R.B. Dunn: Fusing high- and low-level features for speaker recognition, Proc. European Conf. Speech Commun. Technol. (2003) pp. 2665–2668

38.20 W.M. Campbell, J.P. Campbell, D.A. Reynolds, D.A. Jones, T.R. Leek: High-level speaker verification with support vector machines, Proc. ICASSP (2004) pp. I.73–I.76

38.21 G. Doddington: Speaker recognition based on idiolectal differences between speakers, Proc. European Conf. Speech Commun. Technol. (2001)

38.22 A. Dempster, N. Laird, D. Rubin: Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. **39**, 1–38 (1977)

38.23 D.A. Reynolds: Speaker identification and verification using gaussian mixture speaker models, Speech Commun. **17**(1-2), 91–108 (1995)

38.24 D.A. Reynolds: Automatic speaker recognition using Gaussian mixture speaker models, Lincoln Lab. J. **8**(2), 173–192 (1995)

38.25 A.E. Rosenberg, J. DeLong, C.H. Lee, B.H. Juang, F.K. Soong: The use of cohort normalized scores for speaker verification, Int. Conf. Speech Lang. Process. (1992) pp. 599–602

38.26 M. Carey, E. Parris, J. Bridle: A speaker verification system using alphanets, Proc. ICASSP (1991) pp. 397–400

38.27 D.A. Reynolds: Comparison of background normalization methods for text-independent speaker verification, Proc. European Conf. Speech Commun. Technol. (1997) pp. 963–967

38.28 J.L. Gauvain, C.-H. Lee: Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains, IEEE Trans. Speech Audio Process. **2**(2), 291–298 (1994)

38.29 S. Vuuren: *Speaker Verification in a Time-Feature Space*, Ph.D. Thesis (OGI, Beaverton 1999)

38.30 D.A. Reynolds, T.F. Quatieri, R. Dunn: Speaker verification using adapted Gaussian mixture models, Digital Signal Process. **10**(1–3), 19–41 (2000)

38.31 R. Teunen, B. Shahshahani, L. Heck: A model-based transformational approach to robust speaker recognition, Proc. Int. Conf. Spoken Lang. Process. (2000)

38.32 P. Kenny, G. Boulianne, P. Dumouchel: Eigenvoice modeling with sparse training data, IEEE Trans. Speech Audio Process. **13**(3), 345–354 (2005)

38.33 R. Vogt, B. Baker, S. Sriharan: Modelling session variability in text-independent speaker verification, Proc. Interspeech (2005) pp. 3117–3120

38.34 C. Vair, D. Colibro, F. Castaldo, E. Dalmasso, P. Laface: Channel factors compensation in model and feature domain for speaker recognition, Proc. Odyssey Speaker and Language Workshop (2006)

38.35 N. Cristianini, J. Shawe-Taylor: *Support Vector Machines* (Cambridge Univ. Press, Cambridge 2000)

38.36 M. Schmidt, H. Gish: Speaker identification and verification using Gaussian mixture speaker models, Proc. ICASSP (1996) pp. 105–108

38.37 V. Wan, W.M. Campbell: Support vector machines for verification and identification, neural networks for signal processing X, Proc. 2000 IEEE Signal Process. Workshop (2000) pp. 775–784

38.38 A. Ganapathiraju, J. Picone: Hybrid SVM/HMM architectures for speech recognition, Speech Transcription Workshop (2000)

38.39 J.C. Platt: Probabilities for SV machines. In: *Advances in Large Margin Classifiers*, ed. by A.J. Smola, P.L. Bartlett, B. Schölkopf, D. Schurmans (MIT Press, Cambridge 2000) pp. 61–74

38.40 W.M. Campbell: Generalized linear discriminant sequence kernels for speaker recognition, Proc. ICASSP (2002) pp. 161–164

38.41 S. Fine, J. Navrátil, R.A. Gopinath: A hybrid GMM/SVM approach to speaker recognition, Proc. ICASSP (2001)

38.42 V. Wan, S. Renals: SVMSVM: support vector machine speaker verification methodology, Proc. ICASSP (2003) pp. 221–224

38.43 A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, A. Venkataraman: MLLR transforms as features in speaker recognition, Proc. European Conf. Speech Commun. Technol. (2005) pp. 2425–2428

38.44 W.M. Campbell, D.E. Sturim, D.A. Reynolds: Support vector machines using GMM supervectors for speaker verification, IEEE Sign. Process. Lett. **13**(5), 308–311 (2006)

38.45 R. Collobert, S. Bengio: SVMTorch: Support vector machines for large-scale regression problems, J. Machine Learn. Res. **1**, 143–160 (2001)

38.46 J. Louradour, K. Daoudi, F. Bach: SVM speaker verification using an incomplete Cholesky decomposition sequence kernel, IEEE 2006 Odyssey: The Speaker and Language Recognition Workshop (2006)

38.47 P.J. Moreno, P. Ho, N. Vasconcelos: A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In: *Advances in Neural Information Processing 16*, ed. by S. Thrun, L.K. Saul, B. Schölkopf (MIT Press, Cambridge 2004)

38.48 T.S. Jaakkola, D. Haussler: Exploiting generative models in discriminative classifiers. In: *Advances in Neural Information Processing 11*, ed. by M.S. Kearns, S.A. Solla, D.A. Cohn (MIT Press, Cambridge 1998) pp. 487–493

38.49 A. Solomonoff, W.M. Campbell, I. Boardman: Advances in channel compensation for SVM speaker recognition, Proc. ICASSP (2005)

38.50 W.D. Andrews, M.A. Kohler, J.P. Campbell, J.J. Godfrey, J. Hernández-Cordero: Gender-dependent phonetic refraction for speaker recognition, Proc. ICASSP (2002) pp. I.149–I.153

38.51 T. Joachims: *Learning to Classify Text Using Support Vector Machines* (Kluwer Academic, Dordrecht 2002)

38.52 W.M. Campbell, D.A. Reynolds, J.P. Campbell: Fusing discriminative and generative methods for speaker recogntion: experiments on Switchboard and NFI/TNO field data, Proc. Odyssey Speaker and Language Workshop (2004) pp. 41–44

38.53 L. Kukolich, R. Lippman: *LNKnet User's Guide* Manual and software available online at http://www.ll.mit.edu/IST/lnknet (2004)

38.54 *The 2006 NIST Speaker Recognition Evaluation Plan*http://www.nist.gov/speech/tests/spk/2006/sre-06_evalplan-v9.pdf (2006)

38.55 M.A. Pryzbocki, A.F. Martin, A.N. Le: NIST speaker recognition evaluation chronicles part 2, Proc. Odyssey Speaker and Language Workshop (2006)

38.56 J.P. Campbell, H. Nakasone, C. Cieri, D. Miller, K. Walker, A.F. Martin, M.A. Pryzbocki: The MMSR bilingual and crosschannel corpora for speaker recognition research and evaluation, Proc. Odyssey Speaker and Language Workshop (2004) pp. 29–32

38.57 D.E. Sturim, W.M. Campbell, D.A. Reynolds, R.B. Dunn, T.F. Quatieri: Robust speaker recognition with cross-channel data: MIT-LL results on the 2006 NIST SRE auxiliary microphone task, Proc. ICASSP (2007)

38.58 A. Martin, G. Doddington, T. Kamm, M. Ordowski, M. Pryzbocki: The DET curve in assessment of detection task performance, Proc. European Conf. Speech Commun. Technol. (1997) pp. 1895–1898

38.59 W.M. Campbell, D.E. Sturim, W. Shen, D.A. Reynolds, J. Navratil: The MIT-LL/IBM 2006 speaker recognition system: High-performance reduced-complexity recognition, Proc. ICASSP (2007)